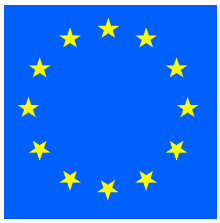
A. D. MCCXXIV

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II



Comunità Europea
Fondo Sociale Europeo

Tesi di Dottorato di Ricerca in
Ingegneria Informatica ed Automatica

XX Ciclo

Il problema del Cutting Stock bidimensionale. Un nuovo approccio algoritmico, nuovi upper bounds ed applicazioni nell'industria del vetro

Ing. Mauro Russo

Facoltà di Ingegneria

Dipartimento di Informatica e Sistemistica

Novembre 2007

Il Coordinatore
Prof. Luigi Pietro Cordella

Il Tutore
Prof. Antonio Sforza

✉ via Claudio, 21- I-80125 Napoli - ☎ *[#39]* (0)81 768 3813 - 🖷 *[#39]* (0)81 768 3816

a Lara

Ing. Mauro Russo, nato a Torre del Greco (NA) il 15/04/1976
ma.russo@unina.it rmauro76@inwind.it m.russo@uniplan.it

INDICE

INDICE DELLE FIGURE

INDICE DELLE TABELLE

RINGRAZIAMENTI

Non è facile riuscire ad esprimere con le dovute parole il ringraziamento a tutti quelli che mi sono stati vicini in questo percorso, che hanno incoraggiato il mio lavoro e che mi hanno sostenuto nei momenti difficili, per cui, nonostante la brevità di questo inciso, sappiate che il mio grazie supera, per estensione, le pagine di questa tesi e raggiunge tutti quelli che hanno condiviso con me anche solo un minuto di questa esperienza, formativa prima sotto il profilo umano e poi sotto quello scientifico, e che mi ha fornito un importante bagaglio di vita.

Desidero ringraziare per prima mia moglie Evi, che ha dovuto farsi carico fin troppe volte della mancanza del mio sostegno, impegnato com'ero talvolta fino a notte fonda. Mi ha aiutato con amore. Ringrazio la mia piccola Lara, dono di Dio sopraggiunto a metà di questo cammino. Quante volte mi ha piegato con i suoi pianti notturni, ma quante altre volte mi ha risollevato con i suoi semplici abbracci. Ringrazio i miei genitori, che hanno preso parte del mio carico, ed i miei fratelli che sicuramente hanno elevato preghiere per me. Ringrazio i miei due nipotini, Gabriele ed Alessandro, ed i miei amici, che ho troppo trascurato.

Un grazie di cuore a coloro che hanno di fatto permesso di realizzare il mio desiderio e che mi hanno dato fiducia. Mi riferisco in particolare sia al mio tutore, il Prof. Ing. Antonio Sforza, sia all'industria che ha finanziato la borsa di dottorato, che ringrazio nella persona del suo amministratore delegato, Dott. Salvatore Prete.

Al mio caro prof, per tutto quanto, dai validi consigli umani alla utile guida scientifica, dalle volte in cui mi ha ospitato a casa sua, a qualunque ora, finanche ad agosto, alle volte in cui mi ha utilmente ripreso, ma soprattutto per la pazienza con cui ha sopportato le mie centinaia di domande.

Al caro Salvatore, sempre pragmatico, ed a tutti i componenti della Uniplan Software e della Mexall Glass, che mi hanno permesso di toccare con mano un contesto applicativo delle problematiche che ho affrontato, dandomi l'opportunità di sviluppare e validare sul campo le mie idee innovative, facendomi così guadagnare la stima dei colleghi che, alle conferenze nazionali ed internazionali, mi invidiavano questa possibilità.

Al mio coordinatore, Prof. Luigi Pietro Cordella, sempre disponibile e sempre preciso nel guidare il lavoro mio e dei miei colleghi.

Grazie agli altri professori, dell'Ateneo e non, che mi hanno saputo trasmettere passione per i loro campi di ricerca, fra cui i Proff. Sansone, Picariello, Bruno, Erto, Fiorenza, Angrisani, Marano e mi fermo qui solo per brevità.

Grazie ai miei compagni di dottorato, con cui spero di continuare a crescere nella ricerca, a Claudio cui auguro di giungere a questo stesso momento con piena soddisfazione, a Simona, esempio straordinario di professionalità.

Grazie ai miei fratelli in fede, uno fra tutti Christian, che date le circostanze è stato quello che, semplicemente ascoltandomi, mi ha aiutato nei momenti più difficili.

Un caro pensiero a tutti i miei insegnanti di matematica, in ordine di tempo, Mariolina, Garofalo, Federico, Raiola, Lecciso, Cipriani, Messano, Magro, Bruno, Guidobaldi, Celentano, Bucci, Tanda, Sforza, Bruno, Fiorenza, ed anche a tutti gli altri.

Tutti voi avete fatto per me molto più di quanto richiestovi, e proprio per questo da apprezzare come inestimabile. Ma solo con le mie forze e con il vostro aiuto, non sarei arrivato a questo punto. C'è qualcuno che più di tutti, oltre ogni immaginabile limite, con la cosa giusta sempre al momento giusto, mi ha sostenuto, nonostante il fatto che proprio questo impegno mi ha talvolta portato a trascurarlo. Ma più ho speso tempo per Lui, più ne ho ricevuto indietro cento volte tanto. E' a Lui, al mio Signore e Salvatore Gesù Cristo che voglio dare il maggior ringraziamento, ed a cui voglio rendere l'onore e la gloria.

Napoli, 26 novembre 2007 Ing. Mauro Russo

INTRODUZIONE

I problemi di Cutting e Packing emergono in un'ampia gamma di problematiche del mondo reale, con particolare enfasi nel settore industriale manifatturiero. Informalmente si può dire che il problema consiste tipicamente nel disporre oggetti su fogli o rulli di materiale grezzo con il vincolo che gli oggetti non devono sovrapporsi l'un l'altro e devono essere completamente posizionati entro i limiti dei suddetti fogli o rulli. In particolare, quando si parla di Cutting, i pezzi da realizzare vengono estratti dal materiale di partenza, mentre quando si parla di Packing, è lo spazio a fare le veci del materiale, ed oggetti pre-esistenti devono essere disposti per sfruttare al meglio lo spazio.

E' facilmente comprensibile come tale tipo di problemi interessi un gran numero di settori industriali, con particolare riferimento, in termini di Cutting, alle produzioni e trasformazioni di vetro, carta, legno, metallo, all'industria tessile, oppure, in termini di Packing, dal settore della logistica a quello della produzione di circuiti integrati VLSI. Nel primo caso va sfruttato al meglio lo spazio tridimensionale dei mezzi di trasporto, mentre nel secondo va sfruttato al meglio lo spazio bidimensionale sui wafer di silicio. Questi sono fra i più importanti esempi applicativi, ma se ne potrebbero citare tanti altri. Del resto, nella maggior parte dei problemi della vita reale, il principale vincolo consiste in una limitata disponibilità di risorsa, spaziale (Packing) o materiale (Cutting), per cui quasi tutti i problemi, per un verso o per l'altro, possono ricadere nella più ampia classe di problemi di Cutting, Packing, Layout e Space Allocation. Riprendendo l'esempio applicativo di prima, un problema di Layout è quello di decidere la disposizione spaziale dei componenti di un circuito microelettronico integrato, su di un wafer di silicio, non con l'obiettivo di porne il maggior numero, dato che si è certi di poterli inserire tutti, ma con l'obiettivo di ridurre le lunghezze delle connessioni, così che i tempi di trasferimento delle informazioni fra le diverse componenti siano minimizzati. Un problema di Space Allocation è invece quello di sfruttare al meglio lo spazio in complessi istituzionali quali possono essere quelli universitari.

Sulla base dei pochi esempi descritti, è evidente che l'applicazione gioca un ruolo fondamentale sia in termini di modellazione del problema sia in termini di metodi risolutivi. Tanto per sottolineare gli aspetti più evidenti, la natura della risorsa in gioco è fondamentale, ad esempio tridimensionale nel caso dei problemi di Packing che emergono nel settore logistico, o bidimensionale nel caso dei problemi di Cutting che emergono nelle produzioni industriali di vetro, metallo o degli altri materiali sopra citati. Entrando nello specifico di questo settore, è pure evidente che il tipo di materiale ha un forte impatto sui vincoli da prendere in considerazione, dato che le tecnologie utilizzate per tagliare i materiali sono di natura verticalizzata. Ad esempio per la carta ed il vetro vi è il vincolo tecnologico di dover utilizzare solo tagli a ghigliottina, e cioè tagli che cominciano e terminano sui bordi della sagoma di materiale che viene divisa in due. Per il metallo invece questo vincolo non si presenta, ma poiché si utilizzano alte temperature per realizzare l'estrazione dei pezzi, vi sono vincoli sulla distribuzione del calore. Considerazioni di natura simile nascono nella scelta della disposizione di pezzi di vetro all'interno di un forno di tempra, che è un tipico problema di Layout.

Dall'altro lato, l'applicazione stessa determina gli obiettivi da prendere in considerazione nel problema e dunque si riflette sui metodi risolutivi. Nell'industria di trasformazione del vetro, ad esempio, i pezzi di vetro devono essere posizionati, a valle del taglio, su carrelli mobili per il trasporto interno. Data la delicatezza del vetro, sorgono vincoli sulle modalità di poggiare tali pezzi sui carrelli, soprattutto per il fatto che le dimensioni possono essere molto differenti da pezzo a pezzo. Ciò implica dei vincoli sui possibili

raggruppamenti di pezzi da associare ai carrelli, per cui nell'estrazione dei pezzi si deve tener conto anche della sequenza con cui essi vengono estratti, con particolare riferimento al caso in cui sono necessarie più lastre di materiale grezzo di partenza. Emerge dunque anche un problema noto in letteratura come Sequencing.

Nel corso dei Capitoli si farà particolare riferimento al settore applicativo del vetro e dunque al problema di Cutting di tipo bidimensionale con taglio a ghigliottina. Nondimeno in diversi punti vi sono descrizioni che fanno riferimento più in generale alla classe dei problemi di Cutting e Packing.

Le prime due parti sono dedicate al problema del Cutting Stock bidimensionale. La prima parte è introduttiva ed è costituita da due Capitoli, il primo dei quali introduce il problema, nell'ambito della classe di problemi di Cutting e Packing.

Inoltre, viene fatta una panoramica degli approcci risolutivi esistenti in letteratura, sia per problemi single-stock che per problemi multi-stock, terminando con una interessante analisi di un lavoro che tratta del problema dello Strip Packing.

Il secondo Capitolo è applicativo e descrive gli elementi tecnologici legati al caso del taglio del vetro piano, evidenziando gli aspetti che hanno maggiormente impatto sui problemi di Cutting bidimensionale, e proponendo modi originali per tenerne conto nei modelli del problema.

La seconda parte presenta i principali risultati originali ottenuti con la ricerca nel triennio di dottorato, attraverso tre appositi Capitoli.

Nel terzo Capitolo è presentato un metodo risolutivo originale per il problema del Cutting Stock, con risultati ottenuti dall'applicazione in una realtà industriale. Il quarto Capitolo presenta un miglioramento originale degli upper bounds utilizzati nella maggior parte degli algoritmi risolutivi noti in letteratura, mentre il quinto Capitolo ne dettaglia interessanti risvolti implementativi. Si sottolinea che i Capitoli dal terzo al quinto sono appositamente scritti in modo da poter essere letti singolarmente, per cui vi sono ripetuti gli elementi introduttivi indispensabili per una lettura avulsa dal contesto. Questo ha lo scopo di rendere immediatamente accessibili i principali risultati descritti in questo lavoro.

La terza parte è invece incentrata su importanti aspetti applicativi. Il sesto Capitolo descrive i principali elementi di un ambiente di produzione per la trasformazione del vetro piano, ponendo in risalto quelli correlati ad aspetti del problema di Cutting Stock. Da essi emerge la necessità di affrontare quello che è noto in letteratura come problema di Sequencing. Si sottolineano inoltre le motivazioni che rendono importante una risoluzione congiunta dei problemi di Cutting e di Sequencing.

Il settimo Capitolo introduce tale tipo di problematica, descrivendo alcuni degli approcci metodologici utilizzati in letteratura per la sua risoluzione, mentre l'ottavo Capitolo descrive gli approcci combinati per il Cutting ed il Sequencing, dettagliando anche due metodi originali proposti a riguardo.

PARTE I

Il problema del

Cutting Stock bidimensionale e

le tecnologie di taglio del vetro

Capitolo 1

Problemi di Cutting e Packing

Il Capitolo comincia con una breve panoramica sui problemi di Cutting e Packing, elencando i principali riferimenti della letteratura in relazione agli aspetti più importanti ed introducendo le principali classificazioni utilizzate per tale classi di problemi. Si evidenzia inoltre la differenza fra gli approcci risolutivi single-stock e multi-stock, sottolineando tuttavia come essi siano strettamente legati.

Il secondo paragrafo è dedicato agli approcci single-stock tratti dalla letteratura, ed ha l'obiettivo di introdurre alcuni dei metodi risolutivi di questo tipo, con particolare riferimento ai metodi cosiddetti "ad hoc", che possono a loro volta essere distinti in metodi *top-down* o *bottom-up*. Il terzo paragrafo tratta invece di alcuni metodi multi-stock, euristici, esatti e meta-euristici. Il quarto paragrafo è dedicato ad una interessante rianalisi di un lavoro sul problema dello Strip Packing bidimensionale, per il quale è emersa la necessità di integrare la relativa dimostrazione con un passo che mancava.

1.1 Problemi, modelli e metodi

La classe di problemi di Cutting e Packing è ampiamente studiata in letteratura per l'elevato numero di inerenti applicazioni industriali, e diverse riviste dedicano appositi numeri speciali all'argomento, come per esempio Dyckhoff e Wäscher (1990), Lirov (1992), Martello (1994a,b), Bischoff e Wäscher (1995), Mukhacheva (1997), Arenales, Morabito e Yanasse (1999), Wang e Wäscher (2002), Hifi (2002), Oliveira e Wäscher (2007).

Campi di applicazione sono infatti l'industria del vetro, della carta, del metallo, quella tessile, nonché in ambito di progettazione di circuiti VLSI, in riferimento al quale si parla di *layout problem*, termine peraltro non esclusivo di tale campo come dimostrano i lavori di Hifi e M'Hallah (2002) e di Degraeve, Gochet e Jans (2002). Esistono infatti differenti terminologie, talvolta dovute al campo di applicazione (Cutting, Packing, Loading, Layout, Floorplans, Space Allocation etc.) e talvolta alla fantasia degli autori, soprattutto nel coniare acronimi per le sottoclassi di problemi. Esistono comunque sforzi tesi ad una omogeneizzazione di diversi aspetti, dalle modalità di classificare i problemi, come in Dyckhoff (1990) e più recentemente in Wäscher, Haußner e Schumann (2007), a quelle di generazione di test come in Gau e Wäscher (1995) e Gradisar, Resinovic e Klijajic (2002) per il caso mono-dimensionale, o in Wang e Valenzuela (2001a, 2001b) per quello bidimensionale. Bibliografie sufficientemente dettagliate ed utili survey si trovano in Hinxman (1980), Dyckhoff e Finke (1992), Dowsland e Dowsland (1992), Sweeney e Paternoster (1992), Dyckhoff, Scheithauer e Terno (1997), Lin (1998).

La ricerca ha mostrato interesse per problemi sia mono-dimensionali che bidimensionali ed in modo crescente anche per quelli tridimensionali, come mostrato da Sweeney e Paternoster (1992), Martello, Pisinger e Vigo (2000), Padberg (2000), Lodi, Martello e Vigo (2002b), Hifi (2004), Fasano (2004, 2007). Esistono studi per il caso *m*-dimensionale come Fekete e Schepers (2004), Lins, Lins e Morabito (2002). Crescente interesse si è sviluppato per le varianti del problema che trattano pezzi con forme non rettangolari, siano esse circolari come in Wang, Huang, Zhang e Xu (2002), Hifi, Paschos e Zissimopoulos (2004) ed Hifi e M'Hallah (2007), oppure generali come in Dowsland e Dowsland (1995), Hifi e M'Hallah (2003), Stoyan, Scheithauer, Gil e Romanova (2004) e Gomes e Oliveira (2006). Una interessante survey per i problemi bidimensionali si trova in Lodi, Martello e Monaci (2002). Testi sull'argomento sono Martello e Toth (1990), Pisinger e Toth (1998) e Kellerer, Pferschy e Pisinger (2004).

Nel corso di questo lavoro si restringe l'attenzione al caso bidimensionale. Le caratteristiche comuni ai problemi di questa classe sono generalmente descrivibili nei seguenti termini: dati n tipi di pezzi rettangoli, ciascuno con una data lunghezza l_i , una data ampiezza w_i ed una data domanda d_i, date le dimensioni L (lunghezza) e W (ampiezza) dei contenitori, detti *bins* o *stocks*, scopo del problema è inserire i pezzi all'interno dei contenitori in modo ortogonale, cioè con i lati paralleli a quelli degli stocks, evitando sovrapposizione fra i pezzi. L'obiettivo da ottimizzare è, a seconda dei casi, il numero di stocks utilizzati (da minimizzare), l'area o una data funzione di utilità (da massimizzare) dei pezzi inseriti nell'unico stock disponibile.

In termini pratici una differenza fra i concetti di Cutting e di Packing è che il primo fa riferimento al caso in cui la risorsa a disposizione sia di materiale da suddividere per ottenere i pezzi, mentre il secondo è legato all'idea di una risorsa spaziale limitata da riempire con i pezzi. Infatti nel Cutting si utilizza il termine *stock* mentre nel Packing il termie *bin*. Questo comporta un maggior interesse, dal punto di vista del Cutting, per il vincolo di tagli a ghigliottina e dal punto di vista del Packing per disposizioni libere di pezzi. Qualche autore, come Gau e Wäscher (1995), esprime una ulteriore differenza pratica, secondo cui nei

problemi di Cutting vi sono pochi tipi di pezzi di diversa geometria, ciascuno con elevata domanda, mentre nel caso del Packing è esattamente il contrario, con conseguenze sulle modalità di risoluzione proposte. Tuttavia il modo in cui molti lavori della letteratura trattano l'uno o l'altro problema è intercambiabile, ed infatti i due casi *a ghigliottina* e *non a ghigliottina* sono affrontati sia per il Cutting che per il Packing.

Nel caso di ottimizzazione con un solo stock a disposizione si parla talvolta di Rectangle Packing Problem (RPP), come in Scheithauer e Sommerweiβ (1998) ed in Huang, Chen e Xu (2007). Se una delle dimensioni dello stock è illimitata, si parla di Strip Packing Problem (SPP), come in Martello, Monaci e Vigo (2003), in Seiden e Woeginger (2005) ed in Boschetti e Montaletti (2007), dove la *strip* identifica appunto una striscia di lunghezza infinita e le *shelves* sono gli strati a mensola di soluzioni a ghigliottina. In altri lavori, come Hifi e M'Hallah (2006), che trattano *stocks* di lunghezza finita, il termine *strip* è invece equivalente a quello di *shelf*. Alcuni autori affrontano problemi in cui non esiste un unico tipo di stock ed allora si parla di *multi-stock*, o *multi-length* nel caso mono-dimensionale, come in Holthaus (2002). Tutti questi problemi sono un'estensione al caso bidimensionale del problema dello zaino (*knapsack problem*).

Esiste anche una versione del problema dello zaino in termini multi-dimensionali, ma non dal punto di vista geometrico. E' il *set packing problem*, o *vector packing problem*, o *multiknapsack*, in cui ciascun pezzo ha un ingombro per ciascuna delle *m* dimensioni, tra di loro indipendenti, come in Fortin e Tsevendorj (2004). Una differente generalizzazione consiste nel caso in cui i pezzi abbiano una diversa utilità per ciascuna dimensione e le strategie proposte dipendono dalla particolare funzione obiettivo, combinazione delle utilità totali relative a ciascuna dimensione. In Hifi, M'Halla e Sadfi (2005), nel caso di *min* come funzione combinatoria, il problema è riferito come *knapsack sharing problem*. Tali estensioni sono rilevanti anche per l'industria del vetro, perché intervengono in fasi intermedie di alcuni dei metodi proposti per il problema del taglio.

Infine si accenna al fatto che molti lavori trattano problemi legati a taluni aspetti pratici delle realtà industriali sopra referenziate. Ad esempio, se si fa riferimento a tecnologie di taglio multi-fase, i relativi problemi vengono indicati come *multistage*, in cui si introducono termini di costo legati ai risultati delle fasi intermedie, come in Zak (2002a, 2002b).

Come indicato, in letteratura esistono diverse classificazioni per questa classe di problemi, fra cui Dyckhoff (1990), sufficientemente generale. In Fayard, Hifi e Zissimopoulos (1998) è presente una classificazione più vicina alla realtà dell'industria del vetro. Il problema del Cutting Stock è indicato con TDC, come sigla di Two-Dimensional Cutting. Si distinguono quattro tipi di problemi, secondo due elementi che fanno riferimento ai pezzi, e supponendo fissata la proprietà *a ghigliottina* per i tagli.

Il primo elemento riguarda l'associazione a ciascuno degli *n* tipi di pezzi una utilità che non sia l'area $l_i \cdot w_i$ ma un dato valore π_i, considerando come utilità globale la somma delle utilità dei pezzi estratti da uno stock. In presenza di una tale utilità il problema è classificato come 'pesato', o *weighted*. Qualche autore classifica il problema pesato come *knapsack loading*.

Il secondo elemento è legato alla limitazione a priori del numero massimo (domanda) di pezzi per ciascuno degli *n* tipi. In presenza di limitazione si parla di problema vincolato, o *constrained*. Tuttavia nella pratica è opportuno considerare anche modelli ibridi in cui alcuni tipi di pezzi abbiano domanda limitata ed altri no, oppure ciascun tipo abbia utilità non crescenti con il numero di pezzi estratti di quel tipo.

Le quattro combinazioni sono allora identificate come CW_TDC (*Constrained Weighted* TDC), UU_TDC (*Unconstrained Unweighted* TDC), UW_TDC e CU_TDC. Il lavoro di Hifi e Ouafi (1997) identifica invece il TDGC (Two-Dimensional Guillotine

Cutting) ed il GTDGC (General TDGC), con la possibilità aggiuntiva di avere stocks di dimensioni diverse. Tale generalizzazione è anche indicata come “multiple stock size”, come in Yanasse, Zinober e Harris (1991), oppure, nel caso mono-dimensionale, come “multiple sotck length”, come in Belov e Scheithauer (2002).

Un’altra semplice ed interessante classificazione si trova in Lodi (1999), che distingue quattro tipi di problemi, ancora secondo due proprietà.

La prima riguarda il fatto che i pezzi abbiano orientamento fisso [O] oppure sia possibile ruotarli di 90 gradi [R]. La seconda fa riferimento ai tagli, distinguendo fra il caso di taglio a ghigliottina [G] oppure libero [F], dove F sta per *free*.

Secondo tale tassonomia, i quattro tipi di problemi si indicano con 2BP|*|* dove il primo * è ‘O’ oppure ‘R’ ed il secondo è ‘G’ oppure ’F’ e 2BP sta per Two-Dimensional Bin-Packing. Tutti i quattro tipi di problemi della tassonomia di Lodi sono indicati, nella classificazione di Dyckhoff (1990) come problemi 2/V/M/I.

In termini di metodi risolutivi, le strategie dipendono soprattutto dalla limitatezza o meno del numero di pezzi per ciascuno degli *n* tipi, dato che questo si riflette sullo stesso modello del problema. Se infatti il problema è non vincolato, secondo la tassonomia di Fayard, Hifi e Zissimopoulos (1998), generalmente il problema consiste nell’ottimizzare il riempimento di un singolo stock, visto che la disposizione ottima è replicabile in un qualsivoglia numero di stock. Se invece il problema è vincolato, generalmente l’obiettivo è minimizzare il numero di stock identici da utilizzare. In questo secondo caso, gli approcci seguiti tendono a sviluppare procedure di costruzione della soluzione articolate su due livelli, uno relativo alla suddivisione dei pezzi fra gli stock e l’altro relativo al posizionamento, per ciascuno stock ed al suo interno, del sottogruppo di pezzi ad esso associato. Da notare che l’insieme *P* di pezzi associato dinamicamente ad uno stock è in genere, per costruzione, tale da essere facilmente posizionabile oppure al confine fra l’insieme degli insiemi di pezzi posizionabili e l’insieme degli insiemi di pezzi non posizionabili. Questa proprietà deriva dal fatto che *P* è in genere incrementato di un pezzo per volta, verificando ad ogni aumento l’inseribilità nello stock. Ciò vuol dire che nei metodi che seguono questo tipo di strategia, la somma delle aree dei pezzi associati ad uno stock non è mai superiore all’area dello stock stesso, mentre nel caso non vincolato si è già osservato che il problema è riempire un solo stock sfruttandone la massima parte possibile con l’insieme illimitato di pezzi a disposizione.

Negli approcci seguiti in letteratura si è dunque approfondito lo studio su aspetti diversi a seconda dei due casi. Nello studio del caso non vincolato, ad esempio, sono stati analizzati lower bounds ed upper bounds supponendo nota una sottosoluzione parziale, corrispondente al posizionamento di un dato insieme di pezzi considerati ‘fissati’, mentre nello studio del caso vincolato si è posta più attenzione all’analisi di euristiche, lower bounds ed upper bounds multi-stock, utilizzati in presenza di soluzioni parziali che consistono in un certo numero di stock ‘completi’ ed un insieme di pezzi ancora da inserire, con assenza di stock solo parzialmente riempiti. In ogni caso gli studi per il caso non vincolato sono stati, nella maggior parte dei casi, generalizzati al caso vincolato dagli stessi autori, essendo utili in istanze del problema in cui l’area totale dei pezzi che si intende associare ad uno stock risulta superiore all’area dello stock stesso. In tale prospettiva gli approcci con obiettivi multi-stock sono una generalizzazione di quelli single-stock, poiché questi ultimi possono intervenire esplicitamente nelle fasi intermedie proprie dei metodi risolutivi adottati nei casi multi-stock. Le due fasi, di suddivisione dei pezzi fra gli stock e di relativo riempimento degli stessi, possono essere, nei due casi estremi, in alternativa, o esplicitamente divise con eventuali feedback, oppure realizzate contemporaneamente con una strategia combinata, come nell’algoritmo di *column generation* di Gilmore e Gomory (1961, 1963), di cui si daranno dettagli nell’ultimo Capitolo. Un esempio del primo estremo, peraltro studiato in letteratura soprattutto nel caso mono-dimensionale, è quello in cui nella prima fase si generano con

strategie single-stock tutti o comunque i più promettenti schemi di taglio single-stock e nella seconda fase si affronta un problema di *vector packing* per scegliere quali schemi realizzare, sulla base della funzione obiettivo e tenendo conto dei vincoli di domanda, considerando nel vector packing una "dimensione" per ciascun tipo di pezzi.

Come esempio dell'altro estremo, oltre ai suddetti lavori di Gilmore e Gomory, si cita il lavoro di Martello e Vigo (1998) in cui si incrementa, in una procedura ad albero esaustiva, di uno per volta la cardinalità degli insiemi di pezzi associati a ciascuno stock.

D'altro canto, durante il processo risolutivo in un approccio single-stock, con riferimento ai problemi a ghigliottina, si può essere di fronte ad una soluzione parziale che prevede uno schema parziale di taglio che suddivida lo stock in sottoparti diseguali. In tali condizioni emerge il problema di come riempire i k sottorettangoli residui, cioè un problema multi-stock con stock di varie dimensioni e di disponibilità finita. Queste considerazioni mostrano lo stretto legame che esiste fra i due tipi di problemi.

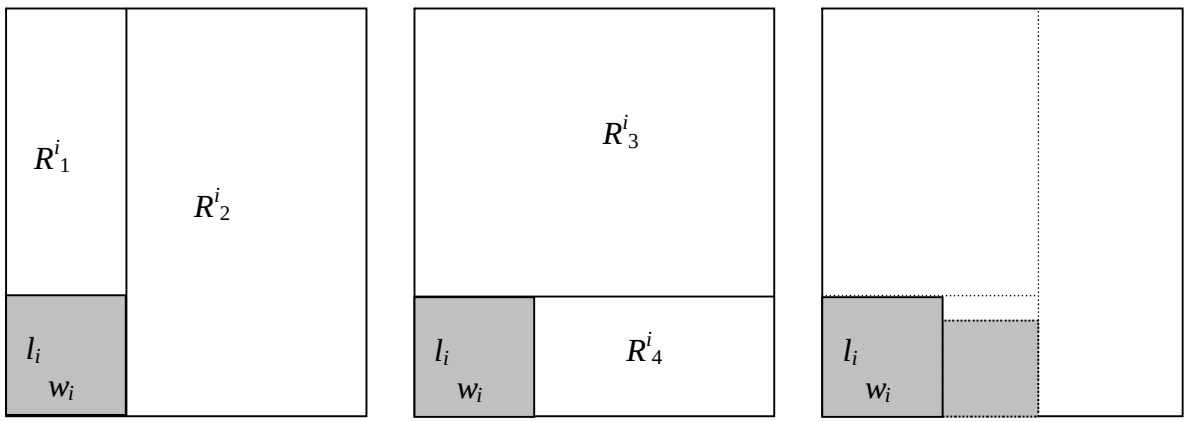
Nel seguito si descrivono alcuni degli approcci single-stock, che cioè hanno l'obiettivo di ottimizzare il riempimento di un solo stock, salvo estendere l'uso delle tecniche e dei risultati al caso in cui l'obiettivo sia la minimizzazione del numero di stock, generalmente affrontato con quelli che saranno di seguito descritti come approcci multi-stock. L'elenco dei metodi risolutivi presentati non ha la pretesa di essere esaustivo e l'obiettivo è semplicemente quello di dare un'idea dell'ampia varietà di tecniche proposte in letteratura.

1.2 Approcci single-stock in letteratura

Nei primi paragrafi vengono descritte delle strategie euristiche, propedeutiche per inquadrare le caratteristiche fondamentali dell'algoritmo meta-euristico proposto da Alvarez-Valdés, Parajòn e Tamarit (2002), fra i migliori noti in letteratura. Si passa poi alla descrizione dei due tipi di approccio *ad hoc* più utilizzati in letteratura per il problema del Cutting Stock bidimensionale, vale a dire l'approccio *bottom-up* e l'approccio *top-down*, indicandone gli elementi essenziali attraverso la descrizione di alcuni dei principali lavori in letteratura ad essi relativi.

1.2.1 Tecniche euristiche costruttive

Si tratta di approcci adatti a problemi a ghigliottina, che inseriscono un pezzo per volta senza possibilità di fasi di feed-back. Supponendo di avere, per tagli già effettuati, un rettangolo R di dimensioni L e W, può essere importante capire quale pezzo posizionare per primo in R ed a tal fine è utile stimare una promettenza della scelta. Se il problema è a ghigliottina, l'aggiunta di un pezzo di dimensioni l_i e w_i genera due probabili scenari, anche se ve ne sono altri possibili, visto che il taglio che realmente estrae definitivamente il pezzo può essere più innestato, come si rappresenta in Figura 1.1 con lo scenario c).



Scenario a) Scenario b) Scenario c)

Figura 1.1 Scenari per l'inserimento di un pezzo

La promettenza è calcolata valutando un upper bound del valore ottimo, somma delle utilità dei pezzi utilizzati nelle soluzioni ottenibili in ciascuno dei due scenari a) e b). Per ciascuno dei quattro rettangoli R^i_k , di dimensioni (L^i_k , W^i_k), si calcola un upper bound, attraverso un algoritmo greedy proposto da Martello e Toth (1990) per il problema dello zaino (mono-dimensionale), utilizzando cioè un rilassamento combinatorio che consiste in un passaggio alla mono-dimensionalità.

In primo luogo si fissa il valore u_j per $j = 1, \ldots, n$, cioè per ciascuno dei tipi di pezzi, secondo il numero b_j di pezzi rimasti (se il problema è vincolato) e secondo ciascuna delle dimensioni del rettangolo R^i_k, calcolando u_j con l'espressione $\min\{ b_j , \lfloor L_k / l_j \rfloor \cdot \lfloor W_k / w_j \rfloor \}$. Esso rappresenta il massimo numero di pezzi di tipo j inseribili nel rettangolo R^i_k.

A partire da $j = 1$, ed inizializzando l'area residua c al valore $L_k \cdot W_k$, si calcola il valore $x_j = \min\{ u_j , \lfloor c / s_j \rfloor\}$, dove s_j è l'area dei pezzi del j-mo tipo di pezzo, aggiornando

inoltre $c = c - x_j \cdot s_j$. La cosa importante, per ottenere un upper bound valido, è pre-ordinare i tipi di pezzi secondo valori non crescenti del rapporto π_j / s_j , cioè dell'utilità specifica, essendo π_j l'utilità assoluta del j-mo tipo di pezzo. Inoltre, se il valore finale dell'area residua c è non nullo, allora bisogna aggiungere all'upper bound il valore della massima utilità specifica fra i pezzi rimasti, moltiplicata per la stessa area residua. Un'alternativa, che però non produce un upper bound valido, ma semplicemente una stima sulla promettenza dello scenario, è quella di non aggiungere tale ultimo addendo, verificando però anche quale sia il massimo valore fra i prodotti $\pi_j \cdot u_j$, corrispondenti a sottosoluzioni fattibili per il rettangolo R^i_k, costituite da pezzi di un solo tipo, disposti a matrice, e per questo dette "omogenee". La promettenza che se ne ricava è nel seguito indicata con $BK1_i(R)$.

Hifi e Ouafi (1997) hanno proposto, per ciascuno dei due scenari, di risolvere in modo esatto il problema dello zaino mono-dimensionale per il rettangolo di area maggiore e generare la migliore soluzione omogenea per il rettangolo di area minore, generando così un valore di promettenza che però non è un upper bound valido, a causa della semplificazione sul rettangolo minore.

Un raffinamento della stima si può ottenere eliminando l'idealità nell'inserimento dei pezzi di utilità specifica massima, che senza perdere di generalità possono essere indicati in numero di x_1. L'unico modo per inserire realmente tali pezzi in un fissato R^i_k è disporne una matrice di dimensioni $m_{1l} = L_k / l_1$ ed $m_{1w} = W_k / w_1$. Tale strategia di calcolo prevede allora di sostituire, nel calcolo di u_j con $j > 1$, il rapporto $\lfloor c / s_j \rfloor$ con il valore $\max\{\lfloor L_{k,1} / l_j \rfloor \cdot \lfloor W_{k,1} / w_j \rfloor + \lfloor L_{k,2} / l_j \rfloor \cdot \lfloor W_{k,2} / w_j \rfloor , \lfloor L_{k,3} / l_j \rfloor \cdot \lfloor W_{k,3} / w_j \rfloor + \lfloor L_{k,4} / s_j \rfloor \cdot \lfloor W_{k,4} / w_j \rfloor \}$ dove $(L_{k,1}, W_{k,1})$, …, $(L_{k,4}, W_{k,4})$ sono le dimensioni dei quattro rettangoli generati nei due possibili scenari determinati dall'inserimento a matrice di x_1 pezzi di tipo 1 nel rettangolo R^i_k, come rappresentato in Figura 1.2. La stima così ottenuta è indicata con $BK2_i(R)$.

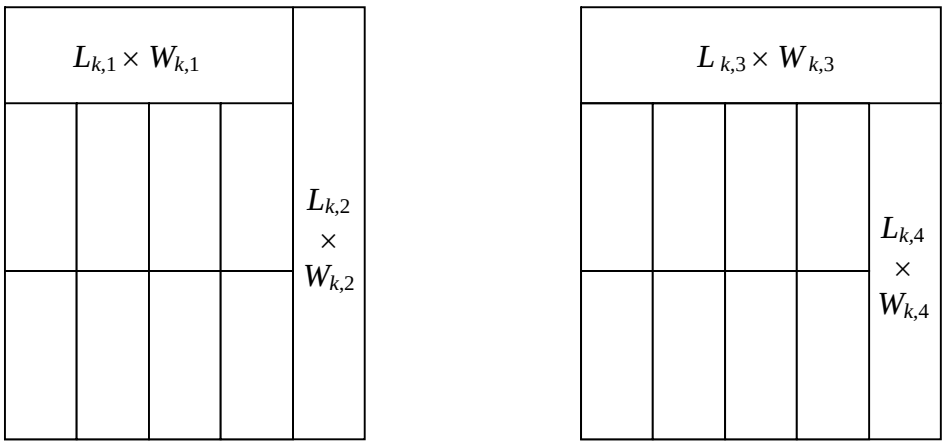


Figura 1.2 Scenari per il calcolo di $BK2_i(R)$

L'algoritmo euristico procede dunque secondo il seguente schema. Detto *RR* l'insieme dei rettangoli residui rimasti, se ne estrae quello di area minima R_h , calcolando per ciascun tipo i di pezzi, per il quale sia rimasto almeno un pezzo, la somma delle due stime ottenute rispettivamente sulla coppia di rettangoli (R^i_1, R^i_2) ed (R^i_3, R^i_4). A seconda che si utilizzi la strategia per BK1 o quella per BK2 si ottiene un algoritmo euristico diverso. La scelta ricade sul tipo i che massimizza il massimo delle due somme, operando poi il taglio corrispondente allo scenario più promettente fra i due. I due sottorettangoli così ottenuti vengono inseriti in

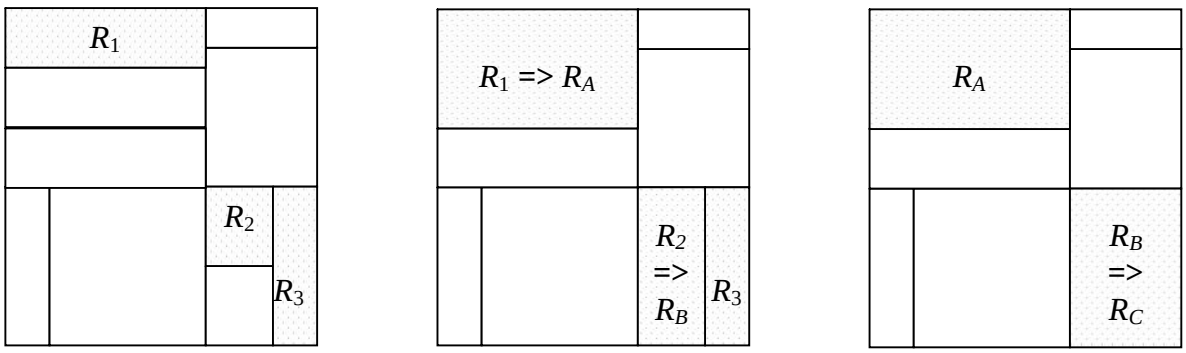
RR al posto di R_h . Se però in R_h non si può inserire nessun pezzo, allora R_h è aggiunto agli scarti. Si prosegue finché *RR* non diventa vuoto.

1.2.2 Algoritmo GRASP

La sigla GRASP sta per Greedy Randomized Adaptive Search Procedure (Feo, Resende e Smith, 1994; Feo e Resende, 1995). A differenza dell'algoritmo euristico costruttivo descritto nel precedente paragrafo, la scelta del pezzo da inserire nel rettangolo residuo R_h estratto, viene fatta con una strategia random, scegliendo non l'indice *i* cui corrisponde la massima stima B_i ($BK1_i$ o $BK2_i$ a seconda dei casi), ma uno degli indici dell'insieme $S_{max} = \{p : B_p \geq \delta \cdot B_i\}$, per cui se la costante δ vale 1, allora si ricade nel caso del paragrafo precedente.

L'algoritmo GRASP prevede però ulteriori passi di miglioramento della soluzione. Una volta riempito uno schema di taglio, a partire dallo stock iniziale, si provvede a fondere, se possibile, ciascuno scarto con il pezzo, o l'insieme di pezzi posti in posizione aderente allo scarto, eliminando tali pezzi e generando così delle nuove aree utilizzabili, che però vengono preventivamente ed ulteriormente aumentate fondendole con gli scarti perfettamente aderenti su qualche lato, se presenti. In Figura 1.3 si riportano le tre parti di una tale fase di restart, dove R_1, R_2 ed R_3 sono gli scarti da cui si parte.

Dopo ciascun restart, a partire dalle nuove aree residue si opera un nuovo ciclo di posizionamento dei pezzi, verificando ad ogni fine ciclo se la nuova soluzione è la migliore ottenuta e terminando quando si verifica un dato criterio di stop. Nell'esempio di figura, a partire dallo scarto R_1 si costruisce l'area libera R_A, mentre a partire dallo scarto R_2 si costruisce R_C, passando per R_B, che poi si estende ad R_C annettendo lo scarto R_3.



a) situazione iniziale b) espansione degli scarti c) ulteriore aggregazione

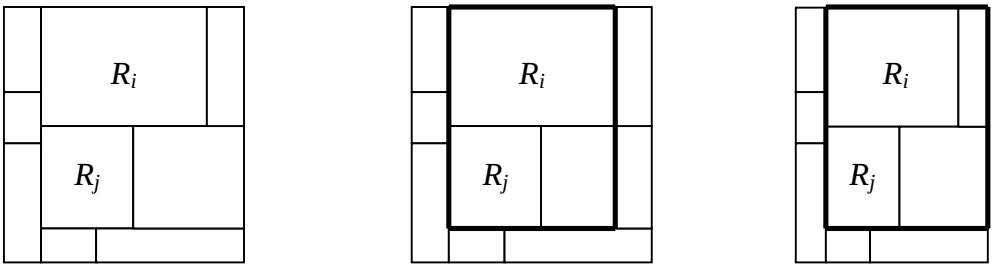
Figura 1.3 Fasi di restart fra i cicli del GRASP

1.2.3 Approccio Tabu Search

Come premessa generale, la tecnica Tabu Search (Glover e Laguna, 1993, 1997), è di tipo euristico ed è nata per superare i limiti di una pura strategia di ricerca locale poiché, per grosse linee, permette di superare i punti di estremo locale attraverso l'ausilio di *tabu lists* con cui si evita l'eventualità di tornare in punti dello spazio delle soluzioni recentemente visitati.

Nel caso del problema in esame, data una soluzione, corrispondente ad uno schema di taglio, bisogna definire cosa significhi operare una 'mossa', cioè quali operazioni realizzare per passare da una soluzione ad un'altra "vicina". Una delle scelte adottate in letteratura è la seguente. Dati due rettangoli (scarti o singoli pezzi) R_i ed R_j , almeno parzialmente a contatto, si considera la minima area rettangolare che li contiene, poi successivamente espansa per

gradi così da contenere completamente anche gli altri rettangoli parzialmente coperti, come nell'esempio in Figura 1.4.



a) selezione di R_i ed R_j b) area minima che li contiene c) espansione minima

Figura 1.4 Generazione area minima

Sull'area così ottenuta viene eseguita la strategia GRASP, con un numero massimo di cicli ridotto visto che l'intera strategia Tabu Search richiede il lancio della procedura GRASP per un elevato numero di volte. L'intorno di soluzioni esplorate è definito in prima battuta da tutte le coppie di rettangoli (R_i , R_j) almeno parzialmente aderenti. Tuttavia non vengono selezionate coppie che generano un'area minima che fosse in partenza priva di scarti. Inoltre vengono registrate le ultime k soluzioni, con k randomicamente ed inizialmente scelto nell'intervallo [0.5 m*, 1.5 m*], dove m* è la radice quadrata del numero di pezzi. Vengono quindi scartate le mosse che determinano una soluzione ottenuta nei precedenti k passi. Ad ogni variazione della soluzione, considerate tutte le coppie (R_i , R_j) come detto prima, viene scelta la mossa che determina la soluzione migliore.

Per ottenere un elevato grado di diversificazione, è conveniente scegliere a caso un R_i per poi verificare tutte le coppie che coinvolgono il rettangolo R_i , piuttosto che verificare tutte le possibili coppie.

Strategie sia di intensificazione che di diversificazione sono ottenute nel seguente modo. Si registrano le q soluzioni migliori ottenute, valutando la frequenza con cui è presente ciascun tipo i di pezzo nelle q soluzioni, secondo l'espressione

$$c_i = \sum_{k=1,\ldots,q} f_{i,k} / M$$

dove $f_{i,k}$ è il numero di pezzi di tipo i presente nella k-ma soluzione ed M è il numero totale di pezzi contenuti considerando l'insieme delle q migliori soluzioni.

Nelle procedure costruttive su indicate e, come detto, utilizzate nella realizzazione della strategia GRASP, si valutano i BK_i considerando non le utilità π_i, ma le utilità modificate, secondo il parametro β, pari a

$$\pi^*_i = \pi_i \cdot (1 + \beta \cdot c_i)$$

Un valore positivo di β determina una strategia di intensificazione poiché facilità l'utilizzo dei pezzi più presenti nelle soluzioni migliori, mentre con $\beta < 0$ si ottiene il comportamento opposto e dunque strategie di diversificazione.

1.2.4 Altre tecniche costruttive

Altri tipi di algoritmi costruttivi, per i casi non a ghigliottina, operano sulla base di un preordinamento dei pezzi, verificando di volta in volta se il pezzo successivo è inseribile o

meno ed inserendolo senza ulteriori considerazioni qualora lo sia. Tali algoritmi sono identificati dalla scelta della posizione, quando possibile e non univoca, per il pezzo successivo. Si riportano nel seguito i casi dell'algoritmo Bottom Left e dell'algoritmo Difference Process, indicati rispettivamente con le sigle BL e DP.

Algoritmo BL

L'algoritmo BL pone il pezzo corrente nella posizione più in basso possibile e poi più a sinistra possibile, come indicato in Figura 1.5.

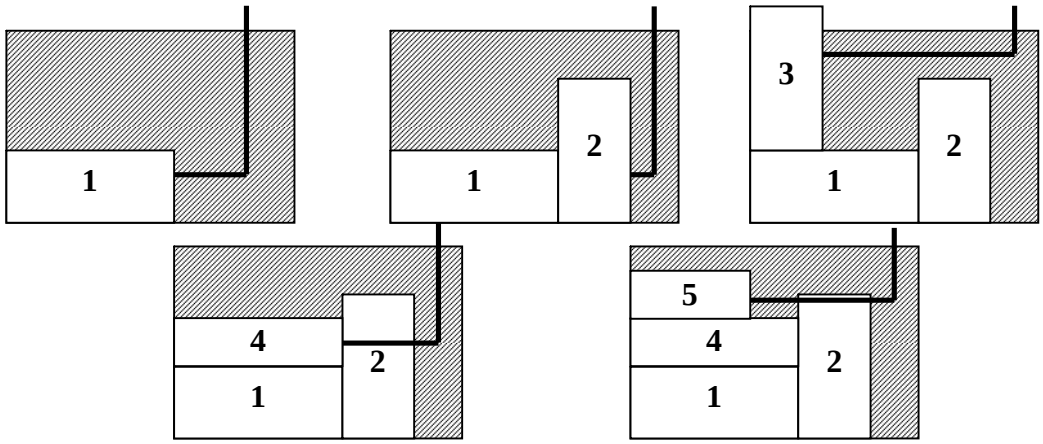


Figura 1.5 Algoritmo Bottom Left

Nel primo passo si inserisce il pezzo di indice 1 all'angolo in basso a sinistra. Il pezzo di indice 2 può poi essere posto con il suo vertice in basso a sinistra sul lato destro del pezzo di indice 1, e cioè più in basso possibile. Il pezzo di indice 3 non può essere posto né a destra del pezzo di indice 2, né sopra il pezzo di indice 1. Il pezzo di indice 4 non può essere posto a destra del pezzo 2, per cui la scelta è ripiegata nella posizione sopra il pezzo di indice 1. analogamente per il epzzo di indice 5.

Algoritmo DP

L'algoritmo DP opera sistemando ciascun nuovo pezzo nella posizione più vicina possibile all'origine (angolo in basso a sinistra dello stock iniziale), fra tutte quelle compatibili con la dimensione del nuovo pezzo, affinché non si sforino i limiti dettati dalle dimensioni dello stock. Nella Figura 1.6 si indicano con un punto ben marcato le posizioni possibili per eventuali pezzi successivi, noti in letteratura come Estreme Points (Perboli, Crainic e Tadei, 2006).

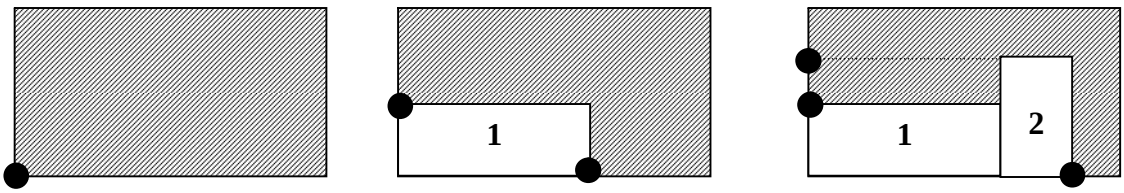


Figura 1.6 Extreme Points

Né l'algoritmo BL né l'algoritmo DP sono in grado di generare schemi come quelli mostrati in Figura 1.7, in cui si riportano le dimensioni dei pezzi.

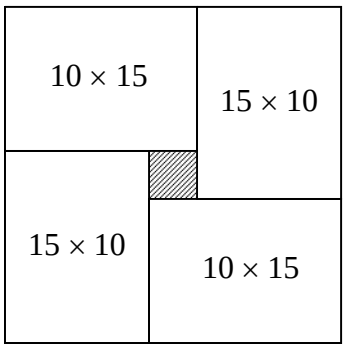


Figura 1.7 Schema non ottenibile

1.2.5 Approccio bottom-up

Wang (1983) ha proposto algoritmi per sviluppare tutti i possibili schemi di taglio, secondo una strategia bottom-up. Infatti non parte dallo stock per scegliere i tagli e ricavare i pezzi, bensì parte dai pezzi, fondendoli in rettangoli crescenti che rientrino nei limiti dimensionali dello stock. Più nel particolare, dati i pezzi $P_1,\ldots,P_n$, li si fonde a due a due generando sia l'agglomerazione orizzontale che quella verticale, come indicato in Figura 1.8, dove si mostra un esempio generalizzato a partire da due rettangoli A_1 ed A_2 che possono essere il risultato di una precedente fusione.

build orizzontale: $(l_1+l_2, \max\{w_1, w_2\})$ *build* verticale: $(\max\{l_1,l_2\}, w_1+w_2)$

Figura 1.8 Modalità di fusione

L'agglomerazione dei pezzi genera un insieme di aree rettangoli (dette *builds* o meta-rettangoli) di livello uno, denotato con $F^{(1)}$, che insieme agli stessi rettangoli iniziali, corrispondenti ai pezzi, costituiscono l'insieme arricchito di livello uno, denotato con $L^{(1)}$, mentre l'insieme iniziale è denotato con $L^{(0)}$. Al passo successivo ciascuna area di $F^{(1)}$ è 'ricombinata' con una qualunque altra area di $L^{(1)}$, sviluppando così sia l'insieme $F^{(2)}$ che l'insieme $L^{(2)} = L^{(1)} \cup F^{(2)}$. La generazione dei *builds* viene iterata, avendo cura ogni volta di eliminare le aree equivalenti da ciascun $L^{(k)}$ nonché quelle che contengono almeno un tipo di pezzo in numero eccessivo rispetto alla domanda corrispondente, o che non rispettino le dimensioni dello stock a disposizione. La procedura termina quando si ottiene un $L^{(k)}$ equivalente al precedente $L^{(k-1)}$ e si setta $M = k - 1$.

L'algoritmo prevede, per evitare una quantità di calcolo eccessiva, di eliminare le aree con una quantità di scarto superiore al valore $\beta_1 \cdot L \cdot W$, dove L e W sono le dimensioni iniziali dello stock e β_1 è una costante con valore nell'intervallo [0,1]. Eliminarle vuol dire considerarle come una soluzione finale (estendendo con scarti in alto ed a destra per

raggiungere le dimensioni dello stock) senza però utilizzarle in aggregati di livello superiore. Se β_1 vale 1, allora non c'è alcun limite, mentre quando β_1 tende a 0, si riduce il tempo di calcolo dell'algoritmo, fino ad imporre, se esistono, solo aree aggregate senza scarti, quando $\beta_1 = 0$. In genere conviene lanciare la procedura con bassi valori di β_1, salvo poi stimare la bontà della soluzione ottenuta sulla base di un upper bound. Comunque un teorema dimostrato da Wang viene in aiuto nella stima. Detta σ la quantità di scarto totale, ottenuta cioè a valle dell'espansione di un rettangolo *R*, di dimensioni *l* e *w*, alle dimensioni dello stock *L* ed *W*, e detta ω la quantità di scarto nel solo rettangolo *R*, risulta $\sigma = W \cdot L - (w \cdot l - \omega)$.

Per ciascun fissato valore di β_1, si denota con σ^* il valore minimo possibile fra quelli associati a tutte le aree di $L^{(M)}$. Certamente esiste un valore minimo di β_1, denotato con β_1^*, capace di generare almeno una soluzione che minimizzi σ al valore σ^*.

Se ω_1^* è il valore corrispondente a β_1^*, definito cioè come $\beta_1^* \cdot W \cdot L$, allora risulta $\sigma^* \geq \omega_1^*$. Infatti non può essere $\sigma^* < \beta_1^* \cdot W \cdot L$, poiché altrimenti la stessa soluzione ottima sarebbe ottenibile con un valore inferiore a β_1^* che dunque non sarebbe il minimo. L'uguaglianza fra σ^* e ω_1^* si ottiene solo se, lanciato l'algoritmo con $\beta_1 = \beta_1^*$, l'aggregato corrispondente alla soluzione ottima risulta avere esattamente le dimensioni *L* e *W*, poiché in caso contrario l'estensione dello scarto determina un σ^* maggiore, né è possibile che prima dell'estensione si avesse $\sigma^* < \omega_1^*$, poiché ciò contraddirebbe nuovamente il fatto che β_1^* è il minimo. Un tipico diagramma di σ in funzione di β_1 è riportato in Figura 1.9 ed è molto utile per comprendere le proprietà discusse in seguito.

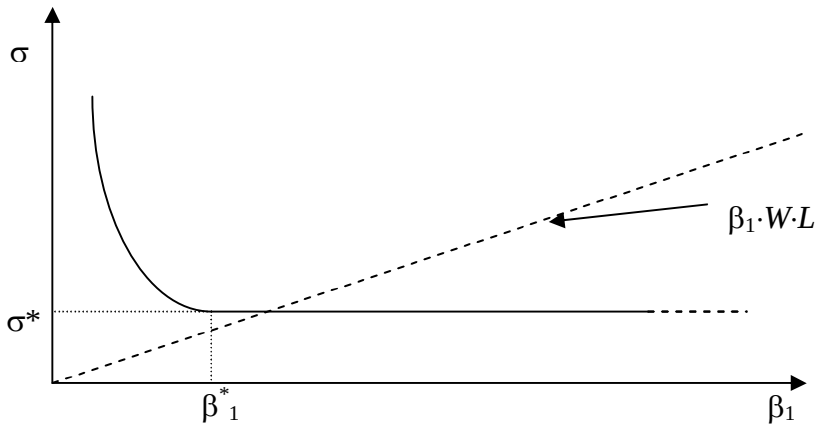


Figura 1.9 Andamento di σ in funzione di β_1

La prima importante proprietà è che $\sigma - \sigma^* \leq | \sigma - \beta_1 \cdot W \cdot L |$.
Se infatti $\beta_1 < \beta_1^*$, allora $\sigma > \sigma^* \geq \beta_1 \cdot W \cdot L$, dalla definizione di β_1^*. Di conseguenza, si ricava $0 < \sigma - \sigma^* \leq \sigma - \beta_1 \cdot W \cdot L$. Se invece $\beta_1 \geq \beta_1^*$, allora $\sigma = \sigma^*$ ed allora la proprietà equivale a $|\sigma - \beta_1 \cdot W \cdot L| \geq 0$, chiaramente valida. Tale proprietà implica un limite alla differenza fra la soluzione trovata e quella ottima. Tuttavia per valori di β_1 elevati può non essere indicativa, ma per tali casi c'è un semplice test per essere certi che valga $\beta_1 \geq \beta_1^*$ e dunque $\sigma = \sigma^*$. La condizione $\sigma \leq \beta_1 \cdot W \cdot L$ è infatti sufficiente ad implicare che $\beta_1 \geq \beta_1^*$. Ciò può essere ricavato osservando il diagramma in Figura 1.9 e può essere formalizzato mostrando che se, per assurdo, risultasse $\beta_1 < \beta_1^*$, allora si ricaverebbe $\sigma < \beta_1^* \cdot W \cdot L \leq \sigma^*$, contro il fatto che σ^* è il valore minimo possibile.

Un corollario derivante da questa seconda proprietà è che se il rettangolo *R* corrispondente alla soluzione ottima ha proprio le dimensioni dello stock, allora la soluzione è ottima e ciò perché in tal caso σ è automaticamente non superiore a $\beta_1 \cdot W \cdot L$ per il modo stesso di operare dell'algoritmo.

Una versione modificata dell'algoritmo prevede l'utilizzo del parametro β_2 al posto di β_1. Esso è utilizzato con un significato diverso. Più nello specifico, qualunque area in $L^{(k)}$ deve essere tale che la sua porzione di scarti deve essere non superiore a β_2. Si tratta dunque di un controllo che limita fin da subito le aree che, in prospettiva, se mantengono la loro porzione di scarti, genereranno aree totali non buone secondo la soglia β_2.

Per questo secondo algoritmo, che a differenza del primo utilizza β_2 e non β_1, si operano le seguenti definizioni. Si indica con β_2^* il minimo valore di β_2 che determina almeno una soluzione ottima. Ad esso corrisponde il valore ω_2^* associato all'area R^*. Se R^* è costruito per passi attraverso le aree $R_1, R_2, \ldots, R_t = R^*$, allora, detti $\omega_1, \omega_2, \ldots, \omega_t$ le rispettive quantità di scarti associati, risulta $\omega_2^* = \omega_t$ e $\beta_2^* = \max_{1 \le i \le t}\{\omega_i / \text{area}(R_i)\}$. Ne consegue che $\sigma^* \ge \omega_2^* \ge_{\forall p \in \{1,\ldots,t\}} \omega_p = \beta_2^* \cdot \text{area}(R_p) \ge \beta_2^* \cdot \text{area}(\rho)$ dove ρ è il minimo rettangolo (*build*) generato per agglomerazione verticale o orizzontale a partire da due pezzi qualsiasi P_i ed P_j, anche uguali, purché ne siano disponibili almeno due.

Valgono dunque analoghe proprietà rispetto all'algoritmo che utilizza β_1, dimostrabili allo stesso modo, salvo considerare in modo differente il caso in cui $R^* = P_i$ per qualche *i*. In tal caso la soluzione ottenuta è certamente quella ottima indipendentemente dal valore scelto per β_2 essendo $\beta_2^* = 0$.

Le due proprietà sono:

a) $\sigma - \sigma^* \le |\sigma - \beta_2 \cdot \text{area}(\rho)|$

b) la condizione $\sigma \le \beta_2 \cdot \text{area}(\rho)$ implica che la soluzione trovata è quella ottima.

Non vale tuttavia un corollario analogo a quello indicato per l'algoritmo che utilizza β_1.

Da un punto di vista pratico, per ottenere buoni valori di β_1 o β_2, basta utilizzare una buona euristica che generi una soluzione con valore degli scarti pari a σ, per poi sfruttare le proprietà:

$$\text{a)}\ \ \beta_1^* \le \sigma / (L \cdot W) \qquad , \qquad \text{b)}\ \beta_2^* \le \sigma / \text{area}(\rho)$$

Se l'euristica è buona, la differenza fra i membri sinistri e destri è bassa e si riesce a scegliere un valore di β_1 (risp. β_2) molto prossimo a β_1^* (risp. β_2^*).

L'approccio bottom-up descritto è facilmente generalizzabile per qualunque tipo di problema che rimanga non pesato, sia esso orientato o meno e vincolato o meno. Lo stesso vale per le evoluzioni della strategia che vengono descritte nei due paragrafi successivi. Si sottolinea che la generalizzazione al caso pesato può essere ottenuta sostituendo allo scarto la "perdita di utilità" rispetto ad un upper bound valutabile ad esempio rilassando il problema e rendendolo mono-dimensionale, vale a dire considerando solo le aree dei pezzi e dello stock.

Una generalizzazione dell'approccio bottom-up di Wang al caso di problemi non a ghigliottina può essere trovato in Pan, Shi e Liu (1996). La generalizzazione dell'approccio consiste nel considerare agglomerazioni fra 5 rettangoli secondo lo schema di Figura 1.10. Il lavoro di Scheithauer e Sommerweiβ (1998) pure utilizza una strategia simile, fondendo però fino a quattro rettangoli, cioè con un rettangolo R5 vuoto rispetto alla Figura 1.10.

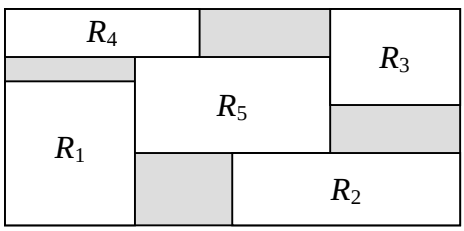


Figura 1.10 Agglomerazione generalizzata

1.2.6 Algoritmo bottom-up modificato

La prima modifica è stata proposta da Oliveira e Ferreira (1990), similmente a quanto indipendentemente proposto in Vasko (1989). Obiettivo della modifica è ridurre la quantità di calcoli tramite la riduzione del numero di aree aggregate generate. A tal fine, per ciascuna potenziale area aggregata *R*, si calcola un lower bound dell'area di scarti che si può ottenere utilizzando *R*. Per valutare tale lower bound, si risolvono quattro problemi non vincolati per le due coppie di rettangoli ottenute con la stessa logica dell'algoritmo euristico costruttivo cui faceva riferimento la Figura 1.1, ipotizzando cioè i due possibili e più semplici scenari per l'aggiunta di una coppia di tagli a ghigliottina. Detta $T_m(a, b)$ la minima area di scarti idealmente ottenibile per il problema non vincolato su un rettangolo di dimensioni *a* e *b*, se *L* e *W* sono le dimensioni dello stock ed (L_R, W_R) le dimensioni di *R*, allora si valuta

$$T = \min\{T_m(L, W - W_R) + T_m(L - L_R, W_R), T_m(L - L_R, W) + T_m(L_R, W - W_R)\}$$

ed il minimo scarto ottenibile con l'utilizzo di *R* è $T + T_R$ dove T_R è l'area scartata in *R*.

Poiché, come analogamente spiegato in relazione alla Figura 1.1, sono possibili altri scenari, *R* non va necessariamente scartato qualora $T + T_R < \beta_1 \cdot W \cdot L$ (o $T + T_R < \beta_2 \cdot \text{area}(R)$ per la seconda versione dell'algoritmo). Tuttavia si può scegliere di eliminare *R*, ottenendo con ciò un algoritmo maggiormente incompleto, poiché non garantisce di trovare la migliore soluzione con scarto limitato dal valore $\beta_1 \cdot W \cdot L$. Un'altra strategia consiste nell'utilizzare l'espressione

$$T = T(L_R, W_R) = T_m(L, W - W_R) + T_m(L - L_R, W)$$

stima nella quale l'area residua in alto a destra è considerata in entrambi gli addendi, così da considerare le due più grandi aree residue possibili.

La semplificazione computazionale si basa sulla possibilità di calcolare una volta per tutte una tabella iniziale di valori $T_m(x, y)$ attraverso una tecnica di programmazione dinamica, che permette di generare i valori di T_m per tutti i valori significativi di (x, y), ottenibili cioè dalle somme delle dimensioni dei pezzi. Per valutare $T_m(x, y)$ quando (x, y) sia non significativo basta più in generale verificare preliminarmente l'insieme di *k* coppie (x_i^*, y_i^*) significative dominate da (x, y) e che non si dominino l'una l'altra, per poi utilizzare la relazione

$$T_m(x, y) = \min_{i=1, \ldots, k}\{T_m(x_i^*, y_i^*) + (x - x_i^*) \cdot y + (y - y_i^*) \cdot x - (x - x_i^*) \cdot (y - y_i^*)\}$$

Tale generalizzazione non è necessaria ai fini dell'algoritmo, poiché tutti i *build* parziali hanno dimensioni (x, y) significative, salvo essere estese all'intera area dello stock senza aggiungere altri pezzi.

Si riporta di seguito anche la relazione ricorsiva su cui si basa il calcolo dei valori $T_m(x, y)$, denotata come *knapsack function* (Gilmore e Gomory, 1966):

$$T_m(x, y) = \min\{T_0(x, y), T_m(x_1, y) + T_m(x_2, y), T_m(x, y_1) + T_m(x, y_2)\}$$
$$\text{al variare di } x_1, x_2, y_1 \text{ ed } y_2 \text{ tali che } x_1 + x_2 \le x,\ y_1 + y_2 \le y,\ 0 < x_1 \le x_2,\ 0 < y_1 \le y_2,$$

dove per (x, y) significativi basta considerare il solo caso $x_1 + x_2 = x$ ed $y_1 + y_2 = y$.

Il valore $T_0(x, y)$ rappresenta lo scarto ottenibile inserendo il pezzo di area massima possibile in un rettangolo di dimensioni (x, y). L'idea su cui si fonda questo calcolo di

programmazione dinamica è quella per cui in un rettangolo R di dimensioni (x, y) vi possa essere un solo pezzo, oppure più pezzi divisi in due gruppi associati a due sottorettangoli contigui che compongono il rettangolo R. La contiguità avviene attraverso un taglio orizzontale alla coordinata y_1 o verticale alla coordinata x_1. Ciò ricalca da un lato l'idea dei build di Wang e dall'altro quella dei metodi top-down che, al contrario, partono dallo stock e provano tagli a tutte le coordinate significative.

Un secondo e più efficiente miglioramento è stato proposto da Viswanathan e Bagchi (1993). Come descritto, l'evoluzione proposta da Oliveira e Ferriera prevede l'ausilio di una stima basata sulla pre-risoluzione con programmazione dinamica, secondo la strategia di Gilmore e Gomory (1966), del problema non vincolato per ogni coppia significativa di dimensioni (x, y).

La novità proposta da Viswanathan e Bagchi è descritta nel seguito. Dato un rettangolo R ottenuto per agglomerazioni verticali e/o orizzontali e detto S lo stock a disposizione, l'upper bound calcolato secondo la *knapsack function* ha il difetto di considerare l'area libera in alto a destra in entrambi gli addendi della stima, come già descritto per il valore $T(L_R, W_R)$. Ciò è necessario poiché non è detto che R venga estratto direttamente con due tagli a partire dallo stock iniziale, ma sono possibili altri scenari. L'innovazione corrisponde nel tener conto di tutti i scenari possibili. Se per un rettangolo aggregato R di dimensioni x ed y, si indica con $g(R)$ la sua area utile e con $h(R)$ la stima dell'area utile aggiungibile con ulteriori *build*, un upper bound per $h(R)$ è

$$U_1(x, y) = L \cdot W - x \cdot y - T(x, y)$$

dove $T(x, y)$ è la stima proposta in Oliveira e Ferriera. Un secondo upper bound proposto, denotato con $U_2(x, y)$, stima l'area utile ulteriormente utilizzabile ancora basandosi su un problema non vincolato, ma estendendo l'attenzione a tutti i possibili scenari. Infatti, il valore $U_2(x, y)$ è calcolato ricorsivamente partendo dal valore $U_2(L, W) = 0$ dove L e W sono le dimensioni dello stock. Supponendo noti i valori di $U_2(x', y')$ al variare di (x', y') nell'insieme $[x, L] \times [y, W] - \{(x, y)\}$, si può calcolare

$$U_2(x, y) = \max\{h_1(x, y), h_2(x, y)\}$$
$$h_1(x, y) = \max\{U_2(x + u, y) + F(u, y) : 0 < u \leq L - x\}$$
$$h_2(x, y) = \max\{U_2(x, y + v) + F(x, v) : 0 < v \leq W - y\}$$

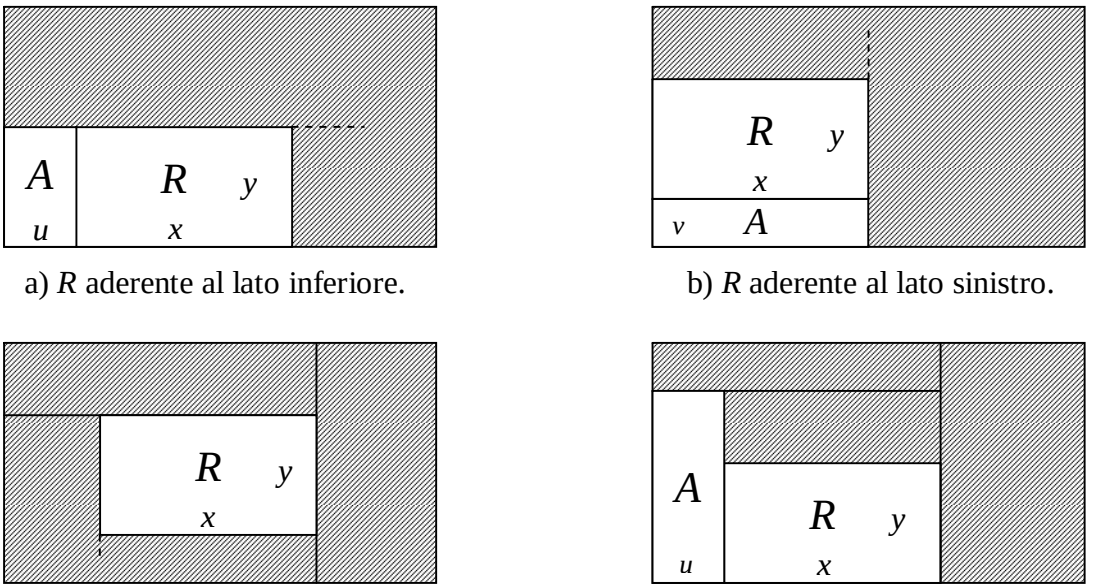
dove $F(x, y)$ è il complementare rispetto al $T_m(x, y)$ descritto nell'approccio di Oliveira e Ferriera, cioè l'area massima ottenibile e non lo scarto minimo ottenibile, per un problema non vincolato su un rettangolo di dimensioni (x, y).

La chiave di lettura di questa ricorsione, ponendo l'attenzione su $h_1(x, y)$, sta nel considerare scenari in cui R, aggregato insieme ad un rettangolo A alla sua sinistra di dimensioni (u, y), costituisca un rettangolo di dimensioni $(x+u, y)$ per il quale è nota l'area massima aggiungibile nel residuo fra "A+R" e lo stock a disposizione. Si tratta dunque di uno scenario dove c'è un taglio orizzontale che sottende almeno "A+R", come in Figura 1.11a.

Da notare che non tutti i valori di x, y, u e v sono importanti, ma solo quelli cui corrisponde un valore della tabella $T_m(a, b)$ e dunque di $F(a, b)$.

Si fa notare anche che il caso $u = 0$, opportunamente scartato poiché $U_2(x+0, y)$ è l'obiettivo della formulazione ricorsiva, equivale al caso $v = W - y$ in scenari dove il taglio orizzontale sottende solo R ed equivale ad un $u' > 0$ nel caso in cui quel taglio sottenda, oltre R, qualche altro rettangolo A' alla sua destra. Nel primo caso infatti vi sarà un taglio verticale che ha R immediatamente alla sua sinistra e nel secondo caso A' può essere equivalentemente spostato a sinistra di R. Analoghe considerazioni valgono per il caso $v = 0$.

Ne consegue che per il valore di $U_2(x, y)$ si restringe l'attenzione ai soli scenari in cui il rettangolo *R* risulti, nello schema finale, a contatto con il lato inferiore o con quello sinistro dello stock. Inoltre, per ciascuno dei due lati, si impone la presenza di un taglio che sottende, oltre *R*, l'area *A* "liberata dallo shift", come nei casi a) e b) di Figura 1.11. Nella stessa figura, i casi c) e d) rappresentano scenari non tenuti in considerazione, ma che sono equivalenti a scenari considerati. Ad esempio il caso d) è equivalente a quello in cui l'area *A* si trovi a destra del blocco di cui *R* fa parte, cioè uno scenario in cui *R* aderisce al lato sinistro dello stock. E' semplice generalizzare la considerazione e mostrare che con un numero finito di opportuni scambi di posizione si riesce sempre a portare *R* ad essere aderente al lato inferiore o a quello sinistro dello stock.



a) *R* aderente al lato inferiore. b) *R* aderente al lato sinistro.

c) *R* non aderente ai lati dello stock d) area *A* di ampiezza superiore ad *y*

Figura 1.11 Possibili scenari per schemi contenenti *R*

Inoltre Viswanathan e Bagchi generano l'insieme $L^{(M)}$ non come proposto da Wang, ma attraverso due liste di rettangoli, OPEN e CLIST. Inizialmente OPEN corrisponde ad $L^{(0)}$, poiché contiene tutti i rettangoli equivalenti ai tipi di pezzi da ottenere. Ad ogni iterazione viene scelto il rettangolo *R* di OPEN cui corrisponde il massimo valore della stima $f'(R) = g(R) + U_2(R)$ e viene combinato con tutti rettangoli di CLIST, avendo preventivamente aggiunto *R* alla stessa CLIST, così da combinare *R* anche con se stesso e producendo così agglomerazioni orizzontali e verticali come proposto da Wang. Tutti i nuovi *build*, ciascuno se compatibile con i vincoli di domanda associati a ciascun tipo di pezzi e, se non ridondante, viene aggiunto alla lista OPEN. Il modo di procedere, secondo cui si sceglie sempre il rettangolo di OPEN più promettente, determina la caratterizzazione dell'algoritmo come di tipo *best-first search*, come indicato dagli stessi autori nel titolo del lavoro. L'approccio di Wang proseguiva invece per livelli secondo il numero totale di pezzi costituenti un *build*.

1.2.7 Approccio top-down

Il lavoro di Christofides e Whitlock (1977) è una pietra miliare per la classe degli approcci single-stock di tipo top-down per il problema a ghigliottina vincolato.

La strategia di ricerca delle soluzioni è ad albero, dove ogni nodo è rappresentato da un insieme di rettangoli residui ottenuti dai tagli già imposti nel ramo che ha portato dalla radice dell'albero al nodo corrente. In ciascun nodo i rettangoli residui non sono associati ad alcun pezzo. A partire dal nodo corrente, ciascun nodo figlio è ottenuto scegliendo prima uno dei rettangoli residui, comune per tutti i nodi figli, e poi quale taglio aggiungere, diverso per ciascun figlio. Il taglio aggiunto può essere anche fittizio, indicato dagli autori come '0-cut', il cui significato è quello di considerare quel rettangolo come area residua definitiva nello schema di taglio, ed inserendolo nell'apposito insieme H_0 che contiene tali rettangoli, mentre con P si denota l'insieme di tutti i rettangoli residui. I rettangoli di H_0 possono, nella soluzione finale, contenere un unico pezzo oppure essere degli scarti.

Dato un rettangolo residuo di dimensioni intere (p, q) è possibile scegliere di operare su di esso un x-cut, cioè un taglio verticale, alle coordinate relative 1, …, p–1 o un y-cut, cioè un taglio orizzontale, alle coordinate relative 1, …, q–1 oppure uno 0-cut. In tutto vi sono p+q–1 scelte possibili. Vengono inoltre fatte le seguenti considerazioni

1. <u>simmetria:</u>

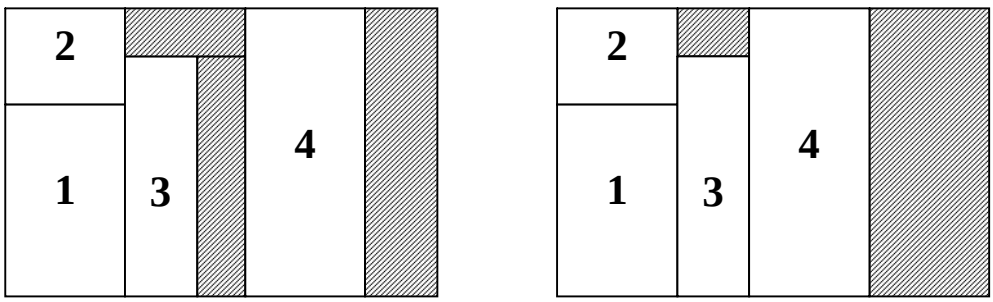
   se su un rettangolo (p, q), si opera un x-cut alla coordinata α, si ottengono due sottorettangoli di dimensioni rispettivamente (α, q) e (p–α, q), cosa che avviene anche se l'x-cut è operato alla coordinata p–α. Analoga considerazione vale per gli y-cut. Dunque è utile limitare le coordinate possibili agli intervalli [1 , $\lceil q/2 \rceil$] per gli x-cut ed [1 , $\lceil p/2 \rceil$] per gli y-cut, dove l'operatore $\lceil\ \rceil$ restituisce il minimo intero non minore del suo argomento.

2. <u>ordinamento dei cut</u>

   se un rettangolo di dimensioni (p, q) è diviso in due con un x-cut alla coordinata α e poi si opera un ulteriore x-cut alla coordinata β sul sottorettangolo di dimensioni (p–α, q), si ottengono tre rettangoli residui di lunghezze rispettivamente α, β e p–α–β, cosa ottenibile anche con una sequenza opposta di x-cut, vale a dire prima alla coordinata relativa β sul rettangolo iniziale e poi alla coordinata relativa α sul sottorettangolo di lunghezza p–β,. Per evitare questa ridondanza, basta imporre l'ulteriore limitazione per cui un rettangolo ottenuto con un x-cut 'ad α' non possa essere immediatamente ed ulteriormente tagliato con x-cuts a coordinate relative inferiori ad α. Ciò vale indipendentemente da se il rettangolo ottenuto con l'x-cut alla coordinata α sia il sottorettangolo destro o sinistro, poiché per il sinistro valgono analoghe considerazioni di ridondanza. Per gli y-cut vale analogo discorso.

3. <u>normalizzazione dei tagli</u>

   dato un rettangolo di dimensioni (p, q) è inutile operare tagli a coordinate relative che non corrispondono esattamente alla somma di dimensioni di qualche insieme di pezzi, poiché in tal caso si ottengono schemi di taglio cosiddetti non normalizzati, di cui si riporta un esempio in Figura 1.12.



a) schema non normalizzato b) schema normalizzato.

Figura 1.12 Normalizzazione degli schemi di taglio

Ai fini della normalizzazione, bisogna calcolare l'insieme N^q (risp. N^p), cui deve appartenere la coordinata relativa di un x-cut (risp. y-cut) quando la dimensione perpendicolare al taglio è q (risp. p), nell'ipotesi di problema orientato, ovvero l'insieme N^{pq} per quello non orientato. La valutazione di tali insiemi è utile anche nelle evoluzioni dell'approccio di Wang, per stabilire quali siano le coppie (x, y) per cui calcolare $T(x, y)$ ed $h(x, y)$ rispettivamente negli approcci di Oliveira e Ferreira (1990) e Viswanathan e Bagchi (1993).

Tali insiemi sono calcolabili all'inizio una volta per tutte, per cui quando si tratta un sottorettangolo è possibile che non si evitino schemi di taglio non normalizzati nel caso di problemi vincolati, cioè con limitato numero di pezzi. Ciò avviene poiché le coordinate fattibili sono quelle valutate secondo l'ipotesi di avere a disposizione tutti i pezzi iniziali, e non è detto che, in termini di normalizzazione, vi sia compatibilità fra le scelte ottime fattibili su sottorettangoli distinti di una stessa soluzione.

Una volta calcolati questi insiemi, si impone che gli x-cut e gli y-cut siano scelti a coordinate appartenenti al corrispondente insieme. Si descrivono di seguito le modalità di calcolo dell'insieme N^q. Si preordinano i tipi di pezzi per ampiezza w non decrescente.

Dato il valore della coordinata x di un possibile x-cut, e fissato un massimo indice r di tipi utilizzabili, esiste un insieme di possibili vettori ξ tali che $x = \sum_{i=1}^{r} \xi_i \cdot l_i$. Se tale insieme è vuoto, allora si definisce $f_r(x) = \infty$, altrimenti, se $i^* = \min_{\xi}\{\max\{i : \xi_i > 0\}\}$, si definisce $f_r(x) = w_i$.

Il valore $F_r(x)$ rappresenta il minimo valore di q affinché si possa realizzare un x-cut alla coordinata x così da ottenere schemi normalizzati ed ipotizzando di utilizzare a sinistra del taglio solo pezzi di indice non superiore ad r. Se n è il numero totale di tipi di pezzo, allora $x \in N^q \Leftrightarrow f_n(x) \le q$.

La definizione con il pedice generico r è stata introdotta poiché è di supporto per la modalità di calcolo della tabella $f_n(x)$. Infatti $f_i(x)$ è calcolabile in modo iterativo, basandosi sulla considerazione che all'aumentare di i, il valore $f_i(x)$ può solo diminuire e se i^* è il minimo valore di i per il quale $f_i(x) < \infty$, allora risulta $f_{i^*}(x) = w_{i^*}$ ed $f_i(x) = w_{i^*}$ anche per ogni $i > i^*$. Inoltre, se $f_{i-1}(x) = \infty$, l'unica possibilità affinché $f_i(x)$ sia finito e pari a w_i, è che sia possibile ottenere x aggiungendo $k \cdot l_i$ ad una coordinata z per la quale valga $f_{i-1}(z) < \infty$, dove k è un possibile numero di pezzi di tipo i da aggiungere. La formalizzazione di queste considerazioni si ottiene con l'espressione di programmazione dinamica

$$f_i(x) = \min\{f_{i-1}(x), \max\{w_i, \min_{1 \le k \le \min\{d_i, \lceil x/l_i \rceil\}}\{f_{i-1}(x - k \cdot l_i)\}\}\}$$

dove d_i è il massimo numero di pezzi di tipo i da ottenere, cioè la domanda associata.

La procedura enumerativa che genera tutti i possibili schemi di taglio, la maggior parte dei quali già normalizzati, prevede come già detto di scegliere ad ogni nodo dell'albero un rettangolo residuo che non appartenga ad H_0 e generare un figlio per ogni possibile taglio x-cut o y-cut soddisfacente le restrizioni emerse dalle tre considerazioni su indicate. Da un punto di vista di struttura dati gli autori hanno proposto di registrare quadruple (p, q, x, y) dove p e q rappresentano le dimensioni di un rettangolo e la coppia (x, y) identifica il cut realizzato secondo la seguente codifica.

Se $1 \le x \le \lceil p/2 \rceil$, allora si tratta di un x-cut alla coordinata relativa x. Se invece x è uguale a $\lceil p/2 \rceil + 1$, diventa significativa y, indicando con $1 \le y \le \lceil q/2 \rceil$ un y-cut alla coordinata relativa y e se vale anche $y = \lceil q/2 \rceil + 1$, allora si sta codificando lo 0-cut.

Per evitare di generare tutti gli schemi possibili, l'algoritmo prevede di valutare un upper bound associato al nodo corrente nel seguente modo.

Sull'insieme H_0 di u rettangoli definitivi, si risolve un problema di massimizzazione che prevede l'associazione a ciascun rettangolo di al più un pezzo fra tutti quelli possibili.
Il modello corrispondente è il seguente

$$\max z = \sum_{k=1}^{u} \sum_{i=1}^{n} a_{i,k} x_{i,k}$$

$$\sum_{i=1}^{n} x_{i,k} \leq 1$$

$$\sum_{k=1}^{u} x_{i,k} \leq B_i$$

$$x_{i,k} \geq 0$$

dove $x_{i,k}$ è una variabile 0-1 che vale 1 se si sceglie di associare un pezzo di tipo i al k-mo rettangolo di H_0 ed $a_{i,k}$ vale π_i (cioè la superficie-utilità dei pezzi di tipo i) se le dimensioni dell'i-mo tipo sono inferiori alle dimensioni del k-mo rettangolo di H_0, altrimenti $-\infty$, il che implica $x_{i,k} = 0$, data la massimizzazione di z.

Il modello corrisponde ad un problema del trasporto, risolvibile in modo efficiente come descritto in Desler e Hakimi (1969), per la sua particolare struttura.

Su ciascun rettangolo dell'insieme $P - H_0$ si calcola un upper bound basato sulla knsapsack function di Gilmore e Gomory (1966), di cui si è discusso in precedenza. Qualora le soluzioni ideali corrispondenti agli upper bound parziali su ciascun rettangolo di $P - H_0$, unite alla soluzione reale sull'insieme H_0, siano tali da soddisfare il vincolo sul numero massimo di pezzi ottenibili per ciascun tipo, allora l'upper bound totale corrisponde ad una soluzione fattibile ed è inutile esplorare il corrispondente nodo. L'altro caso in cui ci si può fermare è quando l'upper bound z^* sia non superiore al valore $\underline{z}$ della migliore soluzione fino ad allora ottenuta.

Si sottolinea anche che il problema del trasporto va risolto solo quando H_0 cambia.

Relativamente alle strategie di branching, gli autori hanno proposto tre possibili strategie per la scelta del rettangolo residuo di $P - H_0$ su cui operare un taglio.

Le più semplici sono quelle di scegliere il rettangolo di superficie minima, oppure quello di superficie massima.

Una seconda strategia prevede di scegliere il rettangolo cui è associato il più alto valore dello specifico upper bound, già calcolato per stimare la bontà del nodo corrente e dunque senza aumento della complessità dell'algoritmo.

La terza strategia prevede di scegliere in base al numero r_i di pezzi di ciascun tipo i corrispondente alla soluzione ideale complessiva ottenuta per la stima dell'upper bound, così come descritto in precedenza. Infatti deve esistere almeno un i per il quale tale valore r_i risulti maggiore di d_i, poiché in caso contrario ciò implicherebbe che la soluzione ideale sia una soluzione fattibile e si opererebbe backtracking, con eventuale aggiornamento della migliore soluzione fattibile trovata. L'idea consiste nello scegliere il valore i^* per il quale è massima la differenza $r_i - d_i$. Infine si sceglie come rettangolo di $P - H_0$, quello che ha il maggior numero di pezzi di tipo i^*, secondo le soluzioni ideali dei corrispondenti problemi non vincolati, calcolate su ciascuno di essi grazie alla strategia di programmazione dinamica utilizzata nella valutazione dei valori di knapsack function.

1.2.8 Approccio top-down modificato

Nei lavori di Hifi (1994) ed Hifi e Zissimopoulos (1997) sono stati proposti miglioramenti all'algoritmo di Christofides e Whitlock (1977), sviluppando dunque un nuovo algoritmo, ivi

denotato con il nome di MCW [Modified Christofides & Whitlock]. Le modifiche consistono in quattro punti:

a) nuovo lower bounds
b) nuovo upper bound
c) nuova strategia di branching
d) sviluppo di un metodo incrementale per la risoluzione del problema del trasporto sull'insieme H_0.

Si sottolinea che i primi tre punti possono essere sfruttati per determinare anche un'evoluzione della strategia bottom-up di Viswanathan e Bagchi (1993). Infatti essa utilizza upper bounds simili associati ai *build* anziché ai rettangoli residui. D'altro canto il lower bound è generale poiché è valutato attraverso una soluzione ottenuta con procedimento euristico costruttivo, salvo il fatto che l'utilizzo della relativa strategia durante tutti i singoli passi della ricerca esaustiva è possibile solo per l'approccio top-down come si comprenderà a breve leggendo il punto A. Hifi (1997b) propone appunto i primi tre punti per l'approccio bottom-up, denotando il conseguente algoritmo con il nome di MVB [Modified Viswanathan & Bagchi]. Di seguito si descrivono le evoluzioni proposte riferendosi all'approccio top-down di Christofides e Whitlock, salvo indicare esplicitamente come ciascun punto possa essere sfruttato anche nell'approccio bottom-up di Viswanathan e Bagchi (1993), così come indicato in Hifi (1997b).

A. *Nuovo lower bound.*
Il lower bound iniziale è calcolato utilizzando la procedura proposta da Fayard e Zissimopoulos (1995), per problemi orientati. La strategia è indicata con il nome di BSC (Best Strip Cutting). Essa prevede di ottenere una soluzione attraverso la combinazione di strip, cioè strisce, ottime.

Le strip orizzontali vengono calcolate nel modo seguente. Si preordinano i tipi di pezzi per ampiezza non descrescente, generando anche una lista di r distinte ampiezze $\underline{w}_1 \le \underline{w}_2 \le \ldots \le \underline{w}_r$. Per ciascuna ampiezza $\beta = \underline{w}_i$, si definisce l'insieme $S_{L,\beta}$ di indici j, denotanti i tipi di pezzi inseribili nella strip di dimensioni (L, β). Si risolve quindi il problema dello zaino, denotato con $K^i_{L\beta}$:

$$f_i(L) = \max \sum_{j \in S_{L,\beta}} v_j z_{ji}$$

$$\text{s.t.} \sum_{j \in S_{L,\beta}} l_j (z_{ji} / g_{ji}) \le L$$

$$z_{ji} \le \min\{\lfloor L / l_j \rfloor \cdot \lfloor \beta / w_j \rfloor , d_j\}$$

dove π_j è l'utilità associata al tipo j, l_j è la lunghezza di quel tipo e z_{ji} è il numero di pezzi di tale tipo per l'i-mo problema.

E' proposta una strategia di programmazione dinamica così da risolvere tutti gli r problemi risolvendo solo $K^r_{L\beta}$. Strategia analoga è adottata per la ricerca delle strip verticali.

L'insieme delle r strip ottime così valutate sono poi utilizzate per la risoluzione di un ulteriore problema dello zaino al fine di generare uno schema di taglio a ghigliottina al più a tre livelli:

$$(K_W) \begin{cases} \max \sum_{i=1}^{r} f_i(L) z'_i \\ s.t. \sum_{i=1}^{r} w_i z'_i \le W \end{cases}$$

dove z'_i è il numero di strip utilizzate di tipo i, generate cioè dal problema $K^i_{L\beta}$.

Nel problema K_W non si fa riferimento al vincolo sul numero di pezzi per ciascun tipo. Poiché il rilassamento sul numero di pezzi è inserito solo nei problemi K_L e K_W, la coppia di problemi $K^r_{L\beta}$ e K_W (insieme con gli analoghi $K^t_{\alpha W}$ e K_L) rappresenta, in termini strategici, una generalizzazione della procedura di Gilmore e Gomory (1966), che però non si limita a soluzioni su tre livelli. Del resto, lo scopo della procedura è generare un lower bound e non un upper bound, oltretutto non ottenibile data la limitazione imposta sui tipi di schemi di taglio che devono essere a tre livelli. Dunque se il problema K_W genera una soluzione fattibile secondo il numero massimo di pezzi, allora il valore associato a tale soluzione è un lower bound. Si fa notare che se il problema iniziale prevedesse un limite di soluzione a tre livelli, come talvolta accade nella pratica, allora la soluzione ottima scelta fra quelle di K_L e K_W rappresenterebbe un upper bound per il problema di partenza.

In caso contrario, se cioè K_W non genera una soluzione fattibile, si risolve appositamente il seguente *set packing problem*

$$(P_W)\begin{cases}\max \sum_{i=1}^{r} f_i(L) z'_i \\ s.t. \sum_{i=1}^{r} w_i z'_i \le W \\ \sum_{i=1}^{r} z_{ji} z'_i \le d_j\end{cases}$$

che genera certamente una soluzione fattibile. Lo stesso discorso vale per l'analogo problema P_L. Ciascuno dei due set packing problem, P_W e P_L, sono risolti solo se l'analogo K_x fallisce, cioè trova una soluzione non fattibile, poiché in caso contrario il problema P_x troverebbe la stessa soluzione. Il lower bound è infine scelto selezionando il massimo dei due valori ottenuti rispettivamente dalla coppia (K_W, P_W) e dalla coppia (K_L, P_L).

In Hifi (1997b) si utilizza lo stesso lower bound iniziale come componente dell'algoritmo MVB.

La strategia descritta è riutilizzabile in tutti i nodi dell'albero di ricerca, come sviluppato da Christofides e Whitlock (1977), sul generico rettangolo di dimensioni (p, q) dell'insieme $P - H_0$, così da valutare una promettenza del nodo. In particolare la risoluzione del problema $K^i_{p\beta}$ non è necessaria, poiché automaticamente ottenuta nella risoluzione di $K^r_{L\beta}$, data la strategia di programmazione dinamica utilizzata. Al contrario il problema K_x va risolto ogni volta, ma a tal fine è stato proposto, congruentemente con la valutazione di un lower bound, una strategia di tipo greedy che sceglie una strip alla volta da aggiungere, secondo il seguente schema algoritmico.

Per la strip t, il valore z'_t è superiormente limitato dalla costante $\min_{0 \le j \le n} \left\{ \frac{d_j}{z_{jt}} \right\}$, avendo aggiungendo in modo fittizio le definizioni $d_0 = W$ e $z_{0i} = \underline{w}_i$. Si definisce inoltre l'insieme I degli indici k di strip con limite superiore z'_t positivo.

Ad ogni iterazione dell'algoritmo, si aggiunge la strip con il più piccolo indice $k \in I$ fra quelli che massimizzano la funzione di utilità $\frac{f_v(L)}{\sum_{j=0}^{n} z_{jk}}$.

Un volta scelto k, ciò determina l'utilizzo del sottoinsieme di pezzi associato alla k-ma strip. Dunque si riducono conseguentemente i valori di d_j, sottraendo z_{jk} per ciascun

$j \in \{1, \ldots, n\}$, riducendo poi anche i valori massimi delle variabili z'_t ed eventualmente l'insieme I se per qualche ulteriore t risultasse un valore massimo aggiornato per z'_t pari a 0.

B. *Nuovo upper bound.*
Si ricorda che Christofides e Whitlock (1977), così come Oliveira e Ferriera (1990) per l'approccio bottom-up, utilizzano un upper bound basato sul rilassamento del vincolo (eliminato) sul numero di pezzi, risolvendo cioè problemi bidimensionali non vincolati con programmazione dinamica, come proposto da Gilmore e Gomory (1966). La modifica consiste nell'aggiungere la valutazione di un upper bound ottenuto dalla risoluzione di un problema che non trascuri il limite sul numero di pezzi, ma sia rilassato nel passaggio, in termini di somma di superfici, dalla bidimensionalità alla mono-dimensionalità. Dato cioè un rettangolo di dimensioni (α, β) il problema è così formulato:

$$(K'_{\alpha\beta}) \begin{cases} \max \sum_{j \in S_{\alpha\beta}} c_j z_j \\ s.t. \sum_{j \in S_{\alpha\beta}} (l_j h_j) z_j \le \alpha \cdot \beta & \text{[rilassamento dimensionale]} \\ z_j \le \min\left\{ \left\lfloor \frac{\alpha}{l_j} \right\rfloor \left\lfloor \frac{\beta}{h_j} \right\rfloor, d_j \right\} & \text{[limite bidimensionale]} \end{cases}$$

dove $S_{\alpha\beta}$ è l'insieme di pezzi che possono essere inseriti nel rettangolo, cioè con dimensioni l e w tali che $l \le \alpha$ ed $w \le \beta$.

Tale problema è risolto per ciascun rettangolo di $P - H_0$, così come nell'algoritmo di Christofides e Whitlock (1977) avveniva per il calcolo dell'upper bound U_G, dove G sta per Gilmore e Gomory, valutato come somma dei valori ottenuti su ciascun rettangolo di $P - H_0$. L'upper bound effettivamente utilizzato è allora $U_{GK} = \min\{U_G, U_K\}$, essendo U_K quello valutato come somma delle soluzioni ottime dei problemi $K'_{\alpha\beta}$ associati ai rettangoli di $P - H_0$. Da sottolineare che così come per U_G, anche per U_K vale il discorso secondo cui basta risolvere una sola volta all'inizio il problema K'_{LH} con programmazione dinamica. In Hifi (1997b), per la strategia MVB, si indica che l'upper bound U_K (denotato con V) può essere utilizzato accanto all'upper bound, denotato con U_2, proposto in Viswanathan e Bagchi (1993), risolvendo il problema mono-dimensionale sull'area con forma di una elle capovolta, lasciata libera, rispetto allo stock, dal generico rettangolo R della lista OPEN. Il valore $\min\{U_K, U_2\}$ di upper bound dell'utilità aggiuntiva è denotato con U'_2.

C. *Strategia di branching.*
Si ricorda che la terza e più ragionata strategia proposta da Christofides e Whitlock (1977) prevedeva di scegliere il rettangolo di $P - H_0$ che avesse il maggior numero di tipi per il quale fosse massima la differenza $r_i - d_i$, essendo i il tipo di pezzo il cui utilizzo r_i nella soluzione del problema rilassato fosse appunto il più lontano dal relativo limite di domanda residua d_i. La strategia proposta, invece, calcola il sottoinsieme di $P - H_0$ che massimizzi l'utilità totale, rispettando i vincoli d_i, ipotizzando di adottare su ciascuno dei rettangoli di tale sottoinsieme la migliore soluzione ottenuta per il calcolo dell'upper bound U_{GK} su problemi rilassati. Se t_j sono variabili 0-1 indicanti il fatto che il j-mo rettangolo R_j di $P - H_0$ viene selezionato o meno, il problema è così formalizzato:

$$\max \sum_j G(p_j, q_j) \cdot t_j$$

$$s.t. \quad \sum_j r''_{ij} t_j \le B_i - r'_i \quad , \quad i = 1, \ldots, n \quad , \quad t_j \in \{0,1\} \quad , \quad j = 1, \ldots, |L - H_0|$$

essendo $G(p_j, q_j)$ il valore ideale associato ad R_j, r'_i ed r''_{ij} il numero di pezzi di tipo i coinvolti rispettivamente nella soluzione del problema del trasporto associato all'insieme H_0 e nella soluzione associata all'upper bound calcolato per R_j.

Una volta risolto tale problema, il rettangolo scelto per il branching si seleziona fra quelli con $t_j = 0$, secondo la classica idea di forzare la costruzione della soluzione laddove quella ideale è maggiormente lontana dal rispettare i vincoli.

Più precisamente si sceglie, nell'insieme identificato dalla proprietà $t_j = 0$, quello che minimizza la quantità $\frac{G(p_j, q_j)}{\sum_{i=1}^{n} r''_{ij}}$, cioè il rapporto tra la qualità della soluzione del problema rilassato ed il numero di pezzi in essa contenuti.

D. *Incrementalità per il problema del trasporto.*
L'insieme H_0 viene, ad ogni iterazione, lasciato inalterato o arricchito con un rettangolo R o, nei backtracking, ridotto di uno o due rettangoli R_1, R_2 . La proposta consiste nel non risolvere il problema del trasporto ogni volta da capo, sviluppando una prima soluzione fattibile, arricchendo, in caso di aggiunta di un rettangolo R, o riducendo, in caso di backtracking, l'ultima soluzione adottata e valida per il precedente insieme H_0.

In caso di arricchimento, si associa ad R il migliore dei pezzi che possono essere inseriti in R e che sono ancora disponibili, cioè non associati agli altri rettangoli di H_0.

In caso di riduzione, quando si elimina un rettangolo R, si lascia semplicemente inalterata l'associazione dei pezzi con gli altri rettangoli rimasti. Questa strategia permette di avere subito a disposizione una soluzione sub-ottimale il cui valore può essere utilizzato come lower bound nella ricerca esaustiva vera e propria della soluzione ottima (per il problema del trasporto) e verifiche sperimentali mostrano che in questo ambito la soluzione sub-ottimale iniziale risulta spesso quella ottima, permettendo così di tagliare molti più rami nell'albero di ricerca associato al problema del trasporto.

1.2.9 Ulteriori miglioramenti

Il lavoro di Cung, Hifi e Le Cun (2000) sviluppa ulteriormente gli algoritmi MVB di Hifi (1997b) ed MWC di Hifi e Zissimopoulos (1997), che come detto condividono le stesse strategie evolutive a partire dagli approcci rispettivamente bottom-up di Viswanathan e Bagchi (1993) e top-down di Christofides e Whitlock (1977). In particolare il lavoro di Cung et al. fa esplicito riferimento all'algoritmo MVB. I miglioramenti proposti riguardano il lower bound iniziale, l'upper bound calcolato nodo per nodo, strategie per evitare soluzioni simmetriche ed infine tecniche implementative delle strutture dati per la velocizzazione dell'esecuzione.

I miglioramenti sono però utilizzabili anche nell'algoritmo MWC relativamente ai punti del lower bound e dell'upper bound.

A. *Miglioramento del lower bound iniziale.*
La strategia euristica per il calcolo del lower bound iniziale è stata proposta da Fayard, Hifi e Zissimopoulos (1998) ed è un miglioramento della strategia di Fayard e Zissimopoulos (1995). Questo lower bound iniziale migliorato può essere utilizzato tanto nell'approccio bottom-up quanto in quello top-down. L'algoritmo euristico di base è il Best Strips Cutting, o BSC, già precedentemente descritto e la sua evoluzione sviluppata da Fayard et al. è

denominata GBSC (General Best Strips Cutting), poiché tiene conto di vincoli di domanda sui pezzi.

L'euristica consiste nel tentare di dividere il rettangolo iniziale di dimensioni (*L*, *W*) in due sottoparti, attraverso un qualsiasi taglio orizzontale (risp. verticale) con coordinata appartenente all'insieme P_L (risp. P_W) che è l'insieme equivalentemente definito come N^W (risp. N^L) in Christofides e Whitlock (1977), come descritto in precedenza In pratica P_W è l'insieme delle possibili coordinate di un taglio verticale che si possa trovare in uno schema di tagli normalizzato. Analoga proprietà vale per P_L.

Una volta suddiviso il rettangolo (*L*, *W*) nelle due parti R_{1i} ed R_{2i}, dove *i* è l'indice del tentativo identificato dalla direzione del taglio e dalla sua coordinata, si applica la tecnica prima spiegata per la BSC a ciascuno dei due, avendo cura di applicarla prima al rettangolo R_{1i} che si trova a sinistra (risp. sotto) il taglio principale e poi la rettangolo R_{2i} secondo il valore residuo del numero di pezzi disponibili. L'unione delle due soluzioni così valutate su R_{1i} ed R_{2i} rappresenta una soluzione fattibile dell'intero problema. La migliore soluzione trovata al variare di *i*, secondo tutti gli equivalenti valori di P_L e P_W, rappresenta la soluzione finale dell'algoritmo ed il valore associato è il lower bound utilizzato appunto da Cung et al. . Per il problema su R_{1i} valgono le considerazioni già fatte per la tecnica BSC. Invece nella soluzione del problema su R_{2i} non si può sfruttare quanto ottenuto dalla soluzione con programmazione dinamica per l'intero rettangolo. Il motivo risiede nel fatto che il problema su R_{2i} deve tener conto della riduzione sul numero di pezzi disponibili secondo la soluzione fissata su R_{1i} rispetto alla quale bisogna mantenere la compatibilità.

Lo sviluppo di tale strategia è fondato su due osservazioni pratiche riscontrate sperimentalmente.

1. La soluzione ottima ha spesso una semplice struttura a strip secondo quanto studiato nel lavoro di Hifi e Zissimopoulos (1997).
2. In generale la soluzione ottima consiste in uno schema in cui il rettangolo iniziale è diviso in pochi sottorettangoli, ciascuno dei quali ha una struttura a strip.

B. *Affinamento dell'upper bound.*

Accanto all'upper bound U''_2 introdotto in Hifi (1997b) e descritto nel punto B del precedente paragrafo, sono utilizzati altri due semplici upper bounds, selezionando poi il minimo fra i tre utilizzati. Si ricorda che nel procedimento bottom-up, si ha a che fare con due liste, OPEN e CLIST, di rettangoli generati dall'agglomerazione di sottorettangoli più semplici attraverso build verticali oppure orizzontali ed a partire da rettangoli equivalenti ai pezzi di partenza. Dato un tale rettangolo *R* di dimensioni (α, β) e dette *L* e *W* le dimensioni dello stock, si calcola il valore $U_3(R) = V(L, W) - g(R)$ dove $g(R)$ è l'utilità nota associata al rettangolo *R* e $V(L, W)$ è l'upper bound, associato all'intero stock e calcolato come ottimo del rilassamento mono-dimensionale precedentemente introdotto. Il valore $U_3(R)$ può in qualche caso essere inferiore a $V(\alpha, \beta)$. In pratica $U_3(R)$ serve ad imporre l'utilizzo di un upper bound $U_{tot}(R)$ sulla promettenza $h(R)$ tale che $g(R) + U_{tot}(R) \leq V(L, W)$.

Inoltre, si definisce $U_4(R) = \sum_{i \in \underline{S}} \pi_i (d_i - b_i)$ dove b_i è il numero di pezzi di tipo *i* utilizzati in *R*, mentre $\underline{S}$ è l'insieme di indici *i* dei tipi di pezzi per cui valgono le relazioni $l_i \leq L - l_R$ e $w_i \leq W - w_R$, oltre a $d_i - b_i \geq 0$. $U_4(R)$ è calcolato sulla falsariga di quanto già proposto in Viswanathan e Bagchi (1993) per gli approcci bottom-up, relativamente al porre la promettenza aggiuntiva $h(R) = 0$ nei casi particolari in cui l'analogo di $\underline{S}$ sia vuoto, sebbene al relativo problema non vincolato sia associata una soluzione a valore positivo.

L'upper bound utilizzato è infine $U_{tot}(R) = \min\{ U'_2(R), U_3(R), U_4(R) \}$.

Il valore di $U_3(R)$ non è utilizzabile nell'approccio top-down, mentre lo è $U_4(R)$, considerando come b_i il numero di pezzi del tipo i utilizzati nella soluzione sviluppata sull'insieme H_0.

C. *Implementazione della struttura dati per* CLIST.
L'implementazione proposta per la CLIST è a matrice sparsa, come mostrato in Figura 1.13, con gli indici di riga e di colonna corrispondenti rispettivamente alle lunghezze ed alle ampiezze ordinate in modo non decrescente, come rappresentato in figura, e limitatamente ai valori utili per ottenere schemi di taglio normalizzati.

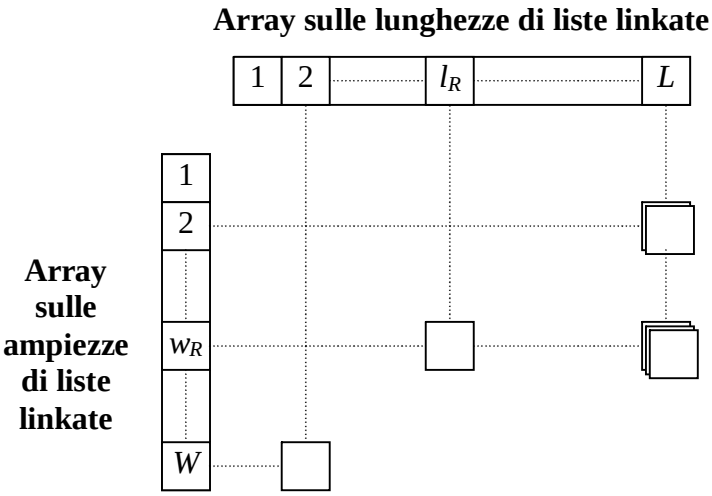


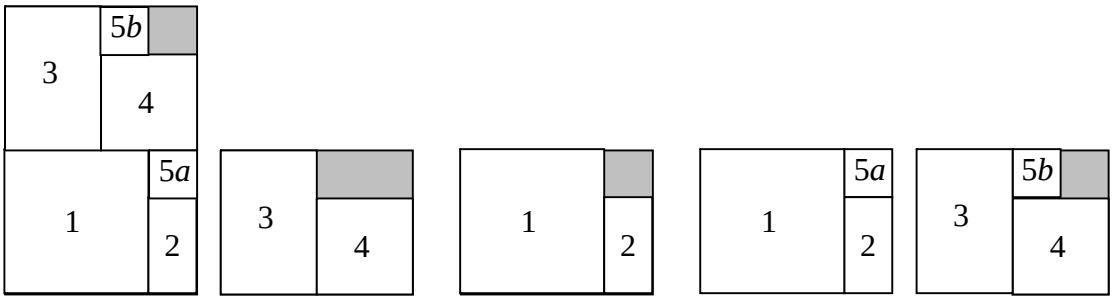
Figura 1.13 Implementazione della CLIST

L'ordinamento per lunghezze e per ampiezze permette di calcolare velocemente i rettangoli candidati per possibili agglomerazioni con il rettangolo R scelto dalla lista OPEN. I rettangoli candidati, rispettivamente per build verticali e per build orizzontali, sono quelli tali che il build generi un rettangolo agglomerato di dimensioni non superiori a quelle dello stock.

D. *Strategie anti-simmetria.*
Un'altra evoluzione consiste nell'approccio alla questione dei rettangoli duplicati. Nel lavoro di Tschöke e Holthöfer (1995) si è proposto, una volta generato un rettangolo T di dimensioni (l_T, w_T) e con numero di pezzi per ciascun tipo i pari a T_{Di}, di verificare se esso fosse dominato da qualche rettangolo T' già presente o dominasse qualche rettangolo T''.
In particolare T' domina T se $l_{T'} \le l_T$, $w_{T'} \le w_T$ e $T'_{Di} \ge T_{Di}$ per ogni i.

L'evoluzione consiste nel verificare tipi di dominanza che comportano uno sforzo computazionale minore, ed indipendente dall'ampiezza corrente di CLIST. Per semplicità espositiva non si descrivono le strategie, tranne una che è errata se applicata, come nell'intento del lavoro che si sta descrivendo, al caso di domanda limitata. Fra le tre strategie proposte, D1, D2 e D3, quella errata è la D1. Rifacendosi alla Figura 1.8 relativa all'approccio bottom-up di Wang (1983), la strategia D1 consiste nell'eliminare un build se lo scarto aggiuntivo è in grado di contenere almeno un pezzo non contenuto in nessuno dei due rettangoli dalla cui fusione è generato il build. Si supponga allora di avere un problema vincolato e non pesato con i seguenti dati: uno stock con dimensioni $L = 4$ e $W = 6$ e cinque tipi di pezzi, tutti con domanda 1, rispettivamente con dimensioni $l_1 = 3$ e $w_1 = 3$, $l_2 = 1$ e $w_2 = 2$, $l_3 = 2$ e $w_3 = 3$, $l_4 = 2$ e $w_4 = 2$ ed infine $l_5 = 1$ e $w_5 = 1$. In questo caso le due soluzioni ottime sono quelle contemporaneamente raffigurate in Figura 1.14a, e distinguibili per il fatto

che il pezzo denotato con il numero 5 è utilizzabile esclusivamente nella posizione *a* o nella posizione *b*. Il difetto della strategia D1 consiste nel non permette la generazione dei builds mostrati nelle Figure 1.14b ed 1.14c, poiché in entrambi i casi, il pezzo di tipo 5 può essere contenuto nello scarto associato al build. La strategia D1 permette invece la generazione dei builds mostrati nelle Figure 1.14d ed 1.14e ma ciascuno di tali due build ha bisogno, per determinare una soluzione ottima, di essere fuso con uno dei due build non generati.



a) soluzioni ottime b) build rigettato c) build rigettato d) build accettato e) build accettato

Figura 1.14 Errore della strategia D1

Si sottolinea infine che gli approcci bottom-up di Wang (1983) e quelli migliorativi di Viswanathan e Bagchi (1993), Hifi (1997b) e di Cung, Hifi e Le Cun (2000) sono facilmente adattabili ad una serie di problemi con limiti sul numero totale di pezzi o di tipi di pezzi in uno stock, o con orientamento libero.

1.3 Approcci multi-stock in letteratura

Si sottolinea innanzitutto che in letteratura il problema multi stock, soprattutto nel caso non a ghigliottina, è meglio noto come Bin Packing. Il primo paragrafo è dedicato a diverse tecniche euristiche, sia per problemi a ghigliottina che non. Segue la descrizione di una tecnica di risoluzione esatta per il problema non a ghigliottina. Infine una tecnica meta-euristica, basata sulla Tabu Search, ancora per il problema non a ghigliottina.

1.3.1 Tecniche euristiche costruttive

La maggior parte degli algoritmi euristici descritti di seguito sono di tipo greedy, ma possono essere ben classificati in due famiglie, una di algoritmi ad una fase ed una di algoritmi a due fasi.

Gli algoritmi ad una fase posizionano direttamente i pezzi negli stock, che sono di dimensioni finite, generalmente utilizzabili per problemi non a ghigliottina.

Gli algoritmi a due fasi posizionano inizialmente i pezzi in uno stock ideale, tecnicamente chiamato *strip*, che abbia una delle dimensioni uguale ad una delle due dimensioni degli stock, supposti identici (in genere quella massima e nel seguito, per convenzione, la lunghezza L), mentre l'altra dimensione è infinita. Nella seconda fase la strip è divisa lungo la sua dimensione infinita, senza spezzare i pezzi posizionati al suo interno, in modo da ottenere *shelfs* equivalenti agli stock del problema. La prima fase equivale alla risoluzione di un problema detto di Strip Packing. Sebbene l'obiettivo nello Strip Packing è minimizzare la parte utilizzata lungo la dimensione infinita, non è detto che le euristiche migliori per tale fine permettano poi di generare nella seconda fase soluzioni migliori per il problema multi stock. In Figura 1.15 si riporta un esempio.

Nell'esempio di figura si suppone che i trenta pezzi identici, che possono essere ruotati, abbiano lunghezza pari ad 1 ed ampiezza pari a 5, mentre gli stock sono di lunghezza 11 ed ampiezza 9. Nel caso a) le tre "righe", due di undici pezzi ed una di otto pezzi, sfruttano al meglio la dimensione fissa della strip, mentre nel caso b), in cui vi sono quindici "righe" da due pezzi ciascuna, questo non succede. Tuttavia, l'orientamento del caso b) è da preferirsi quando avviene la separazione della strip in blocchi equivalenti agli stock. Infatti, nel caso a), si utilizzano tre stock, mentre nel caso b) soltanto due.

Inoltre la maggior parte degli algoritmi euristici costruttivi, sia per il Bin Packing che per lo Strip Packing, seguono una logica a mensola, tecnicamente *shelf*, inserendo cioè i pezzi per righe, per convenzione da sinistra a destra, ciascuna delle quali costituisce un livello o, appunto, una shelf. La prima shelf ha il suo lato inferiore corrispondente al lato inferiore della strip o dello stock e ciascuna nuova shelf è appoggiata sul lato superiore del pezzo posto più in alto fra quelli contenuti nella shelf precedente. Nel caso del Bin Packing le shelves sono poi mantenute sovrapposte l'una all'altra se l'algoritmo è ad una fase, salvo interrompere la sequenza descritta e passare a nuovi stock qualora vi sia incompatibilità in ampiezza del pezzo successivo da aggiungere, dato il limite imposto dall'ampiezza dello stock.

Se invece l'algoritmo è a due fasi, le shelves sono ricombinate come in un problema di packing mono-dimensionale secondo le ampiezze delle shelves e l'ampiezza W degli stock. La logica a shelf produce soluzioni con tagli a ghigliottina, con livello massimo pari a tre, a meno che non si aggiungano altre fasi all'euristica che riposizionino i pezzi negli stock.

Recenti studi sul legame fra Strip Packing e Bin Packing sono riportati in Han, Iwama, Ye e Zhang (2007).

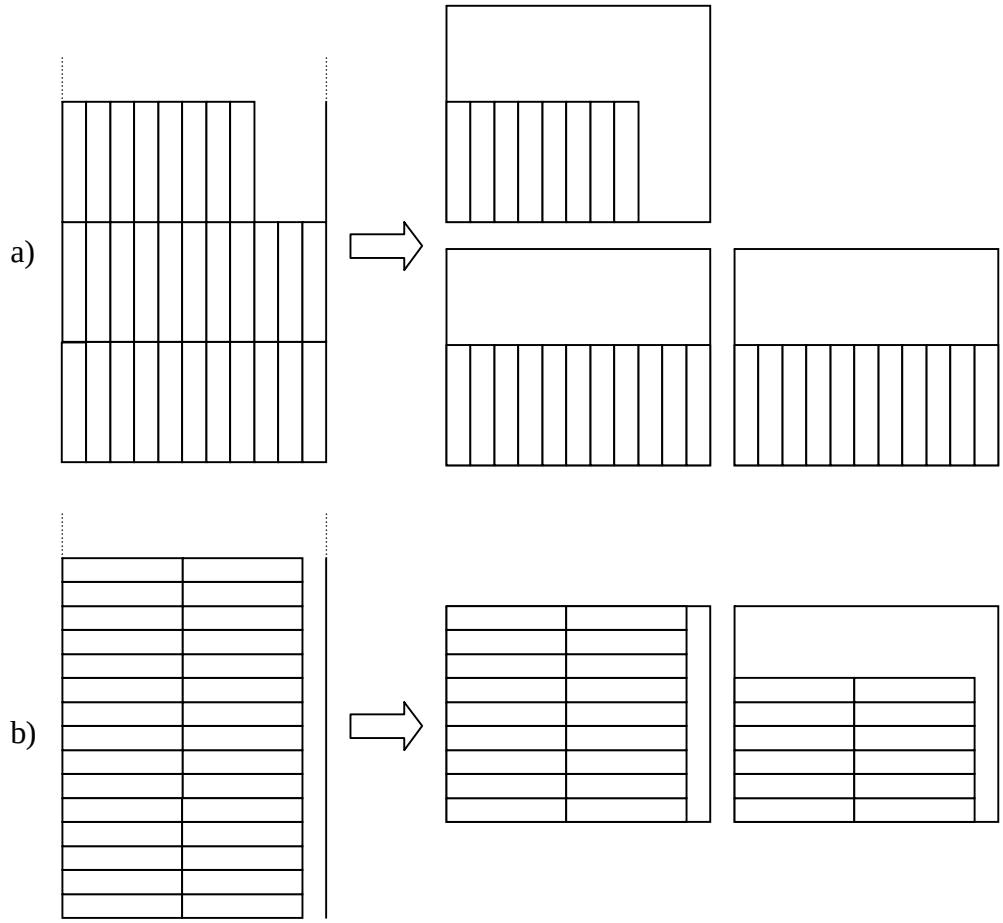


Figura 1.15 Strip Packing e multi-stock

Tre classiche strategie per lo Strip Packing prevedono il preordinamento dei pezzi per ampiezza non crescente o, a parità di ampiezza, per lunghezza non crescente. Nella prima, denominata Next-Fit Decreasing Height (NFDH), il pezzo corrente p_j è giustapposto a sinistra nella shelf corrente se ciò è possibile, altrimenti è posto a sinistra in una nuova shelf, giustapposta sopra la vecchia shelf, che diventa quella corrente e con ampiezza limitata dall'ampiezza del pezzo p_j che è il primo inserito nella nuova shelf. La limitazione dell'ampiezza è automaticamente dovuta al preordinamento dei pezzi nella sequenza con cui sono considerati.

Nella seconda, denominata First-Fit Decreasing Height (FFDH), si verifica l'inseribilità del pezzo corrente p_j in tutte le shelves già presenti, inserendo il pezzo nella prima shelf in cui è inseribile, se ve ne sono, altrimenti in una nuova shelf.

Infine, nella terza strategia, denominata Best-Fit Decreasing Height (BFDH), a differenza dell'FFDH, il pezzo corrente p_j è posto nella shelf, fra quelle possibili, in cui è minimizzato lo spazio orizzontale della shelf che resta inutilizzato, purché vi siano shelves 'possibili', altrimenti è inserito in una nuova shelf.

In Figura 1.16 si riporta un esempio cui sono applicate le tre strategie indicate.

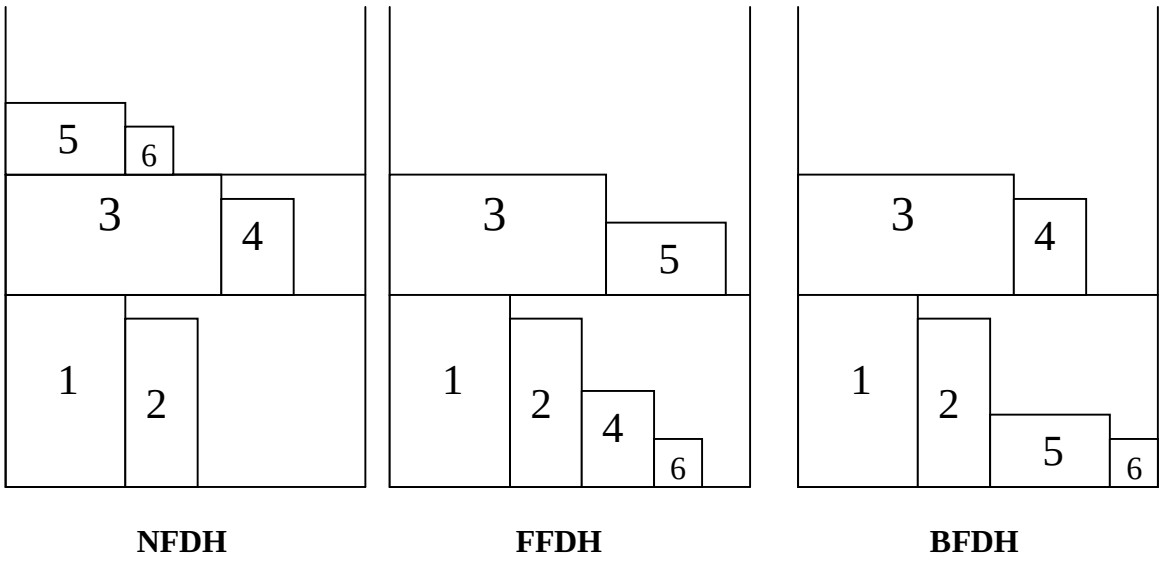


Figura 1.16 Confronto fra le tre strategie della prima fase

Senza il preordinamento dei pezzi per ampiezza decrescente, ciascuna di queste tre strategie potrebbe risultare arbitrariamente scadente. Più precisamente, dato un algoritmo euristico **A**, ed un'istanza **I** del problema, il valore della funzione obiettivo prodotto da **A** su **I** è denotato con A(I), mentre il valore della soluzione ottima è indicato con OPT(I). Supponendo che OPT(I) sia il valore minimo, cioè che il problema sia a minimizzare, il rapporto $\rho(I) = A(I)/OPT(I)$ indica la bontà dell'algoritmo **A** sull'istanza **I**. La qualità dell'algoritmo **A** è esprimibile attraverso il comportamento medio, cioè $E[\rho(I)]$, dove la media statistica indicata può dipendere dalle ipotesi statistiche sui valori numerici che identificano le istanze del problema. Un altro indice di qualità è il comportamento peggiore, vale a dire il massimo valore di $\rho(I)$ al variare delle istanze **I**. Tale valore è indicato come *absolute worst-case performance*. Per una descrizione approfondita di questi concetti, si veda Garey e Johnson (1979).

Un ulteriore e più indicativo indice di qualità sul caso peggiore è l'*asyntotic worst-case performance*, che scarta i valori peggiori relativi ad istanze del problema la cui dimensione è limitata da un qualsivoglia numero intero *N*. Formalmente l'asyntotic worst-case performance è definito come il più piccolo valore ρ fra quelli per i quali esista un numero intero finito *N* tale che valga la relazione $\rho(I) \leq \rho$ per ogni istanza **I** di dimensione superiore ad *N*. Tale valore minimo è indicato con ρ^{∞}.

Coffman, Garey, Johnson e Tarjan (1980) hanno provato che, normalizzando le ampiezze dei pezzi in modo che $\max_i\{w_i\} = 1$, risultano, rispettivamente,

$$NFDH(I) \leq 2OPT(I) + 1$$

ed

$$FFDH(I) \leq (17/10)OPT(I) + 1$$

dove l'eventuale non normalizzazione delle ampiezze ha effetti solo sul termine additivo. Questi risultati valgono, come detto, per lo Strip Packing.

Gli algoritmi a due fasi seguono le stesse logiche di quelli ad una fase, salvo considerare analoghe strategie nel riposizionamento delle shelves. Nell'algoritmo Hybrid First-Fit (HFF), la prima fase è equivalente all'FFDH, mentre nella seconda fase, in cui le

shelves risultano, nell'ordine di generazione, già ordinate per ampiezza decrescente, si adotta l'analoga strategia First-Fit Decreasing come in un problema mono-dimensionale che tenga conto delle ampiezze delle shelves e dell'ampiezza degli stock.

Chung, Garey e Johnson (1982), che hanno proposto la strategia HFF, hanno mostrato che, normalizzando le ampiezze dei pezzi al valore massimo uno, risulta

$$HFF(I) \leq (17/8)OPT(I) + 5$$

ma non è stato ancora trovato un esempio che segua esattamente il valore 17/8, come invece vale per l'NFDH e l'FFDH.

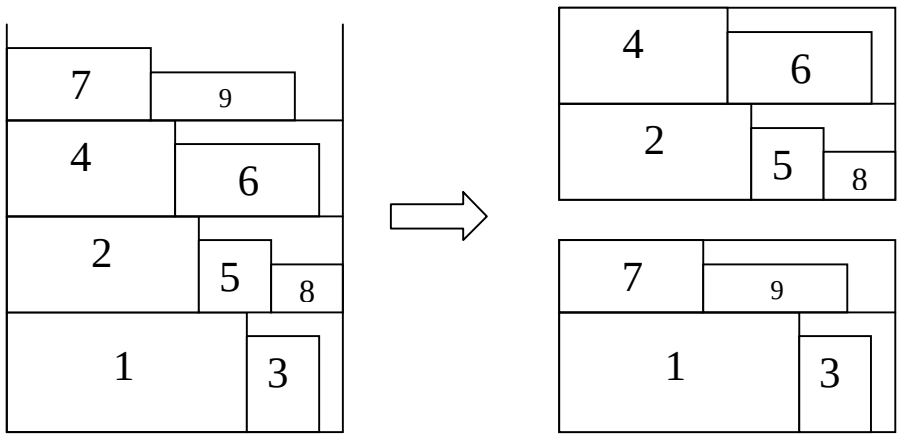
Per il peggior esempio trovato si ottiene il valore (91/45)(OPT(I) – 1) per cui si dice che il bound 17/8 non è stato ancora dimostrato essere "fine".

Berkey e Wang (1987) hanno invece per primi proposto quello che qui sarà indicato, per omogeneità con i nomi delle altre strategie, l'algoritmo Hybrid Best-First, che utilizza il BFDH nella prima fase e la strategia Best-First Decreasing nella seconda fase. Gli autori hanno comunque indicato l'algoritmo con il nome di Finite Best-Strip (FBS).

Infine si riporta la strategia Hybrid Next-Fit (HNF) che utilizza l'NFDH nella prima fase e l'analoga strategia di Next-Fit Decreasing nella seconda fase. La natura della strategia, dovuta al "Next-Fit", rende l'intero algoritmo equivalente all'algoritmo ad una fase in cui il pezzo corrente p_j è posto nella shelf corrente, se possibile, altrimenti in una nuova shelf, se c'è abbastanza spazio verticale, oppure nella prima shelf di un nuovo stock. Frenk e Galambos (1987) hanno provato che il comportamento asintotico nel caso peggiore è caratterizzato dalla relazione

$$HNF(I) \leq 3.382OPT(I) + 9$$

ed il bound è fine, mentre per l'NFDH nello Strip Packing valeva un fattore 2.
In Figura 1.17 si riporta un esempio per la strategia HFF.



a) prima fase b) seconda fase

Figura 1.17 Le due fasi dell'algoritmo HFF

Per ciascuna delle strategie a due fasi esiste la corrispondente strategia ad una fase, che condivide la modalità della prima fase ed usa, nella seconda, la tecnica Next-Fit, mantenendo cioè la sequenza delle shelves così come costruite nella prima fase, cioè come presenti nella strip. Le tre strategie ad una fase sono indicate con il nome di Finite Next-Fit (FNF), Finite First-Fit (FFF) e Finite Best-Fit (FBF) che dunque utilizzano nella prima fase rispettivamente l'NFDH, l'FFDH e il BFDH. L'FNF è equivalente alla strategia HNF, come già detto.

Le tre strategie a due fasi indicate sono le più semplici ed infatti generano schemi di taglio a ghigliottina ed hanno complessità computazionale al più $O(n \cdot \log n)$ se si utilizzano le

opportune strutture dati. Inoltre sono state descritte per il caso 2BP|O|* (secondo la tassonomia di Lodi, 1999), non avendo preso in considerazione la possibile rotazione dei pezzi. Per tener conto dell'orientamento non fisso, basta innanzitutto preordinare i pezzi ancora secondo ampiezze non crescenti o, a parità di ampiezze, per lunghezza non crescente, ma solo dopo aver definito come lunghezza la dimensione maggiore e come ampiezza quella minore, così come per lo stock, in conformità alla convenzione utilizzata anche per problemi orientati. Quando un pezzo è posto con la sua dimensione maggiore parallelamente alla dimensione maggiore dello stock, che per convenzione è posta come lunghezza, allora si dice che l'orientamento del pezzo è orizzontale, altrimenti verticale. Inoltre, quando per un pezzo si verifica l'inseribilità in una shelf, si considerano entrambi i possibili orientamenti, purché non sia il primo della shelf stessa nel qual caso si considera solo l'orientamento orizzontale. Per gli altri pezzi, si preferisce l'orientamento verticale, verificando che non si superi l'ampiezza del primo pezzo della shelf. L'orientamento verticale è infatti quello che determina un migliore utilizzo dello spazio della shelf.

Inoltre è possibile inserire una fase intermedia in cui, considerando l'area rettangolare minima contenente le shelves, si verifica l'inseribilità dell'intera area-shelf, ruotata di 90°, nello spazio residuo delle altre shelves, con strategie di tipo Next-Fit, First-Fit o Best-Fit. Tale fase genera comunque schemi di taglio a ghigliottina, ma con livelli di taglio fino a quattro. Se però in questa fase intermedia si considerano le shelves in ordine inverso, si aumenta la possibilità di operare più cicli di reinsiemento, generando schemi di taglio con livello non limitato a priori, e pari al più ad 1 + 2P se P è il numero di shelves ottenute nella prima fase.

Vi sono altre due strategie 'a shelf', più complesse delle tre riportate. La prima di esse è la Floor-Ceiling (FC) descritta in Lodi, Martello e Vigo (2002a), sviluppata come estensione delle tre strategie 'a shelf' di base che posizionano i pezzi con il loro lato inferiore appoggiato sul lato inferiore della shelf, sfruttando dunque solo il 'pavimento' (floor) della shelf e non anche il 'soffitto' (ceiling). La strategia FC invece verifica l'inseribilità di un pezzo anche secondo una disposizione con il lato superiore del pezzo aderente al lato superiore della shelf, con priorità di posizionamento da destra a sinistra, al contrario di quanto avviene sul floor, dove è da sinistra verso destra. La politica utilizzata è la Best-Fit, con priorità ai floors delle shelves già aperte, ma si possono sviluppare altrettante strategie secondo le politiche First-Fit o Next-Fit. La tecnica Floor-Ceiling può produrre schemi non a ghigliottina a meno che non si abbia l'accortenza di shiftare i pezzi sul ceiling verso sinistra, quando opportuno, in modo da far corrispondere l'ascissa dei loro lati destri con quella del lato destro di uno dei pezzi sul floor, preferendo quello più a destra possibile. Se il problema è ad orientamento libero, allora si preferisce, quando vi sia la possibilità di scelta, l'orientamento verticale sul ceiling e, sul floor, quello verticale se il ceiling non è vuoto e quello orizzontale viceversa. Tali scelte hanno l'obiettivo di massimizzare la probabilità di inserire ulteriori pezzi nella stessa shelf. Si sottolinea che la strategia prevede di inizializzare un ceiling solo con pezzi che non sono inseribili nel floor.

In Figura 1.18 si riporta un semplice esempio di applicazione dell'algoritmo FC.

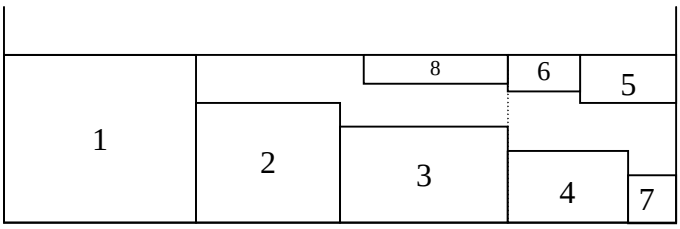


Figura 1.18 Algoritmo FC

L'altra strategia 'a shelf', pure descritta in Lodi, Martello e Vigo (2002a), è indicata con il nome di Knapsack Packing (KP). Essa prevede, una volta inserito il primo pezzo p_0 in una shelf, di riempire la shelf con un algoritmo 1BP, cioè di bin packing mono-dimensionale, in cui a ciascun pezzo sia associato un costo pari alla sua lunghezza l_i, un'utilità pari alla sua area $w_i \cdot l_i$ ed una capacità dello zaino pari ad $L - l_0$. Il problema 1BP è noto essere NP-hard, per cui può essere conveniente risolverlo attraverso apposite tecniche euristiche.

Come spunto originale si indica che la strategia KP potrebbe essere arricchita con un passo di riempimento del 'ceiling', con strategie Next-Fit, First-Fit o Best-Fit, adottando le opportune verifiche se lo schema di taglio deve essere a ghigliottina.

Se il problema non è orientato, si preferisce imporre, per i calcoli di costo ed utilità, l'orientamento verticale per i pezzi inseribili in tale modo.

Si riportano infine strategie non a shelf, che producono in genere schemi non a ghigliottina, per cui sono adatti ai casi 2BP|*|F. La prima di esse è la strategia Bottom-Left (BL), peraltro già descritta agli inizi del Capitolo, sviluppata per il caso 2BP|O|F, che prevede di preordinare i pezzi per lunghezza non crescente e di inserire ciascun pezzo nella posizione più in basso possibile e poi più a sinistra possibile. Il preordinamento dei pezzi produce una performance assoluta, per lo Strip Packing, indicata da $BL(I) \leq 3 \cdot OPT(I)$.

La complessità computazionale dell'algoritmo è di $O(n^2)$, se implementato come indicato da Chazelle (1983). Berkey e Wang (1987) hanno proposto il BL per il bin-packing (Finite Bottom-Left o FBS), aggiungendo l'apertura di un nuovo stock quando il pezzo corrente non sia inseribile nello stock corrente (né in quelli precedenti se la strategia non è di tipo Next-Fit). Anche qui sono possibili diverse strategie di verifica dell'inseribilità negli stock non correnti, determinando una complicazione se la strategia non è di tipo Next-Fit.

Un'altra strategia è l'Alternate Direction (AD), anch'essa per il caso 2BP|O|F, ancora descritta in Lodi, Martello e Vigo (2002a). Con l'utilizzo di un lower bound, si considera un dato numero iniziale di stock, riempiti come se si inserisse una singola shelf in ciascuno di tali stock, dopodichè, per ciascuno stock si inseriscono 'bande' di pezzi in alternanza da destra verso sinistra e da sinistra verso destra, dove il primo pezzo di ciascuna banda è inserito con strategia Bottom-Right o Bottom-Left a seconda della direzione corrente, mentre gli altri pezzi della banda corrente sono inseriti, se la direzione corrente è 'da destra a sinistra' (risp. 'da sinistra a destra') con il lato destro (risp. sinistro) aderente e parzialmente sovrapposto al lato sinistro (risp. destro) dell'ultimo pezzo precedentemente inserito nella banda, nella posizione più in basso possibile. La banda corrente è aumentata con il primo pezzo (First-Fit) inseribile fra quelli rimasti disponibili e gli stock sono considerati uno per volta. E' probabile che gli stock iniziali non siano sufficienti a contenere tutti i pezzi, nel qual caso si apre un nuovo stock per volta in cui si ripete l'intera strategia, per cui il riempimento di una prima shelf equivale al riempimento della prima banda, da sinistra a destra, poiché lo stock correntemente aperto è uno solo. In Figura 1.19 si riporta un esempio di applicazione dell'AD, con lower bound iniziale pari a 2, mentre nel Box 1.1 è presente una codifica informale dell'algoritmo, così come data dagli autori che hanno proposto la strategia. Sembra che non siano stati realizzati studi sperimentali o analitici per verificare quali fossero le modifiche più adatte al caso di problema non orientato.

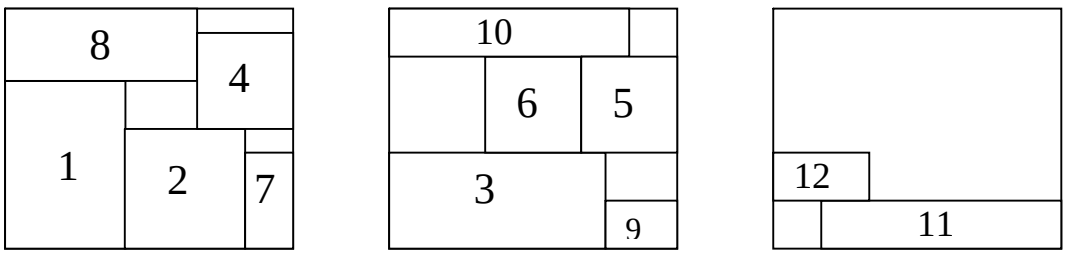


Figura 1.19 Algoritmo AD

Algorithm AD_{OF}:
ordina i pezzi secondo ampiezze w_j non crescenti.
commento : fase 1
for j := 1 **to** n **do**
if il pezzo p_j può essere posto sul fondo di qualche stock **then**
posiziona p_j, giustapposto a sinistra, sul fondo dello stock per il quale si minimizza lo spazio residuo a destra.
commento : fase 2
i := 0
repeat
i : = i + 1, right_to_left := true, n_fail = 0;
repeat
sia p_j il primo pezzo ancora non posizionato che può essere posto nell' i-mo stock, secondo il corrente valore di right_to_left.
if j = nil **then**
n_fail := n_fail + 1, right_to_left := not right_to_left;
else
posiziona p_j nell'i-mo stock secondo right_to_left, n_fail := 0
until n_fail = 2
until tutti i pezzi risultano posizionati.
end

Box 1.1 Formalizzazione dell'algoritmo AD (per il caso 2BP|O|F)

Per il caso 2BP|R|F, Lodi (1999) ha proposto la strategia Touching Perimeter (TP). In tale strategia si preordinano i pezzi per area non crescente e, a parità di area, per eccentricità non crescente. Utilizzando un certo lower bound si predetermina un numero minimo di stock, come nel caso dell'AD. Il pezzo corrente p_j è posizionato nello stock per il quale è minima, nel migliore dei casi possibili, la porzione di perimetro del pezzo non aderente né ai lati dello stock né a lati di altri pezzi già posizionati, con l'obiettivo di minimizzare le aree 'intrappolate' e per questo non riutilizzabili. In uno stock vuoto la posizione di un pezzo è, per convenzione, quella nell'angolo in basso a sinistra, con orientamento orizzontale, visto che i quattro angoli e gli otto corrispondenti posizionamenti sono equivalenti dal punto di vista della porzione perimetrale di aderenza.

La strategia TP è particolarmente adatta nel caso 2BP|R|F, ma è utilizzabile anche nel caso 2BP|O|F. In Figura 1.20 si riporta un esempio applicativo.

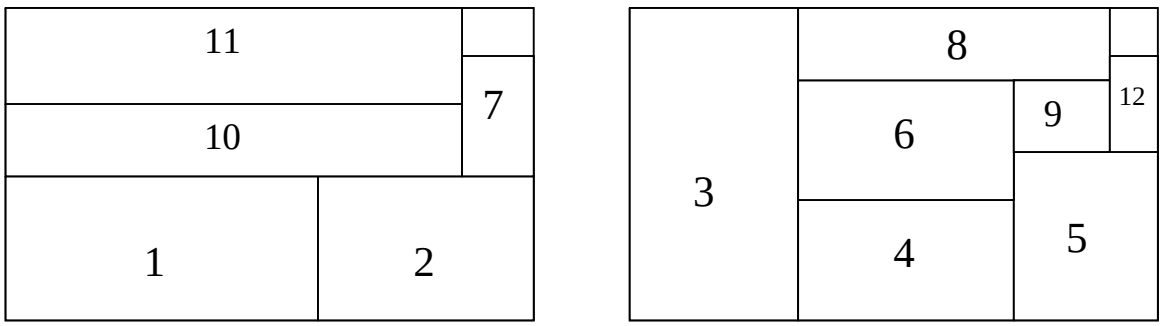


Figura 1.20 Algoritmo TP

1.3.2 Un algoritmo esatto

Un algoritmo esatto per il caso multi-stock è stato proposto da Martello e Vigo (1998) e prevede uno schema branch & bound su due livelli, uno di associazione pezzi-stock e l'altro di packing in ciascuo stock. Il primo livello è chiamato *outer branch-decision tree*, in cui a ciascun nodo il pezzo corrente è assegnato ad uno degli stock aperti o, se il numero di questi ultimi è inferiore a $z^* - 1$, ad un nuovo stock, dove z^* è il numero di stock nella migliore soluzione correntemente trovata.

Al secondo livello, chiamato *inner branch-decisione tree*, nello stock a cui è assegnato il pezzo corrente, si verifica la possibilità di inserire il nuovo insieme di pezzi ad esso associato, provando in prima battuta, per velocizzare la procedura, a mantenere inalterato il posizionamento dei pezzi già precedentemente assegnati e verificando le sole posizioni in cui il nuovo pezzo ha il lato inferiore aderente a quello inferiore dello stock o a quello superiore di uno dei pezzi già presenti ed il lato sinistro aderente a quello sinistro dello stock o al lato destro di uno dei pezzi già presenti, cioè con il vertice in basso a sinistra del nuovo pezzi negli angoli concavi dell'inviluppo della frontiera generata dai pezzi già presenti. Tali punti vengono anche chiamati *corner points*. La Figura 1.21 riporta un esempio.

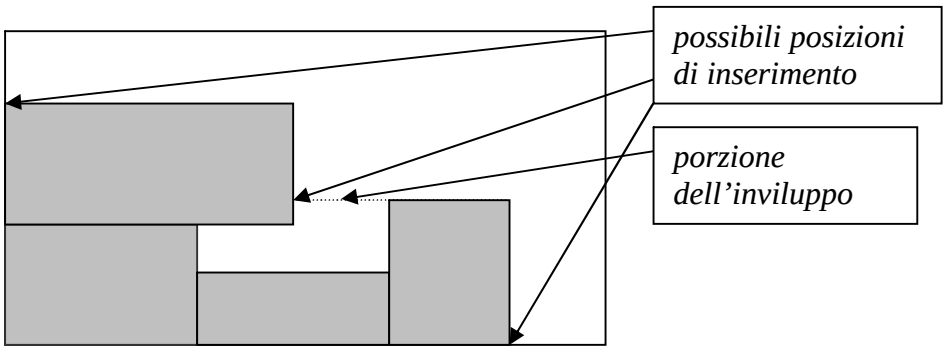


Figura 1.21 Inviluppo e Corner Points

Questo tentativo iniziale è utile quando i pezzi già assegnati allo stock corrente occupino una porzione relativamente piccola della sua area. Se il tentativo fallisce, allora è necessaria una procedura esatta che generi un nuovo packing del nuovo insieme di pezzi. A tal fine viene calcolato un lower bound (per il numero di stock necessari) evitando la ricerca del packing se esso produce un valore di 2. Poi viene prodotto un packing attraverso una euristica per tentare di costruire un packing single-stock e se ciò non avviene si passa ad una procedura esaustiva con tecnica branch & bound (*inner tree*). Trascurando le considerazioni utili a capire quando tagliare il sottoalbero di ricerca correntemente visitato, la logica è quella di provare tutti gli ordinamenti possibili dell'insieme dei pezzi e, per ciascun posizionamento dei primi $n - 1$ pezzi, provare il posizionamento dell'n-mo pezzo in tutti i corner points relativi all'inviluppo convesso della frontiera degli $n - 1$ pezzi già posizionati, eventualmente in entrambe le rotazioni possibili se il problema è ad orientato libero. Se in più il problema non è 'free', ma a ghigliottina, ad ogni aggiunta di un pezzo, bisogna decidere se e quale taglio a ghigliottina aggiungere.

Se neanche questa procedura esatta produce un packing con un solo stock, allora l'intera procedura opera un backtracking, generando poi un nuovo nodo dell'outer tree, se possibile, altrimenti termina.

In ogni nodo dell'inner tree possono essere utili le tecniche studiate negli approcci single-stock, per verificare in anticipo che il posizionamento del corrente sottoinsieme di pezzi già considerati non può produrre una soluzione che utilizzi un solo stock.

1.3.3 Un algoritmo meta-euristico

Altri approcci al problema multi-stock prevedono l'utilizzo di tecniche approssimate, in genere meta-euristiche, fra cui il gli Algoritmi Genetici o ricerche locali, fra cui la Simulated Annealing e il Tabu Search. Per una descrizione completa di questo tipo di tecniche e per l'acquisizione delle definizioni di base si rimanda ad Aarts e Lenstra (1997), Goldberg (1989) e Glover e Laguna (1997).

Nel caso degli algoritmi genetici, la codifica di una soluzione può corrispondere ad una permutazione dei pezzi, il cui posizionamento è determinato ad esempio dall'algoritmo Bottom-Left, secondo l'ordine corrispondente alla permutazione, o comunque da un prefissato algoritmo di posizionamento applicato a valle del preordinamento dettato dalla permutazione.

Nel caso delle tecniche di ricerca locale, una possibile definizione di intorno di una soluzione prevede, dato un valore intero k ed un pezzo p_j associato ad uno stock, di considerare una ulteriore k-pla di stock e di ridisporre i pezzi precedentemente assegnati alla k-pla di stock aggiungendo a tale insieme il pezzo p_j , ridisponendo l'insieme aumentato in al più k+1 stock.

A titolo di esempio si riporta la strategia Tabu Search utilizzata in Lodi (1999). Lo schema algoritmico seguito è indipendente da quale delle quattro classi (2BP|*|*) sia quella cui appartiene il problema. Questo aspetto influenza solo la costruzione della soluzione iniziale e l'euristica di riposizionamento dei pezzi durante la scansione degli intorni. Per la selezione dello stock da cui estrarre un pezzo, si considera una 'funzione di riempimento' associata all'insieme P_i dei pezzi associati all'i-mo stock e definita come

$$\phi(P_i) = \alpha \frac{\sum_{j \in P_i} w_j \cdot l_j}{W \cdot L} - \frac{|P_i|}{n}$$

che rappresenta la difficoltà di riposizionare i pezzi dell'insieme P_i. Infatti tale valore aumenta all'aumentare della porzione di area utilizzata ed al diminuire del numero di pezzi dell'insieme P_i. Il parametro α è posto pari a circa 20 in Lodi (1999), sulla base di prove sperimentali.

Lo stock selezionato è quello con valore minimo della funzione di riempimento. Per ciascun pezzo p_j associato a tale stock, si prova a reinserirlo insieme ai pezzi di altri k stock, con il quale costituisce l'insieme S, utilizzando una data procedura euristica **A**. Se essa produce un packing con meno di k stock oppure un packing con k stock, ma lo stock selezionato aveva un solo pezzo, allora si è generata una soluzione con un numero di stock inferiore e tale 'mossa' è comunque accettata, decrementando poi il valore del parametro k, purché non già unitario. Se invece si utilizzano k stock e lo stock selezionato aveva altri pezzi oltre al pezzo p_j che si è provato a riposizionare, allora la mossa è accettata solo se il pezzo p_j non è nella tabu list associata al valore k. Nel caso si dovessero utilizzare k+1 stock, e solo se $k > 1$, si seleziona ulteriormente il t-mo stock, fra l'insieme dei k+1 usati, i cui indici formano l'insieme I, con il minor valore di $\phi(P_t)$ e si considera l'insieme di pezzi $T = P_t \cup (P_i \setminus \{p_j\})$ ricalcolando una soluzione per l'insieme di pezzi T con l'euristica **A** verificando il corrispondente numero di stock utilizzati. Si noti che l'insieme di pezzi T, prima dell'applicazione di **A**, è distribuito su due soli stock, cioè quello selezionato inizialmente e da cui si è estratto il pezzo p_j, e quello selezionato a valle del primo riposizionamento. Solo se l'applicazione di **A** determina l'utilizzo di un solo stock e se il pezzo p_j non è nella tabu list, si considera come penalty il valore $\min\{\phi(T), \min_{h \in I \setminus \{t\}} \phi(P_h)\}$. Se il valore della penalty non è in una seconda tabu list, allora la mossa è accettata. In tutti gli altri casi la mossa non è accettata e se il primo riposizionamento utilizza k+1 stock oppure, con $k = 1$, il secondo riposizionamento utilizza due stock, allora il valore di k viene aumentato e, se già pari ad un

predefinito valore massimo, viene imposta un'azione di diversificazione della ricerca. La diversificazione consiste nel selezionare come stock da cui estrarre i pezzi p_j, non quello con minimo valore della funzione di riempimento, bensì quello con d-mo valore, se ordinati in modo non decrescente. Il valore di d è aumentato di 1 per ogni azione di diversificazione imposta. Se esso raggiunge il valore z pari al numero di stock correntemente utilizzati, o comunque un valore d_{max} prestabilito, si impone una diversificazione più drastica, riposizionando, con l'euristica **A**, i pezzi degli $\lceil z / 2 \rceil$ stock con valori massimi della funzione di riempimento, resettando a 0 il valore di d e svuotando le tabu list. La meta-euristica si ferma dopo un prefissato tempo o se si trova una soluzione con valore pari al lower bound del numero di stock necessari.
Le euristiche utilizzate in Lodi (1999) variano con la classe e sono:

- knapsack packing (KP) per il caso 2BP|O|G
- per il caso 2BP|R|G si considera knapsack packing con una fase aggiuntiva uguale a quella descritta come fase intermedia aggiungibile per le strategie Strip Packing.
- l'Alternate Direction per il caso 2BP|O|F
- la tecnica Touching Perimeter per il caso 2BP|R|F.

1.4 Analisi sullo Strip Packing problem

Relativamente agli algoritmi ad una fase descritti in precedenza, si ritiene utile riportare interessanti risultati raggiunti dalla ricerca negli ultimi anni. Il lavoro di Seiden e Woeginger (2005) riassume tali risultati apportando anche importanti contributi. Esso in particolare fa riferimento allo Strip Packing con tagli a ghigliottina.

Definendo con $G_k(I)$ la minima ampiezza ottenibile per l'istanza **I** del problema bidimensionale a ghigliottina e non orientato e limitando le soluzioni affinché non contengano più di *k* stages, sono riportati i seguenti risultati sulla *asymptotic performance ratio*.

1.4.1 Caso di strip packing a due livelli

Se $k = 2$, dato un qualunque numero $r \geq 1$ ed una qualunque costante positiva c, esiste certamente un'istanza **I** tale che

$$G_2(I) > r \cdot OPT(I) + c$$

e ciò significa che, scegliendo soluzioni con due soli livelli di taglio, non si ha a priori nessuna assicurazione sulla qualità della migliore soluzione ottenibile.

E' abbastanza semplice mostrare la validità di tale risultato. Fissato un numero positivo λ e supposto che la strip abbia lunghezza unitaria, basta considerare un istanza **I** avente λ^2 pezzi di lunghezza $1/\lambda$ ed ampiezza nell'intervallo $[1 - 1/\lambda, 1]$. Le ampiezze vanno costruite in modo da essere tutte diverse tra loro. Si noti che i ragionamenti nell'intervallo continuo [0, 1] sono estendibili a casi discreti, considerando una lunghezza opportunamente elevata così da poter scegliere dimensioni intere, e comunque tutte distinte, dei pezzi.

Per l'istanza I_λ così costruita, una qualunque soluzione a due livelli può contenere un solo pezzo per ogni shelf, per cui tutte le soluzioni a due livelli che non contengano tagli inutili sono equivalenti ed utilizzano una porzione dell'ampiezza della strip almeno pari a $\lambda^2 - \lambda$, dato che l'ampiezza di ciascun pezzo è non inferiore ad $1 - 1/\lambda$. D'altro canto una soluzione fattibile a ghigliottina con tre livelli di taglio permette di organizzare shelves con λ pezzi disposti l'uno accanto all'altro con ciascuna shelf di ampiezza al più pari ad 1. Di conseguenza l'ampiezza utilizzata da una tale soluzione è al più λ e se ne ricava che

$$G_2(I)/OPT(I) \geq \lambda - 1$$

ed il fatto di non aver ipotizzato alcun limite per λ implica l'asserto.

1.4.2 Caso di strip packing a tre livelli

Per il caso $k = 3$, si dimostra che, denotato con h_∞ il valore della sommatoria $\sum_{i=1}^{\infty} \frac{1}{t_i - 1}$, dove gli elementi dell'insieme $\{t_i\}$ sono quelli della sequenza di Salzer (1947), allora per ogni fissato $\varepsilon > 0$, esiste una costante c_ε, tale che una qualunque istanza **I** soddisfa la relazione

$$G_3(I) \leq (h_\infty + \varepsilon) \cdot OPT(I) + c_\varepsilon$$

Ciò vuol dire che, pur restringendo l'attenzione a soluzioni con soli tre livelli di taglio, è possibile ottenere, per istanze con valori ottimi sufficientemente elevati, soluzioni peggiori di al più il 69.11% rispetto all'ottimo, in termini di porzione utilizzata dell'ampiezza della strip, dato che $h_\infty \approx 1.69103$. Inoltre il numero h_∞ è il minimo numero possibile per il quale valga la suddetta proprietà.

Facendo un passo indietro, la sequenza di Salzer è così definita:

$$\begin{cases} t_1 = 2 \\ t_{i+1} = t_i \cdot (t_i - 1) + 1 \qquad i \geq 2 \end{cases}$$

In particolare, indicando con m_i il valore $t_i - 1$, dalla seconda uguaglianza si ricava la relazione $m_{i+1} = m_i \cdot (m_i + 1)$.
La sequenza dei numeri di Salzer soddisfa la proprietà

$$\sum_{i=1}^{\infty} \frac{1}{t_i} = 1 \tag{1.1}$$

Infatti è semplicemente dimostrabile per induzione che

$$\sum_{i=1}^{d} \frac{1}{t_i} = 1 - \frac{1}{t_d (t_d - 1)} \tag{1.2}$$

da cui discende che, facendo tendere d all'infinito, si ottiene una sommatoria di valore 1.

Il lavoro di Csirik e Woeginger (1997) aveva già mostrato la validità del valore h_∞ nell'ambito dello studio degli algoritmi on-line, che cioè associano in sequenza ciascun pezzo per volta ad una shelf in maniera definitiva e senza conoscere i pezzi successivi della sequenza. Ovviamente la validità di h_∞ per l'algoritmo on-line appena referenziato rappresenta un risultato addirittura più forte di quello obiettivo del teorema, che riguarda algoritmi in generale.

Per completezza si indica che il suddetto lavoro fa riferimento alla classe di algoritmi a shelves introdotta da Baker e Schwarz (1983) e denotati con SHELF(A, α), dove α è un parametro appartenente all'intervallo]0, 1[ed **A** è un algoritmo on-line per il caso single-stock (o bin packing) mono-dimensionale. Più dettagliatamente, fissati **A** ed α, l'algoritmo SHELF(A, α) separa i pezzi in classi C_s, con $s \in \boldsymbol{N}$, associando un pezzo di ampiezza w_i alla classe identificata dall'unico numero naturale positivo s che soddisfa la relazione $\alpha^{s+1} < w_i \leq \alpha^s$. L'algoritmo **A** consiste nello scegliere in quale delle shelves associate alla classe C_s e correntemente aperte si debba inserire il pezzo, o eventualmente se aprire una nuova shelf.

L'algoritmo **A** per il quale si mostra che SHELF(A, α) ha una *asymptotic performance ratio* pari ad h_∞ è l'algoritmo armonico introdotto da Lee e Lee (1985) per il bin-packing mono-dimensionale.

Il lavoro di Csirik e Woeginger (1997) poneva in risalto l'analogia fra le performance asintotiche sia dell'algoritmo armonico che del precedente miglior algoritmo, il First Fit Shelf, per il caso mono-dimensionale e per il caso bidimensionale ottenuto con α sufficientemente prossimo ad 1, lasciando aperta la questione sulla validità di tali analogie per altri algoritmi studiati per il caso mono-dimensionale, come quello di Richey (1991) per il quale valeva la ratio 1.5888 e basato anch'esso sul packing armonico, con gli opportuni miglioramenti.

Tuttavia un primo risultato negativo che limita ad h_∞ il miglior caso possibile per la classe degli algoritmi SHELF(A, α) è fornito proprio in Csirik e Woeginger (1997), attraverso istanze con gruppi di pezzi di pari lunghezza ed appositamente costruite con l'ausilio della squenza di Salzer. Un ulteriore risultato negativo è fornito invece in Seiden e Woeginger (2005), qui descritto e che generalizza questo limite a qualsiasi algoritmo che produca schemi a ghigliottina 3-staged. Si tratta della parte negativa del teorema enunciato per il caso $k = 3$.

Di seguito è riportata la dimostrazione della validità di $h_\infty + \varepsilon$, per ogni fissato $\varepsilon > 0$, per SHELF(A, α) sotto un'opportuna scelta di α ed utilizzando come **A** l'algoritmo armonico,

che suddivide (per ciascuna classe C_s) i pezzi in sottoclassi identificate dal loro indice, da 1 a *p*, con *p* numero naturale positivo, ed avendo tralasciato il pedice *s* per semplicità espositiva. La suddivisione in classi avviene in funzione dell'intervallo in cui cade la lunghezza l_i di ciascun pezzo, secondo la seguente regola.

$$classe(l_i) = \begin{cases} j & if \quad \frac{1}{j+1} < l_i \le \frac{1}{j}, j < p \\ p & if \quad 0 \le l_i \le \frac{1}{p} \end{cases}$$

Ad ogni classe viene associata una sola shelf aperta per volta che viene riempita con i pezzi associati alla classe, uno dopo altro secondo l'ordine e finché è possibile inserirvene uno. Quando ciò non è più possibile, si apre una nuova shelf e la precedente viene definitivamente chiusa, comportando dunque una ridotta complessità di calcolo. Per le shelves delle sottoclassi $j < p$, c'è spazio per esattamente *j* pezzi, né più né meno.

Lee e Lee (1985) introducono una funzione-utilità da associare a ciascun pezzo, secondo la lunghezza, utile ai fini della prova del teorema.
Tale funzione-utilità è così definita:

$$g_p(x) = \begin{cases} \frac{1}{j} & if \quad classe(x) = j < p \\ \frac{px}{p-1} & if \quad classe(x) = p \end{cases}$$

Essa associa al generico valore di lunghezza *x*, compreso in [0, 1], il massimo valore di lunghezza che determinerebbe l'associazione alla stessa classe, quando essa sia inferiore a *p*, ed *x* stessa, moltiplicata per un opportuno fattore, nel caso di classe *p*. Il diagramma di $g_p(x)$ è dunque quello riportato in Figura 1.22.

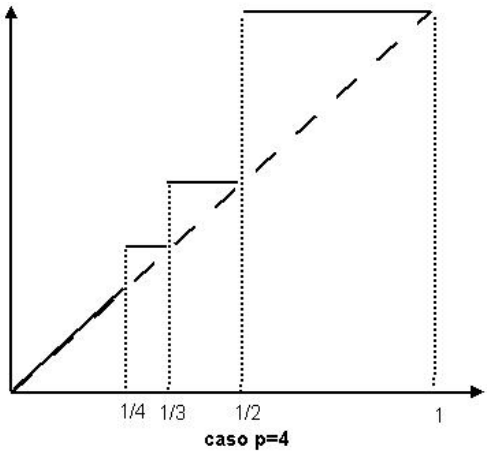


Figura 1.22 Andamento di $g_p(x)$

Inoltre, detto *u* l'intero che soddisfa la relazione $t_u \le p < t_{u+1}$, si definisce

$$G_p = \sum_{i=1}^{u} \frac{1}{t_i - 1} + \frac{p}{(t_{u+1} - 1)(p-1)}$$

Lee e Lee (1985) dimostrano le due seguenti proprietà.

Lemma 1.1. $\forall p \ge 2 \qquad G_p \ge h_\infty \quad$ e $\quad \lim_{p \to \infty} G_p = h_\infty$.

Prova. Si indica preliminarmente che il valore di $p = 1$ è escluso per non annullare il denominatore del secondo addendo nella definizione di G_p. Il lemma è dimostrato implicitamente, provando sia l'impossibilità per un algoritmo on-line $O(1)$-space di avere una ratio asintotica migliore di h_∞, sia la possibilità di costruire un algoritmo on-line $O(1)$-space di ratio asintotica non maggiore di G_p ed uguale a G_p nel caso $p = t_u$. Qui non si riportano tutte queste prove, dando però una dimostrazione equivalente del lemma.
Si può innanzitutto verificare per induzione che

$$\sum_{i=1}^{u} \frac{1}{t_i} = 1 - \frac{1}{t_{u+1} - 1} \tag{1.3}$$

Tale uguaglianza equivale di fatto alla (1.2).

E' inoltre evidente che al crescere di p, per u costante, il valore di G_p decresce. Infatti, quando p passa dal valore $t_{u'} - 1$ al valore $t_{u'}$, e cioè u passa da u' ad u' + 1, la differenza fra il vecchio ed il nuovo valore di G_p vale

$$\left[\frac{1}{t_{u'} - 1} + \frac{t_{u'}}{(t_{u'+1} - 1)(t_{u'} - 1)}\right] - \frac{t_{u'} - 1}{(t_{u'} - 1)(t_{u'} - 2)} = \frac{1}{t_{u'} - 1}\left[\frac{t_{u'}}{t_{u'+1} - 1} - \frac{1}{t_{u'} - 2}\right] =$$

$$\frac{1}{t_{u'} - 1}\left[\frac{1}{t_{u'} - 1} - \frac{1}{t_{u'} - 2}\right]$$

e dunque è strettamente negativa. Ciò conferma che G_p decresce strettamente al crescere di p. Poiché inoltre, per la (1.1), G_p tende ad h_∞, allora risulta $G_p \geq h_\infty$ per $p \geq 2$.

Lemma 1.2. $\forall p \geq 2$, data una qualunque sequenza di numeri reali $x_1, x_2, \ldots, x_q$, appartenenti all'intervallo [0, 1] e tali che $\sum_{i=1}^{q} x_i \leq 1$, vale la disuguaglianza $\sum_{i=1}^{q} g(x_i) \leq G_p$ avendo omesso il pedice p nell'indicazione di g(·) per semplicità.

Tale lemma sarà il cuore della dimostrazione sia per il caso mono-dimensionale che bidimensionale.
Prova. Facendo un passo indietro, il valore $g(x_i)$ può essere interpretato come costo, in termini di spazio, che comporta la presenza di un pezzo di classe j. Infatti nelle shelves relative alla classe j entrano esattamente j pezzi, per cui il costo associato è $1 / j$. Nelle shelves relative alla classe p, al contrario, non c'è alcun limite a priori, e le shelves sono certamente riempibili fino a superare la porzione $(p - 1) / p$, per cui si può dire che un pezzo di lunghezza l ha una utilità al più di $l \cdot p/(p - 1)$, nel senso che un insieme di pezzi di classe p e di lunghezze tali che la loro somma sia pari a $(p - 1)/p$ determina l'occupazione minima raggiungibile per la shelf, per cui a tale insieme viene associato un costo totale pari ad 1.

Per dimostrare il lemma, si suppone senza perdere di generalità che $x_1 \geq x_2 \geq \ldots \geq x_q$. Si sottolinea inoltre che il rapporto fra utilità e costo $x_i / g(x_i)$, quando $x_i \in]\, 1/(j+1), 1/j]$, ha un massimo pari ad 1 per $x_i = 1/j$ ed un estremo inferiore, pari a $(j+1) / j$, quando x_i tende ad $1/(j+1)$, poiché $g(x_i)$ è costante e pari ad $1/j$. Dunque il rapporto inverso fra costo ed utilità $g(x_i)/x_i$ assume un valore minimo pari ad 1 ed ha un estremo superiore, che decresce con j. Per $x_i > 1/p$, il rapporto $g(x_i)/x_i$ è invece costante e pari a $p/(p - 1)$ e dunque equivalente all'estremo superiore del caso $j = p - 1$.

Inoltre è sufficiente mostrare il lemma per il caso $\sum_{i=1}^{q} x_i = 1$, dato che la $g(x)$ è strettamente crescente. Nel seguito si denota con I(j) l'intervallo di valori di lunghezze che determinano l'associazione con la classe j. Se $x_1 \notin \mathrm{I}(m_1)$, allora tutti gli x_i sono non maggiori di $1/2 = 1/(m_1+1) = 1/t_1$. Ne consegue che per tutti gli x_i vale la disuguaglianza $g(x_\mathrm{i})/x_\mathrm{i} \le 3/2$ che è l'estremo superiore del caso $j = 2 = t_1$, mentre più in generale, per $j > 1 = m_1 = t_1 - 1$, il valore 3/2 rappresenta un upper bound non avvicinabile. Dunque la somma $\sum_{i=1}^{q} g(x_i)$ è maggiorata dal valore $\frac{3}{2}\sum_{i=1}^{q} x_i = \frac{3}{2} = \frac{1}{t_1 - 1} + \frac{1}{t_2 - 1} < G_p$.

Più in generale, se $x_1 \in \mathrm{I}(m_1)$, $x_2 \in \mathrm{I}(m_2)$, …, $x_{\mathrm{r}-1} \in \mathrm{I}(m_{\mathrm{r}-1})$, ma $x_\mathrm{r} \notin \mathrm{I}(m_\mathrm{r})$, con $r \le u$, allora tutti gli x_i, con $i \ge r$, sono non maggiori di $1/t_\mathrm{r}$ e ne consegue inoltre che per ogni $i \ge r$ l'upper bound dei rapporti $g(x_\mathrm{i})/x_\mathrm{i}$ è pari a $(t_\mathrm{r}+1)/t_\mathrm{r}$.

Infatti valgono le relazioni

$$\sum_{i=r}^{q} x_i = 1 - \sum_{i=1}^{r-1} x_i < 1 - \sum_{i=1}^{r-1} \frac{1}{t_r} = \frac{1}{t_r - 1} \tag{1.4}$$

avendo usato la definizione degli x_i per la prima uguaglianza, l'appartenenza $x_\mathrm{i} \in \mathrm{I}(m_\mathrm{i})$ per la disuguaglianza e la (1.2) per l'ultima uguaglianza. Se ne ricava che per $i \ge r$, gli elementi x_i possono appartenere solo ad insiemi I_b con $b \ge t_\mathrm{r} - 1$.

Dunque si ottiene

$$\sum_{i=1}^{q} g(x_i) = \sum_{i=1}^{r-1} g(x_i) + \sum_{i=r}^{q} g(x_i) \le \sum_{i=1}^{r-1} \frac{1}{t_i - 1} + \sum_{i=r}^{q} \frac{x_i (t_r + 1)}{t_r} < \sum_{i=1}^{r-1} \frac{1}{t_i - 1} + \frac{t_r + 1}{t_r (t_r - 1)}$$

ma vale anche

$$\frac{t_r + 1}{t_r (t_r - 1)} = \frac{1 + 1/t_r}{t_r - 1} = \frac{1}{t_r} + \frac{1}{t_r (t_r - 1)} = \frac{1}{t_r} + \frac{1}{t_{r+1} - 1}$$

per cui, essendo $r \le u$, si ottiene

$$\sum_{i=1}^{q} g(x_i) < \sum_{i=1}^{r+1} \frac{1}{t_i - 1} < G_p$$

Se infatti $r < u$, rispetto alla definizione di G_p manca almeno l'ultimo termine, mentre se $r = u$, l'ultimo termine risulta mancare del coefficiente $p/(p - 1)$, superiore ad 1.

L'ultimo caso possibile è quello per cui $x_\mathrm{i} \in \mathrm{I}(m_\mathrm{i})$ per $i = 1, 2, \ldots, u$. In tal caso, ne consegue, analogamente alla (1.4), che $\sum_{i=u+1}^{q} x_i < \frac{1}{t_{u+1} - 1}$, mentre $g(x_\mathrm{i}) = x_\mathrm{i} \cdot p/(p - 1)$ per $i \ge u+1$, visto che per tali indici gli elementi x_i appartengono ad insiemi I_b con $b \ge t_{\mathrm{u}+1} - 1$.

Se ne ricava

$$\sum_{i=1}^{q} g(x_i) = \sum_{i=1}^{u} g(x_i) + \sum_{i=u+1}^{q} g(x_i) = \sum_{i=1}^{u} \frac{1}{t_i - 1} + \sum_{i=u}^{q} \frac{x_i \cdot p}{(p-1)} < \sum_{i=1}^{u} \frac{1}{t_i - 1} + \frac{p}{(t_{u+1} - 1)(tp - 1)} = G_p$$

Tutto ciò mostra che G_p è un maggiorante per le somme $\sum_{i=1}^{q} g(x_i)$.

Il valore G_p è però anche estremo superiore di tali somme, e ciò può essere mostrato ponendo $x_\mathrm{i} = 1/(t_\mathrm{i}+1) + \varepsilon$ per $i \le u$ ed $x_{\mathrm{u}+1} = 1/(t_{\mathrm{u}+1} - 1) - u \cdot \varepsilon$, cui corrisponde una somma $\sum_{i=1}^{u+1} g(x_i) = G_p - \varepsilon \cdot \frac{u \cdot p}{p - 1}$ e cioè avvicinabile a piacere al valore di G_p se si fa tendere ε a zero.

Una tale sequenza x_1, x_2, …, x_{u+1} è di fatto una sequenza costruita in modo greedy poiché utilizza di volta in volta valori di x_i che tendono a massimizzare il rapporto $g(x_i)/x_i$.

Ai fini della prova per il caso bidimensionale, si associa a ciascun pezzo, di lunghezza l_i ed ampiezza w_i, un peso pari al prodotto $w_i \cdot g_p(l_i)$. Vale allora il seguente

Lemma 1.3. $\forall p \geq 2$ e dato un qualunque insieme Z di pezzi con dimensioni entrambe in [0, 1], la somma dei pesi di tutti i pezzi di Z, identificata dal simbolo $\Pi(Z)$, è non maggiore del prodotto $G_p \cdot \mathrm{OPT}(Z)$, essendo $\mathrm{OPT}(Z)$ l'ampiezza minima possibile di uno strip packing per i pezzi di Z.

Prova. La prova discende direttamente dal Lemma 1.2. Basta infatti considerare una generica sezione orizzontale su una soluzione ottima. Tale sezione interseca (o lambisce) un insieme di pezzi, di lunghezze l_1, l_2, …, l_q, la cui somma è non superiore ad 1. Di conseguenza la somma π delle utilità $g_p(l_i)$, per il lemma 1.2, è non superiore a G_p.

In Figura 1.23 si riporta una rappresentazione grafica di una possibile sezione, che tornerà utile anche per il caso k = 4-stage.

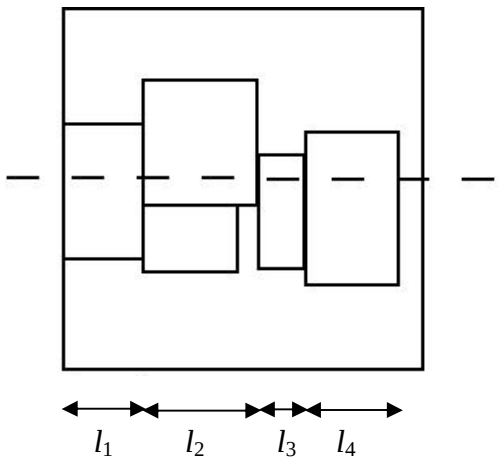


Figura 1.23 Sezione orizzontale

Definendo con $\pi(w)$ la somma $\sum_{i=1}^{q(w)} g(x_i)$ delle utilità dei $q(w)$ pezzi incontrati dalla sezione orizzontale alla coordinata verticale w e facendo un'operazione di integrazione in dw dei due membri della disequazione $\pi(w) \leq G_p$ sull'intera ampiezza $\mathrm{OPT}(Z)$, si ottiene la disequazione

$$\Pi(Z) \leq G_p \cdot \mathrm{OPT}(Z)$$

Senza il passaggio dalle lunghezze ai pesi, si sarebbe ottenuta la naturale relazione di maggiorazione di $\mathrm{OPT}(Z)$, area dello schema ottimo, rispetto alla somma delle aree dei pezzi di Z. Infatti la lunghezza della strip vale 1 per cui $\mathrm{OPT}(Z)$ è anche l'area della porzione di strip utilizzata.

Si passa infine a dimostrare un'ultima proprietà espressa dal seguente lemma.

Lemma 1.4. $\forall p \geq 2$, $\forall \alpha : 0 < \alpha < 1$, $\forall$ insieme Z di pezzi con dimensioni entrambe in [0, 1],

l'algoritmo SHELF(H_p, α) produce uno strip packing con ampiezza

$W(Z) \leq \Pi(Z)/\alpha + p/(1-\alpha)$, dove H_p è l'algoritmo armonico di parametro p.

Prova. Si denoti con Z_{sj} l'insieme dei pezzi associati nel primo passo alla classe C_s secondo il parametro α e secondo l'ampiezza dei pezzi stessi, e nel secondo passo alla sottoclasse associata all'intervallo I_j secondo il parametro p e secondo la lunghezza dei pezzi. L'ampiezza delle shelves in cui sono inseriti i pezzi di Z_{sj} vale dunque al più α^s, essendo tale valore un upper bound per le ampiezze dei relativi pezzi.

Per $j < p$, le shelves complete contengono j pezzi, di utilità pari ad $1/j$. Se allora si indica con z_{sj} il numero di tali shelves complete e con $\Pi(Z_{sj})$ il peso totale ad esse associato, vale la disuguaglianza $\Pi(Z_{sj}) \geq z_{sj} \cdot \alpha^{s+1}$.

Per $j = p$, le shelves complete hanno una lunghezza occupata non inferiore ad $1-1/p$, essendo $1/p$ il valore della massima lunghezza dei pezzi associati dall'algoritmo armonico H_p alla sottoclasse relativa all'intervallo I_p. Data la linearità di $g_p(x)$ nel caso di sottoclassi relative ad I_p, si può esprimere la somma delle utilità associate alle lunghezze come la somma delle stesse lunghezze moltiplicata per il coefficiente $p/(p-1)$, secondo la definizione di $g_p(x)$. Un lowe bound per il valore $\Pi(Z_{sp})$ è allora dato dall'ipotesi peggiore in cui l'ampiezza è pari al minimo possibile, cioè α^{s+1}. Ne consegue $\Pi(Z_{sp}) \geq z_{sp} \cdot \alpha^{s+1}$, come per il caso $j < p$, grazie al fatto che il fattore $p/(p-1)$ è bilanciato dal valore $1-1/p$ di minima lunghezza occupata.

La conseguenza è che valgono le relazioni

$$\Pi(Z) = \sum_{j=1}^{p} \sum_{s=0}^{+\infty} \Pi(Z_{sj}) \geq \sum_{j=1}^{p} \sum_{s=0}^{+\infty} z_{sj} \alpha^{s+1} = \sum_{j=1}^{p} \sum_{s=0}^{+\infty} (z_{sj} + 1) \alpha^{s+1} - \sum_{j=1}^{p} \sum_{s=0}^{+\infty} \alpha^{s+1} =$$

$$\alpha \sum_{j=1}^{p} \sum_{s=0}^{+\infty} (z_{sj} + 1) \alpha^{s} - p \frac{\alpha}{1-\alpha}$$

Il primo addendo dell'espressione finale, eccezion fatta per il coefficiente α, è però un maggiorante del valore $W(Z)$ determinato dall'algoritmo, e che uguaglia se $\forall s > 0$ e $\forall j \in \{1, 2, \ldots, p\}$ rimane una shelf non aperta, e tutte le shelves sono di ampiezza α^s. Ovviamente il numero infinito di shelves che dovrebbe restare aperto implica che le seguenti disuguaglianze, equivalenti, valgono in senso stretto, provando così il lemma.

$$\Pi(Z) > \alpha \cdot W(Z) - p \frac{\alpha}{1-\alpha} \Leftrightarrow W(Z) < \frac{\Pi(Z)}{\alpha} + \frac{p}{1-\alpha}$$

Dai lemmi 1.3 ed 1.4 si ricava che $W(Z) < \frac{G_p}{\alpha} OPT(Z) + \frac{p}{1-\alpha}$. Dunque, scelto p opportunamente elevato ed α opportunamente prossimo ad 1, si può ottenere una ratio asintotica comunque vicina al valore h_∞ cui tende G_p al crescere di p. Il prezzo da pagare è una costante $p / (1 - \alpha)$ che crescere come un infinito di ordine due.

Dunque risulta dimostrata l'esistenza di un algoritmo 3-stage, in particolare di un algoritmo on-line a shelf del tipo SHELF(A, α), in grado di avere ratio asintotica comunque prossima ad h_∞.

Per dimostrare ora che h_∞ è il minimo valore per cui vale tale proprietà, si costruisce dapprima una apposita istanza $I_{\alpha,\varepsilon}$ per un fissato intero α ed un fissato valore reale ε, entrambi positivi. Per tale istanza si mostra la validità della relazione $G_3(I_{\alpha,\varepsilon}) > (h_\infty - \varepsilon/2) \cdot OPT(I_{\alpha,\varepsilon})$,

nonché la crescita proporzionale fra $G_3(I_{\alpha,\varepsilon})$ ed α. Si ottiene poi il risultato considerando valori di α divergenti. Si capirà più avanti la necessità del fattore 1/2 accanto ad ε.

Al contrario di quanto fatto nel lavoro di Csirik e Woeginger (1997) per provare tale risultato negativo restringendo l'attenzione alla sola classe di algoritmi SHELF(A, α), in questo caso risulta necessaria l'ipotesi di lunghezze tutte diverse fra i pezzi dell'istanza $I_{\alpha,\varepsilon}$.

Innanzitutto si valuta il minimo valore finito *d* che soddisfi la disuguaglianza $\sum_{i=1}^{d} \frac{1}{t_i - 1} \geq h_\infty - \frac{\varepsilon}{2}$. Tale posizione sarà utile alla fine della dimostrazione.

Si definisce quindi il valore $\delta = \frac{1}{d}\left(1 - \sum_{i=1}^{d} \frac{1}{t_i}\right)$ che è sicuramente positivo data la (1.1).

In particolare il valore di δ è inferiore alla differenza $\frac{1}{t_{d-1}} - \frac{1}{t_d}$ poiché δ dell'ordine di $(t_{d-1})^{-4}$ mentre la differenza indicata è dell'ordine di $(t_{d-1})^{-1}$. Infatti, utilizzando la (1.2), si ottiene $\delta = \frac{1}{d} \cdot \frac{1}{t_d (t_d - 1)}$.

Come si vedrà più avanti, la validità di tale relazione semplifica la complicazione descrittiva della procedura costruttiva di $I_{\alpha,\varepsilon}$. Tuttavia l'unica condizione necessaria ai fini della dimostrazione è che la sommatoria $\sum_{i=1}^{\infty} \frac{1}{t_i}$ sia non maggiore di 1, così da poter scegliere δ positivo. Naturalmente se la sequenza $\{t_i\}_{i \in N}$ cambia, allora il teorema risulta dimostrato per un valore diverso da h_∞, ma certamente inferiore ad esso, data la validità della prima parte del teorema. Si sottolinea in particolare che, sebbene la sequenza $\{t_1 = 2, t_2 = 2, t_i = 0 \ \forall \ i \geq 3\}$, per la quale vale la (1.1), è tale che $\sum_{i=1}^{\infty} \frac{1}{t_i - 1} = 2 \geq h_\infty$, tale sequenza ha il difetto di avere solo un numero finito di elementi non nulli, determinando con ciò un valore di δ nullo per *d* sufficientemente grande e dunque l'inammissibilità del procedimento costruttivo che è alla base della prova.

L'istanza $I_{\alpha,\varepsilon}$ è costruita secondo il seguente procedimento. Di essa fanno parte *d* gruppi di pezzi, $Z_1, Z_2, \ldots, Z_d$. Il primo gruppo è costituito da $n_1 = \alpha$ elementi di ampiezza W_1 pari ad 1 e di lunghezze tutte diverse, contenute nell'intervallo aperto $]1/2, 1/2 + \delta[$, dove 1/2 è scelto poiché corrisponde ad $1/t_1$. Detto inoltre N_{i-1} il numero di pezzi dei gruppi di indice da 1 ad $i - 1$, allora il gruppo Z_i ha n_i pezzi con $n_i = \alpha \cdot (t_i - 1) \cdot N_{i-1}$. I pezzi del gruppo Z_i hanno tutti un'ampiezza W_i tale che $n_i \cdot W_i = \alpha$ e lunghezze nell'intervallo aperto $]1/t_i, 1/t_i + \delta[$. Come si vedrà la scelta di avere $n_i \cdot W_i = \alpha$ risulterà funzionale alla dimostrazione così come la posizione $n_i = \alpha \cdot (t_i - 1) \cdot N_{i-1}$.

La precisazione fatta precedentemente su δ rende semplice capire che, una volta scelte lunghezze diverse in ciascun gruppo, allora l'intera istanza del problema non ha alcuna coppia di pezzi di pari lunghezza. Tale proprietà è importante ai fini della dimostrazione. Si torna comunque sul fatto che anche laddove vi fosse sovrapposizione fra gli intervalli aperti indicati, basterebbe comunque scegliere lunghezze tutte diverse, avendo cura di confrontare anche pezzi di gruppi diversi.

Il primo passo della dimostrazione consiste nel costruire una apposita soluzione su quattro livelli di taglio. Tale soluzione consiste di α shelves di ampiezza unitaria, ciascuna con *d* fette verticali dette *crates*. Tali crates contengono pezzi dello stesso gruppo ed in ogni shelf c'è un crate per ciascun gruppo. Poiché i pezzi di ciascun gruppo Z_i sono lunghi al più $1/t_i + \delta$, allora la lunghezza del crate non supera tale valore e, di conseguenza, affiancando *d*

crates, uno per ciascun gruppo, si ottiene una lunghezza totale di al più $\sum_{i=1}^{d}\left(\frac{1}{t_i}+\delta\right) = d\cdot\delta + \sum_{i=1}^{d}\frac{1}{t_i}$ che vale appunto 1 per costruzione di δ. In ciascun crate vengono posti n_i/α pezzi, per cui l'ampiezza di ciascun crate è $W_i \cdot n_i/\alpha$ che vale 1 per costruzione. Inoltre, per ciascun gruppo, i pezzi vengono esauriti da α crates e ciò vuol dire che con α shelves di ampiezza unitaria si utilizzano tutti i pezzi. Dunque esiste una soluzione (su quattro livelli di taglio) di ampiezza α e di conseguenza α è un upper bound del valore minimo di ampiezza utilizzata, raggiungibile con il packing ottimo.

Il secondo passo consiste nel valutare un lower bound del valore ottimo raggiungibile con soluzioni di al più tre livelli. Poiché i pezzi hanno tutti lunghezza differente, in una soluzione a tre livelli, ciascun crate può contenere al più un pezzo, ma non due, indipendentemente dal fatto che appartengano o meno allo stesso gruppo. Si indica allora con x_i il numero di shelves che hanno almeno un pezzo del gruppo Z_i e nessuno di quelli dei gruppi da Z_1 a Z_{i-1}. Una tale shelf è denotata come di tipo *i*. Si indica inoltre con X_{i-1} la somma $x_1+x_2+\ldots+x_{i-1}$. Dunque un pezzo qualsiasi del gruppo Z_i è in una delle $x_i + X_{i-1}$ shelves di gruppo non superiore ad *i*. Ciò implica che

$$n_i \geq x_i + X_{i-1}$$

ma soprattutto che, poiché in una shelf possono esserci al più $t_i - 1$ crates-pezzi del gruppo Z_i,

$$n_i \leq (x_i + X_{i-1})\cdot(t_i - 1) \leq (x_i + N_{i-1})\cdot(t_i - 1)$$

dove in tale implicazione interviene l'utilizzo di intervalli aperti per la scelta delle lunghezze dei pezzi, nonché la relazione $X_{i-1} \leq N_{i-1}$.

Dalla posizione fatta, per $i > 1$, su n_i/α, pari appunto a $(t_i - 1)\cdot N_{i-1}$, si ottiene

$$x_i \geq (\alpha - 1)\cdot N_{i-1} = \frac{\alpha - 1}{\alpha}\cdot\frac{n_i}{t_i - 1}.$$

Per $i = 1$, si può scrivere $x_1 \geq n_1 = \alpha$ e quindi, essendo $t_1 = 2$, vale la relazione ancora più forte

$$x_1 \geq n_1/(t_i - 1) = 1 + \frac{\alpha - 1}{\alpha}\cdot\frac{n_i}{t_i - 1}$$

L'ampiezza della soluzione consiste nella somma delle ampiezze delle shelves e dunque nella sommatoria $\sum_{i=1}^{d} x_i \cdot W_i$, ma poiché $n_i\cdot W_i = \alpha$, risulta che tale ampiezza totale maggiora la sommatoria $\sum_{i=1}^{d}\frac{(\alpha - 1)}{t_i - 1}$, avendo oltretutto tralasciato l'addendo unitario ottenuto per $i = 1$. Tale sommatoria è dunque un lower bound per $G_3(I_{\alpha,\varepsilon})$.

Il terzo ed ultimo passo consiste nel mostrare che il rapporto fra il lower bound di $G_3(I_{\alpha,\varepsilon})$ e l'upper bound di $OPT(I_{\alpha,\varepsilon})$ vale $\frac{(\alpha - 1)}{\alpha}\sum_{i=1}^{d}\frac{1}{t_i - 1}$, cioè maggiore o uguale a $\frac{(\alpha - 1)}{\alpha}\left(h_\infty - \frac{\varepsilon}{2}\right)$, data la definizione di *d* che permette la maggiorazione del valore $(h_\infty - \varepsilon/2)$.

Affinché tale lower bound di $G_3(I_{\alpha,\varepsilon})/OPT(I_{\alpha,\varepsilon})$ sia superiore ad $(h_\infty - \varepsilon)$ è sufficiente porre $\alpha \geq 2\frac{h_\infty}{\varepsilon} - 1$, sottolineando che la scelta iniziale di un coefficiente 1/2 al fianco di ε è necessaria per poter qui ottenere tale valore minimo di α. Questo dimostra il teorema.

Da ciò si può ricavare che, per ogni fissato ε > 0 e comunque sia fissata una costante $c > 0$, esiste sempre un'istanza **I** per la quale valga la relazione

$$G_3(I) > (h_\infty - \varepsilon)\cdot OPT(I) + c$$

che è equivalente a

$$G_3(I)/\ OPT(I)\ > (h_\infty - \varepsilon) + c\ /\ OPT(I).$$

Per provarlo, ci si basa sulla prima parte del teorema. Si supponga α sufficientemente grande, diciamo superiore al valore, denotato con $\alpha_{2h\infty}$, che permette di costruire, con il processo utilizzato nel teorema, istanze per le quali $G_3(I)\ /\ OPT(I) < 2h_\infty$, e cioè tali che $OPT(I) > \alpha/(2h_\infty)$. Allora basta considerare l'istanza $I_{\alpha,\varepsilon/2}$ con un valore di α che soddisfi anche la disuguaglianza $(2c \cdot h_\infty)/\alpha < \varepsilon/2$, poiché ciò implica le relazioni:

$$G_3(I_{\alpha,\varepsilon/2})/\ OPT(I_{\alpha,\varepsilon/2})\ > (h_\infty - \varepsilon/2) = (h_\infty - \varepsilon) + \varepsilon/2 > (h_\infty - \varepsilon) + c\ /\ OPT(I).$$

La posizione da fare è allora $\alpha \geq \max\{\alpha_{2h\infty}\ ,\ 4c \cdot h_\infty/\varepsilon\ ,\ 2h_\infty/\varepsilon\ \}$, per poi costruire $I_{\alpha,\varepsilon/2}$.

1.4.3 Caso di strip packing a quattro livelli

Il lavoro di Seiden e Woeginger (2005), che si sta descrivendo, riprende e dimostra con un procedimento più compatto il risultato già ottenuto da Kenyon e Remilla (2000). Tale risultato afferma che è possibile costruire, per qualunque $\varepsilon > 0$, un algoritmo di complessità polinomiale che abbia una ratio asintotica inferiore ad $1 + \varepsilon$.

Si riporta la prova fornita da Seiden e Woeginger (2005), sottolineando le finalità con cui vengono fatte le diverse posizioni.

Fissato un $\varepsilon > 0$, si stabilisce un valore intero positivo s proporzionale ad $1/\varepsilon$ secondo un legame mostrato alla fine della prova. Si definisce inoltre $t = s^2$ ed infine $\delta = 1/s$. Ne consegue tra l'altro che $t \cdot \delta = s$.

Il valore di δ è utilizzato per selezionare i pezzi conseguentemente considerati piccoli ed estromessi dai problemi rilassati in seguito usati. Tale selezione permette di ottenere un problema lineare rilassato con un numero di variabili che cresce con s in un modo da poter essere risolto in tempo polinomiale.

Sono dunque considerati "grandi" i pezzi con lunghezza superiore a δ, che decresce pressoché linearmente con ε. Si definisce con W l'ampiezza totale di tali pezzi "grandi". Se dunque A_Z è l'area totale di tutti i pezzi, essa è non inferiore al prodotto $\delta \cdot W$ che è appunto un lower bound, non eguagliabile, dell'area dei pezzi grandi.

Si ipotizza inoltre che $W \geq 2t$. Le istanze del problema che non soddisfano questa ipotesi, hanno un'area totale di pezzi grandi limitata dall'upper bound $\delta \cdot 2t$. Alla fine della prova, si potrà allora aumentare di tale quantità la costante che interviene nella relazione di ratio asintotica, così da tenere in considerazione anche queste istanze limitate nel valore di ottimo. Nel seguito si descriverà il motivo per cui si è fatta l'ipotesi $W / t \geq 2$.

I pezzi grandi sono divisi in t gruppi $B_1, B_2, \ldots, B_t$, costruiti considerando tali pezzi ordinati per lunghezza non crescente, ed associando a ciascun B_i un insieme di pezzi consecutivi in maniera tale che valgano due proprietà:

a) la somma W_i delle ampiezze dei pezzi di ciascun B_i appartiene all'intervallo $]W/t - 1,\ W/t + 1[$
b) denotate con L_i , per $i = 1, 2, \ldots, t$, le lunghezze, per ciascun gruppo B_i, del primo pezzo ad esso associato, e con L_{t+1} la lunghezza del più stretto (in termini di ampiezza) pezzo grande (associato quindi al gruppo B_t), valgono le relazioni $L_1 \geq L_2 \geq \ldots \geq L_t \geq L_{t+1} > \delta$, dove l'ultima è dovuta alla definizione di pezzi grandi.

Una tale partizione dei pezzi grandi non è necessariamente unica, ma ciò che conta è che ne esista certamente almeno una. Ma questo può essere dimostrato informalmente con il supporto della Figura 1.24.

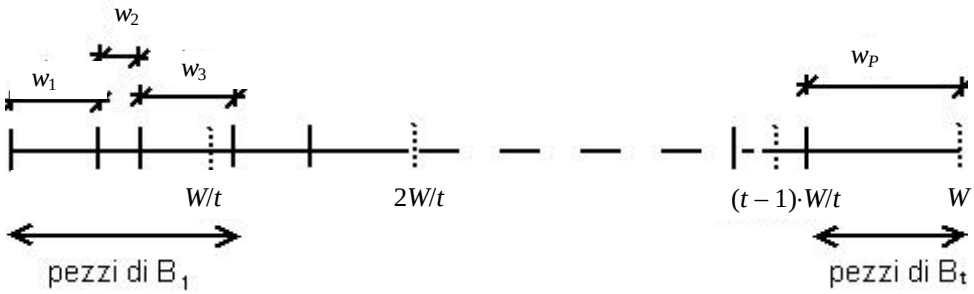


Figura 1.24 Sequenza dei pezzi

Se P è il numero totale di pezzi grandi, i tratti verticali continui in figura rappresentano le somme parziali delle ampiezze dei primi (secondo l'ordine delle lunghezze) p pezzi, con $p = 1, 2, \ldots, P$. I tratti verticali tratteggiati denotano invece i multipli di W/t.

Una possibile partizione che soddisfi le proprietà a) e b) si ottiene allora associando a ciascun B_i innanzitutto i pezzi che, in termini di ampiezza, cadano completamente nella i-mo intervallo, mente i pezzi a cavallo di due intervalli vengono associati al gruppo relativo all'intervallo precedente. In tal modo ad ogni B_i viene associato un insieme di pezzi consecutivi le cui ampiezze, sommate, possano dare un valore comunque vicino, ma inferiore, a $W/t + 1$ nel caso di sola aggiunta a destra ed un valore comunque vicino, ma superiore, a $W/t - 1$ nel caso di sola sottrazione a sinistra. E' facile mostrare che se si rafforzasse la proprietà a) considerando intervalli del tipo $]W/t - 1 + \varepsilon, W/t + 1 - \varepsilon[$ con ε piccolo a piacere, esisterebbe sempre un insieme totale di pezzi per il quale non esista nessuna partizione che soddisfi le due proprietà.

Si sottolinea che per poter avere pezzi a cavallo di al più una soglia, è necessaria l'ipotesi $W/t \geq 2$ precedentemente fatta e che trova in questo punto la sua motivazione. In caso contrario, si rischierebbe di avere gruppi di pezzi senza alcun pezzo associato e cioè con valore W_i nullo, facendo venire meno l'ipotesi di positività delle W_i che sarà usata in seguito.

La procedura costruttiva prevede di risolvere dapprima un problema LP di linear programming, il cui valore di ottimo (minimo) è mostrato essere sufficientemente vicino ad OPT(I), per poi costruire a partire dalla soluzione ottima di LP una soluzione di **I** mostrando che il peggioramento è sufficientemente contenuto ai fini della validità del teorema.

Per poter definire LP e mostrare le proprietà di LP(I), vanno innanzitutto definite due istante modificate di **I**, dette I^- ed I^+.

Per ciascuna delle due istanze si lasciano inalterati i pezzi piccoli. I pezzi grandi vengono ingranditi nel caso di I^+ e ridotti nel caso di I^-, alterando le lunghezze.

Per l'istanza I^+ le lunghezze dei pezzi di B_i vengono aumentate ad L_i, mentre per l'istanza I^- esse vengono ridotte ad L_{i+1}. Per analogia si identifica con B_i^+ (risp. B_i^-) il corrispondente insieme per l'istanza I^+ (risp. I^-), analogo al gruppo B_i per l'istanza **I**.
Appare chiaro che valgono le relazioni

$$OPT(I^-) \leq OPT(I) \leq OPT(I^+).$$

Per ciascuna delle due istanze si introduce un problema di linear programming, denotando tali problemi rispettivamente con LP^+ ed LP^-. Il primo è usato nella procedura costruttiva, mentre il secondo interviene per dimostrare la validità della soluzione che si ottiene dal primo.

La definizione di questi due problemi di linear programming si basa sull'idea di sezioni orizzontali di una soluzione dello Strip Packing, come già mostrato in Figura 1.23, durante le considerazioni per il caso k = 3-stage. In particolare qui si definisce *pattern* un vettore Q di t elementi $(q_1, q_2, \dots, q_t)$ dove il valore q_i rappresenta il numero di pezzi di lunghezza L_i individuati dalla sezione. Si sottolinea che per l'istanza I^+ (risp. I^-) q_i rappresenta il numero di pezzi toccati del gruppo B_i^+ (risp. B_{i-1}^-). Poiché ciascun L_i è maggiore di δ, il numero q_i di pezzi, con tale lunghezza, individuati dalla sezione è inferiore ad $1/\delta$. Ciò limita il numero di possibili pattern al valore superiore $(1/\delta)^t$, cioè s^{s^2}. Questo limite motiva la divisione fra pezzi grandi e piccoli, poiché i problemi LP contengono una variabile per ogni possibile pattern. Alla fine del paragrafo si tornerà su questo aspetto.

Si noti che, detto $Q(\nu)$ il pattern identificato da una sezione alla coordinata verticale ν su una generica soluzione dello Strip Packing, allora l'integrazione di $q_i(\nu)$ nella variabile ν determina il prodotto fra il numero di pezzi di lunghezza L_i e la somma delle ampiezze di tali pezzi. Per l'istanza I^+ tale somma è pari a W_i, mentre per l'istanza I^- essa è pari ad W_{i-1} poiché i pezzi di lunghezza L_i sono, nell'istanza I^-, quelli del gruppo B_{i-1} dell'istanza **I**, ed a cui è associata appunto l'ampiezza W_{i-1}.

Per ogni valore di *i*, il contributo dato all'integrale da ciascun pattern Q interviene solo quando ν identifica una sezione cui è associato appunto il pattern Q. Inoltre a ciascuna sezione è associato uno ed un solo pattern, eccezion fatta per un numero finito di coordinate ν che danno contributo nullo all'integrale e che corrispondono alle coordinate per le quali la sezione lambisce sul lato inferiore, o su quello superiore, qualche pezzo. Per ciascun fissato pattern Q il valore q_i è costante. Ciò vuol dire che, suddiviso l'integrale in una somma di contributi dovuti a ciascun possibile pattern Q, ognuno di questi contributi può essere espresso come $q_i \int \delta(q,\nu) d\nu$, essendo $\delta(q,\nu)$ una funzione che vale 1 se la sezione alla coordinata ν identifica il pattern Q e 0 altrimenti.

Si definisce allora come $\Phi(Q)$ il valore dell'integrale $\int \delta(q,\nu) d\nu$, indicato come 'porzione di ampiezza totale associata alle sezioni che individuano Q'. Tale valore cambia al variare della soluzione dello Strip Packing su cui è definito, ma è comunque non negativo.
Per quanto detto in precedenza, per l'istanza I^+ valgono le uguaglianze:

$$\sum_{Q \in \Omega} q_i \Phi(Q) = W_i \qquad \forall i \in \{1, \dots, t\}$$

dove con Ω si denota l'insieme di tutti i possibili pattern.
Il problema LP^+ è allora definito nel seguente modo:

$$\begin{cases} \min \quad \sum_{Q \in \Omega} \Phi(Q) \\ s.t. \quad \sum_{Q \in \Omega} q_i \Phi(Q) \geq W_i \qquad \forall i \in \{1, \dots, t\} \\ \Phi(Q) \geq 0 \qquad \forall Q \in \Omega \end{cases}$$

dove i pattern Q e le relative componenti q_i sono fissate e costanti e le $\Phi(Q)$ sono le variabili.

Poiché tutti i vincoli sono soddisfatti dalle soluzioni dell'istanza I^+, il valore ottimo di LP^+ rappresenta un lower bound per $OPT(I^+)$, ma non ha a priori alcuna relazione certa né con con OPT(I), né con $OPT(I^-)$.

Per mostrare allora che il valore ottimo di LP^+ è sufficientemente vicino ad OPT(I), si definisce anche il problema LP^-, che per analoghi motivi rappresenta un rilassamento combinatorio del problema I^-. Esso esclude i pezzi del gruppo B_t^- ed è così definito:

$$\begin{cases} \min \quad \sum_{Q\in\Omega} \Psi(Q) \\ s.t. \quad \sum_{Q\in\Omega} q_i \Psi(Q) \geq W_{i-1} \qquad \forall i \in \{2,\ldots,t\} \\ \Psi(Q) \geq 0 \qquad \forall Q \in \Omega \end{cases}$$

Si fa utilizzo della notazione con la lettera Ψ anziché Φ per rappresentare il fatto che si tratta di una variabile del problema LP^-. Si noti inoltre che vi è un vincolo in meno rispetto ad LP^+ dato che i va da 2 a t e non da 1 a t. Valgono inoltre le relazioni

$$OPT(LP^-) \leq OPT(I^-) \leq OPT(I) \quad , \quad OPT(LP^-) \leq OPT(LP^+).$$

Si mostra ora come sia possibile passare dalla soluzione ottima Ψ^* di LP^- ad una soluzione fattibile Φ' di LP^+.

Innanzitutto si noti che LP^+ ha $|\Omega|$ variabili e t vincoli di disuguaglianza. Ciò vuol dire che le soluzioni basiche di LP^+ hanno al più t valori non nulli.

Per LP^- vale un'analoga proprietà, con il valore t–1. Inoltre, l'assenza di vincoli per i = 1, implica che, per il pattern Δ = (1, 0, …, 0), si ha $\Psi^*(\Delta) = 0$, dato l'obiettivo di minimizzazione.

La soluzione Φ' di LP^+ è allora così definita:

$$\Phi'(Q) = \begin{cases} \frac{W}{t} + 1 & se \quad Q = \Delta \\ 0 & se \quad \psi^*(Q) = 0 \quad \text{e} \quad Q \neq \Delta \\ \psi^*(Q) + 2 & se \quad \psi^*(Q) > 0 \end{cases}$$

e permette di definire un upper bound per il valore $OPT(LP^+)$ sufficientemente vicino al valore di Ψ^* e quindi di $OPT(LP^-)$.

Infatti Φ' è prima di tutto una soluzione fattibile di LP^+ poiché, per i = 1, vale

$$\sum_{Q\in\Omega} q_i \Phi'(Q) \geq \Phi'(\Delta) = \frac{W}{t} + 1 > W_1$$

dove nel primo passaggio si è estratto il solo termine della sommatoria relativo al pattern Δ, che infatti ha l'elemento di indice i = 1 con valore 1, nel secondo si è utilizzata la definizione di Φ' e nel terzo è intervenuto il modo di costruire i gruppi B_1, …, B_t con la relativa limitazione sulla somma delle ampiezze dei pezzi di ciascun gruppo.

Per un fissato i > 1, data la validità del relativo vincolo di LP^-, e per la positività di W_{i-1}, esiste almeno un pattern R che abbia i-mo elemento, denotato con r_i, maggiore o uguale ad 1 e tale che $\Psi^*(R) > 0$. Dunque $\Phi'(R) = \Psi^*(R)+2$ e ciò comporta che

$$\sum_{Q\in\Omega} q_i \Phi'(Q) \geq \sum_{Q\in\Omega} q_i \psi^*(Q) + 2r_i \geq W_{i-1} + 2 \geq W_i$$

dove nell'ultima disuguaglianza è stata usata ancora la proprietà che limita tutte le W_i nello stesso intervallo $]W/t - 1 + \varepsilon, W/t + 1 - \varepsilon[$, di ampiezza almeno pari a 2.

Analizzando inoltre il valore associato alla soluzione Φ', si ottiene

$$\sum_{Q\in\Omega} \Phi'(Q) =$$

$$\Phi'(\Delta) + \sum_{Q\in\Omega \,:\, \psi(Q)=0, Q\neq\Delta} \Phi'(Q) + \sum_{Q\in\Omega \,:\, \psi^*(Q)>0} \Phi'(Q) = \left(\frac{W}{t} + 1\right) + \sum_{Q\in\Omega \,:\, \psi(Q)=0, Q\neq\Delta} (\psi^*(Q) + 2) \leq$$

$$\left(\frac{W}{t} + 1\right) + 2(t-1) + OPT(LP^-)$$

avendo sfruttato il fatto che al più t – 1 valori di Ψ^* sono non nulli.

Inoltre, dalla disuguaglianza $A_Z \geq \delta \cdot W$, si ottiene

$$OPT(LP^+) < OPT(LP^-) + 2t + A_Z/(\delta \cdot t)$$

e quindi

$$OPT(LP^+) < OPT(I) + 2t + A_Z / s \qquad (1.5)$$

La costruzione della soluzione approssimata per l'istanza **I** parte dalla soluzione Φ^*, ottima per LP^+. Essa prevede l'utilizzo, per ogni pattern $Q \in \Omega$, di un numero di shelves pari a $\lceil \Phi^*(Q)/s \rceil$, tutte di ampiezza $(s+1)$ e contenenti ciascuna q_i potenziali crates di lunghezza L_i ed appunto ampiezza $(s+1)$. Tali crates sono riempiti ponendo in pila l'uno sull'altro i pezzi dell'insieme B_i^+ scelti di volta in volta con strategia greedy, nel senso che si satura un crate per volta prima di utilizzarne un altro, inserendo i pezzi in sequenza per ampiezza non crescente così da riporre l'attenzione solo sul corrente crate aperto. Se nel procedimento di riempimento di tutti i potenziali crates individuati, i pezzi di B_i^+ terminano, allora i crates potenziali ancora non aperti, e cioè rimasti vuoti, vengono eliminati, lasciando nella relativa shelf una maggiore lunghezza sfruttabile per i crates con pezzi piccoli come si dirà in seguito.

Questa eliminazione è necessaria per poter dimostrare la validità del teorema nel caso, descritto in seguito, in cui il posizionamento dei pezzi piccoli determini l'utilizzo di nuove shelves. Poiché le ampiezze di tali pezzi sono non maggiori di uno, allora ciascun crate saturato da tale strategia di riempimento è pieno ad almeno un livello di ampiezza s. Può inoltre esservi un solo crate non saturato, a causa dell'esaurimento dei pezzi di B_i^+.

Si coglie qui l'occasione per indicare che in casi pratici si può pensare di utilizzare strategie di riempimento dei crates più sofisticate, prese in prestito dalla teoria del Bin Packing mono-dimensionale.

Si dimostra ora innanzitutto che c'è spazio a sufficienza in tali crates per inserire tutti i pezzi di ciascun B_i^+. Infatti il numero totale di crates in cui sono inseriti i pezzi di B_i^+ è $\sum_{Q\in\Omega} \left(q_i \cdot \left\lceil \Phi^*(Q)/s \right\rceil \right)$ che è non minore di W_i/s, grazie al vincolo $\sum_{Q\in\Omega} q_i \Phi^*(Q) \geq W_i$ presente nel problema LP^+. Se dunque per assurdo non vi fosse spazio a sufficienza, ciò vorrebbe dire che tutti i crates di lunghezza L_i sarebbero saturi e cioè pieni ad un livello di ampiezza almeno pari ad s. Dunque l'ampiezza totale usata all'interno di tali crates sarebbe pari ad $s \cdot \sum_{Q\in\Omega} \left(q_i \cdot \left\lceil \Phi^*(Q)/s \right\rceil \right)$, che è non minore di W_i. Ma ciò contraddice l'ipotesi, secondo cui la costruzione di tali crates, avvenuta ponendo in pila l'uno sull'altro i pezzi di B_i^+, avrebbe prodotto un'ampiezza totale usata inferiore alla somma delle ampiezze dei pezzi di B_i^+ stesso e cioè inferiore a W_i.

Il secondo passo dell'algoritmo costruttivo prevede di posizionare i pezzi piccoli, cioè di lunghezza non maggiore di δ. Tali pezzi sono divisi in classi, definendo di classe μ, con μ valore intero non negativo, un pezzo piccolo la cui lunghezza λ soddisfa entrambe la relazione $\delta(1-\delta)^{\mu+1} < \lambda \leq \delta(1-\delta)^{\mu}$. Per ciascuna classe si costruiscono appositi crates di ampiezza $(s+1)$ riempiti con pezzi posti in pila l'uno sull'altro, con la stessa strategia, e quindi le stesse proprietà di riempimento, utilizzata per i pezzi grandi. A partire da $\mu = 0$, ciascun crate è inserito con tecnica First-Fit in una delle shelves costruite nel primo passo, cioè per i pezzi grandi. Se per un crate non c'è spazio a sufficienza in nessuna delle shelves già costruite, si introduce una nuova shelf, in cui entreranno solo pezzi piccoli.

Il passo finale consiste nel sostituire ai pezzi dell'istanza I^+ i corrispondenti pezzi dell'istanza **I**, ottenendo così una soluzione 4-staged per l'istanza **I**. Si nota che la soluzione per I^+ è 4-staged solo a causa della presenza, per qualche $\mu \geq 0$ di eventuali pezzi piccoli di differente lunghezza nella classe μ.

Si sviluppa allora un upper bound considerando separatamente due casi:

a) il secondo passo non introduce alcuna nuova shelf.
b) il secondo passo introduce una o più nuove shelves.

Nel primo caso l'ampiezza APP(I) della soluzione così costruita è pari ad (s+1) moltiplicato il numero di shelves ottenute dalla prima fase, e cioè

$$\text{APP(I)} = (s+1) \cdot \sum_{Q \in \Omega} \left\lceil \Phi^*(Q)/s \right\rceil \le (s+1) \cdot \left(t + \sum_{Q \in \Omega} \frac{\Phi^*(Q)}{s} \right)$$

dove la disuguaglianza è ottenuta dalla constatazione che al più t elementi di Φ^* possono essere non nulli. Si noti infatti che nell'espressione a destra non c'è più l'approssimazione per eccesso.

Si sottolinea che se l'eliminazione di potenziali crates determinasse lo svuotamento di qualche shelf, essa sarebbe mantenuta per fare spazio ai pezzi piccoli, così come avviene per la lunghezza residua, inferiore ad 1, delle altre shelves. Nel caso particolare in cui una tale shelf non dovesse essere utilizzata neanche per i pezzi piccoli, si otterrebbe una disuguaglianza stretta, rafforzando cioè la prova, che resterebbe quindi valida.
Ora, poiché la sommatoria dei valori $\Phi^*(Q)$ è pari ad OPT(LP$^+$), risulta

$$\text{APP(I)} \le (s+1) \cdot t + \frac{s+1}{s}\left(OPT(I) + 2t + \frac{A_Z}{s} \right) \le \left(1 + \frac{1}{s}\right)^2 OPT(I) + s(s+1)(s+2)$$

avendo usato la (1.5), che lega OPT(LP$^+$) ad OPT(I), nella prima disuguaglianza, e la relazione $A_Z \le$ OPT(I) nella seconda.

Se allora si sceglie s in modo tale da rendere $(1 + 1/s)^2 < 1 + \varepsilon$, si ottiene una ratio asintotica inferiore ad $(1 + \varepsilon)$, al prezzo di una costante aggiuntiva che tende ad essere proporzionale ad $(1/\varepsilon)^3$.
La disuguaglianza equivale a

$$s \ge \frac{1 + \sqrt{1+\varepsilon}}{\varepsilon} \tag{1.6}$$

Per valori di ε sufficientemente piccoli, la relazione diventa $s \ge \frac{2}{\varepsilon}$, mentre più in generale una condizione sufficiente è $s \ge \frac{2}{\varepsilon} + \sqrt{\frac{1}{\varepsilon}}$, ottenuta utilizzando $\sqrt{1+\varepsilon} < 1 + \sqrt{\varepsilon}$.

Se invece la seconda fase introduca nuove shelves, da un lato ciò è un caso peggiore, ma dall'altro permette di essere certi di alcuni fattori di riempimento, come si discute nel seguito.

I crates saturi della prima fase, per l'istanza I$^+$, sono riempiti per l'intera lunghezza poiché tutti i pezzi di B_i^+ hanno lunghezza identica. Dunque essi hanno un fattore di riempimento di almeno $s/(s+1)$ e c'è al più un crate non saturo per ciascun indice $i \in \{1, 2, \ldots, t\}$. La presenza di al più un crate non saturo è dovuta all'ipotesi di eliminazione dei crates potenziali mai aperti. Ciascuno dei crates non saturi determina un'area aggiuntiva pari all'area di un equivalente crate vuoto e cioè $L_i \cdot (s+1)$. Per i crates saturi di classe μ, riempiti con pezzi piccoli, il fattore di riempimento è almeno $(1 - \delta) \cdot s/(s+1)$ poiché la lunghezza del crate è pari a $\delta \cdot (1 - \delta)^\mu$ e la lunghezza dei pezzi ivi inseriti è maggiore di $\delta \cdot (1 - \delta)^{\mu+1}$. Il fattore $s/(s+1)$ ha la stessa motivazione che per i crates riempiti con pezzi grandi. Si sottolinea che il fattore $s/(s+1)$ per questi ultimi crates è un lower bound valido perché si sta ragionando per l'istanza I$^+$, che si ricorda avere gli stessi pezzi piccoli dell'istanza **I**, ma i pezzi grandi con lunghezza eventualmente aumentata. Inoltre, anche per i crates con pezzi piccoli, vi può essere un certo numero di crates non saturi, uno per ciascun indice $\mu \ge 0$. Per ciascuno di tali valori di μ, l'area del crate non saturo è pari ad $(s+1) \cdot \delta \cdot (1 - \delta)^\mu$, essendone $(s+1)$ l'ampiezza e $\delta \cdot (1 - \delta)^\mu$ la lunghezza.

Bisogna fare un'ulteriore considerazione, non presente nel lavoro di Seiden e Woeginger (2005), sul fattore di riempimento dei crates con pezzi grandi e che permette di

utilizzare A_Z anziché A^+_Z. In particolare, restringendo l'attenzione alle aree dei pezzi grandi, e denotate con A_{big_Z} l'area totale dei pezzi grandi dell'istanza **I**, e con $A^+_{big\_Z}$ l'area totale dei pezzi grandi dell'istanza I^+, il rapporto $A_{big\_Z}/A^+_{big\_Z}$ identifica il fattore di riempimento medio utilizzabile nel valutare l'upper bound per l'area totale A^+_{crates} dei crates dell'istanza I^+.

Si ricorda che l'aumento di lunghezza che permette di passare dai pezzi grandi dell'istanza **I** a quelli grandi dell'istanza I^+ avviene, per ciascun gruppo B_i^+, con un fattore di al più L_i/L_{i+1}. Se si definisce con λ_i tale rapporto, si può mostrare che $\prod_{i=1}^{t} \lambda_i = \frac{L_1}{L_{t+1}} < \frac{1}{\delta}$.

Il rapporto $A_{big\_Z}/A^+_{big\_Z}$ è pari alla somma dei valori λ_i, ciascuno moltiplicato per il peso W_i/W. Dunque un upper bound per tale rapporto è

$$\max_{\substack{\lambda_1,\ldots,\lambda_t: \\ \prod_{i=1}^{t}\lambda_i=\frac{1}{\delta},\ \ \lambda_i\geq 0 \ \ \forall i}} \sum_{i=1}^{t} \frac{W_i}{W}\lambda_i$$

Si può mostrare induttivamente che, al crescere del numero naturale *t*, ed indicati con ω_i i coefficienti W_i/W, tale massimo vale

$$\left(\frac{1}{\delta}-1\right)\max_{i=1,\ldots,t}\{\omega_i\}+\sum_{i=1}^{t}\omega_i \qquad (1.7)$$

Infatti, preordinati i coefficienti in modo che $\omega_1 \geq \omega_2 \geq \ldots \geq \omega_t$, e detto *b* il prodotto fra i valori λ_1 e λ_2, la somma $\omega_1 \cdot b/\lambda_2 + \omega_2 \cdot \lambda_2$ è massima, per qualunque fissato valore di *b*, quando λ_2 vale 1, cui corrisponde una somma $\omega_1 \cdot b + \omega_2$. Infatti, quando λ_2 assume l'altro valore estremo, cioè *b*, mentre λ_1 vale 1, la somma assume valore $\omega_1 + \omega_2 \cdot b$.

Induttivamente, se l'espressione (1.7) vale per un dato valore *t*, allora per *t*+1, detto *b* il valore del prodotto fra i valori $\lambda_1, \ldots, \lambda_{t+1}$, allora per ogni *b* tutti tali valori appartengono all'intervallo [1 , *b*]. Per massimizzare la somma $\omega_{t+1}\cdot\lambda_{t+1} + \omega_1\cdot\lambda_1 + \omega_2\cdot\lambda_1 + \ldots + \omega_t\cdot\lambda_t$ basta considerare che b/λ_{t+1} è il valore conseguentemente imposto, per un dato valore di λ_{t+1}, al prodotto fra i valori $\lambda_1, \ldots, \lambda_t$. Ancora una volta il massimo si ottiene per $\lambda_{t+1} = 1$, e tale massimo vale $\omega_{t+1}\cdot\lambda_{t+1} + (\omega_1\cdot b/\lambda_{t+1} + \omega_2 + \ldots + \omega_t) = \omega_{t+1}\cdot 1 + (\omega_1\cdot b/1 + \omega_2 + \ldots + \omega_t)$, cioè pari a $(b-1)\cdot\omega_1 + \omega_1 + \omega_2 + \ldots + \omega_{t+1}$.

Dunque si ricava $\frac{A^+_{big\_z}}{A_{big\_z}} \leq \left(\frac{1}{\delta}-1\right)\frac{\max_{i=1,\ldots,t}\{W_i\}}{W} + \frac{\sum_{i=1}^{t} W_i}{W} \leq \left(\frac{1}{\delta}-1\right)\frac{1+W/t}{W}+1$.

Affinché tale fattore sia inferiore al fattore $1/(1-\delta)$ valido per i crates dei pezzi piccoli, deve valere la disuguaglianza

$$\frac{1}{W}+\frac{1}{t} \leq \frac{1}{\left(\frac{1}{\delta}-1\right)^2} = \frac{1}{(s-1)^2}$$

Una condizione sufficiente a tal fine è $t \geq (s-1)^2$, ed essa risulta anche necessaria non potendo limitare il rapporto W/t. La posizione iniziale per cui $t = s^2$ assicura la validità di tale proprietà e permette dunque di utilizzare il valore $1/(1-\delta)$ come lower bound per i fattori di riempimento, in termini di lunghezza, di tutti i crates. Per l'area totale dei crates valgono dunque le relazioni:

$$A^+_{crates} \leq \frac{s+1}{s(1-\delta)}A_Z + \sum_{i=1}^{t} L_i(s+1) + \sum_{\mu=0}^{\infty}\delta(1-\delta)^{\mu}(s+1) \leq$$

$$\frac{s+1}{s(1-\delta)}A_Z + t\cdot(s+1) + (s+1)$$

dove la seconda disuguaglianza è ottenuta grazie alle disequazioni $L_i \leq 1$ ed al fatto che $\sum_{\mu=0}^{\infty} \delta(1-\delta)^{\mu} = 1$.

Ora tutte le shelves saturate nella seconda fase hanno una lunghezza utilizzata di almeno (1 – δ), poiché il più grande crate per pezzi piccoli, quello per μ = 0, ha lunghezza pari a δ. Questa considerazione, che permette di legare il valore di APP(I^+) ad A^+_{crates}, sfrutta l'ipotesi per cui la seconda fase aggiunge nuove shelves rispetto alla prima fase provocando dunque la saturazione ad un livello almeno pari ad (1 – δ) di tutte le shelves tranne eventualmente che per l'ultima di esse, la quale comporta un aumento dell'ampiezza del packing di (*s*+1).
Vale dunque la disuguaglianza

$$APP(I^+) \leq \frac{1}{1-\delta} A^+_{crates} + (s+1) \leq \frac{s+1}{s(1-\delta)^2} A_Z + (s+1)\left(\frac{t+1}{1-\delta}+1\right)$$

Infine, dalla relazione $A_Z \leq$ OPT(I), si può ricavare

$$APP(I^+) \leq \frac{s+1}{s(1-\delta)^2} OPT(I) + (s+1)\left(\frac{t+1}{1-\delta}+1\right) =$$

$$\frac{s(s+1)}{(s-1)^2} OPT(I) + (s+1)\frac{s^3+2s-1}{s-1}$$

Anche in questo secondo caso, scegliendo opportunamente *s* affinché $\frac{s(s+1)}{(s-1)^2} \leq 1+\varepsilon$, si ottiene una const(ε) proporzionale ad $(1/\varepsilon)^3$.

In particolare la disuguaglianza equivale ad $\varepsilon \cdot s^2 - (3+2\varepsilon)\cdot s + (1+\varepsilon) \geq 0$ ed è soddisfatta se vale

$$s \geq \frac{3+2\varepsilon+\sqrt{9+8\varepsilon}}{2\varepsilon} \qquad (1.8)$$

Si sottolinea che la condizione (1.6) del primo caso è meno stringente poiché $\frac{3+2\varepsilon+\sqrt{9+8\varepsilon}}{2\varepsilon} \geq \frac{1+\sqrt{1+\varepsilon}}{\varepsilon}$ equivale a $(1+2\varepsilon)+(\sqrt{9+8\varepsilon}-\sqrt{4+4\varepsilon}) \geq 0$ che è sempre verificata per valori positivi di ε.

Per valori di ε sufficientemente piccoli, la relazione (1.8) diventa $s \geq \frac{3}{\varepsilon}+\frac{10}{3}$, avendo utilizzato $\sqrt{9+8\varepsilon} = 3\cdot\sqrt{1+\frac{8}{9}\varepsilon} \approx 3\cdot\left(1+\frac{4}{9}\varepsilon\right) = 3+\frac{4}{3}\varepsilon$. Più in generale una posizione che soddisfa tale disuguaglianza è $s \geq \frac{3}{\varepsilon}+1+\sqrt{\frac{2}{\varepsilon}}$, ottenuta considerando che $\sqrt{9+8\varepsilon} < 3+2\sqrt{2\varepsilon}$.

Una ulteriore semplificazione può portare alla condizione $s \geq \frac{3+\sqrt{2}}{\varepsilon}+1+\sqrt{2}$, che tiene conto sia dei valori di ε minori di 1 che maggiori di 1.

Dovendo *s* essere intera, si conclude con s = $\lceil (9/\varepsilon + 5)/2 \rceil$, avendo usato la maggiorazione $\sqrt{2} \leq 1.5$, sia nell'ottenere $3+\sqrt{2} \leq 4.5$ sia nell'ottenere $1+\sqrt{2} \leq 2.5$. Nel lavoro di Seiden e Woeginger invece si proponeva fin dall'inizio il valore *s* = $\lceil 6/\varepsilon \rceil + 1$, senza

tra l'altro motivare tale scelta. In definitiva, per valori di ε sufficientemente piccoli, si ottiene una const(ε) che aumenta come $\frac{273}{8} \cdot \frac{1}{\varepsilon^3}$.

Si riporta infine la considerazione fatta da Seiden e Woeginger circa la complessità computazionale. Si è prima indicato che il problema LP^+ ha un numero di variabili pari ad $(1/\delta)^t$, cioè s^{s^2}, con *s* proporzionale ad 1/ε, mentre i termini noti coinvolti nelle disequazioni che esprimono i vincoli associati, sono i valori W_i, tutti non maggiori di *n*. Dunque, con algoritmi standard per la programmazione lineare è possibile risolvere LP^+ in un tempo polinomiale rispetto a log(*n*) ed esponenziale rispetto ad 1/ε e ciò è sufficiente per mostrare che l'algoritmo costruttivo sviluppato rappresenta un APTAS. Essi sottolineano però che, invocando il risultato ottenuto da Karmakar e Karp (1982) sul bin Packing mono-dimensionale frazionario, si può raggiungere un tempo polinomiale sia rispetto a log(*n*) che rispetto ad 1/ε, mostrando dunque che l'algoritmo sviluppato è un AFPTAS.

Segue infine un'analisi originale dei risultati descritti. Esplicitando nei due casi la ratio asintotica ed il valore di const(ε) in termini di *s*, δ e *t*. Detto $\eta = \min\left\{1, (1-\delta)\left[1+\frac{1-\delta}{\delta \cdot t}\right]\right\}$, si hanno le seguenti espressioni

$$r_1 = \left(1+\frac{1}{s}\right)\left(1+\frac{1}{\delta \cdot t}\right)$$

$$r_2 = \frac{s+1}{s(1-\delta)^2}\frac{1}{\eta}$$

$$const_1(\varepsilon) = t(s+1)\left(1+\frac{2}{s}\right) + 2\delta \cdot t$$

$$const_2(\varepsilon) = (s+1)\left[\frac{t+1}{1-\delta}\frac{1}{\eta}+1\right] + 2\delta \cdot t$$

avendo aggiunto l'addendo $2\delta \cdot t$ ad entrambe le costanti per tener conto del caso $W/t < 2$.

Ipotizzando di scegliere valori che rendano le ratio r_1 ed r_2 inferiori ad 1 + ε, con ε prossimo a 0, le espressioni possono essere approssimate a

$$\eta \approx \min\left\{1, 1-\delta+\frac{1}{\delta \cdot t}\right\}$$

$$r_1 \approx 1+\frac{1}{s}+\frac{1}{\delta \cdot t}$$

$$r_2 \approx \left[1+\frac{1}{s}+2\delta\right]\frac{1}{\eta}$$

$$const_1(\varepsilon) \approx t \cdot s$$

$$const_2(\varepsilon) \approx \frac{t \cdot s}{\eta}$$

Se $t \geq (1/\delta)^2$, si ha η = 1, $r_1 < r_2$ e $const_1(\varepsilon) = const_2(\varepsilon)$. Allora, per un fissato valore di *s*, poiché *t* non interviene nell'espressione della ratio r_2, δ è scelto pari ad (ε – 1/*s*)/2 e *t* è mantenuto al suo valore minimo possibile, cioè $(1/\delta)^2 = 4/(\varepsilon - 1/s)^2$, da cui si ricava $const(\varepsilon) = 4s^3/(\varepsilon \cdot s - 1)^2$, che assume valore minimo quando $s = 3/\varepsilon$, cui corrisponde $\delta = \varepsilon/3$ e $t = 9/\varepsilon^2$. Ciò mostra che le posizioni fatte inizialmente, secondo cui $\delta = 1/s$ e $t = s^2$, sono quelle ottime, con $const(\varepsilon) = 27/\varepsilon^3$.

In caso contrario, si ha $\eta = 1-\delta+\frac{1}{\delta \cdot t}$ ed $\frac{1}{\eta} \approx 1+\delta-\frac{1}{\delta \cdot t}$, con $const_2(\varepsilon) > const_1(\varepsilon)$ ed $r_2 \approx 1+\frac{1}{s}+3\delta-\frac{1}{\delta \cdot t}$. Si distinguono allora due sottocasi:

se $t \geq (2/3)/\delta^2$, risulta ancora $r_2 \geq r_1$, indipendentemente dal valore di s. Se dunque si fissa s, si può risolvere il problema non lineare

$$\begin{aligned} \min \quad & t(1+\delta)-1/\delta \\ s.t. \quad & 3\delta - \frac{1}{\delta \cdot t} = \varepsilon - \frac{1}{s} \\ & t \geq \frac{2}{3}\frac{1}{\delta^2}, 0 \leq \delta \leq 1, t \geq 1, t \in N \end{aligned}$$

Se infine $t \leq (2/3)/\delta^2$, si ottiene $r_1 \geq r_2$, ancora indipendentemente dal valore di s. Stavolta però, fissato un valore di $s > 1/\varepsilon$, per rendere minima const(ε) = $t \cdot s$ si può risolvere il problema

$$\begin{aligned} \min \quad & t(1+\delta)-1/\delta \\ s.t. \quad & \frac{1}{\delta \cdot t} = \varepsilon - \frac{1}{s} \\ & 1 \leq t \leq \frac{2}{3}\frac{1}{\delta^2}, 0 \leq \delta \leq 1, t \in N \end{aligned}$$

Si sottolinea infine che in tutti e tre i paragrafi, per i valori di k = 2, 3 e 4-staged, non si è mai utilizzata l'ipotesi di considerare una soluzione ottima a ghigliottina. Dunque tutti i risultati sono estendibili al caso di Bin Packing bidimensionale senza taglio a ghigliottina, sebbene tutti i processi costruttivi analizzati producano soluzioni a ghigliottina.

Capitolo 2

Tecnologie di taglio del vetro

Il Capitolo comincia con una classificazione dei vetri piani, basata sulle differenze tecnologiche di realizzazione del taglio.

Segue, nel secondo paragrafo, una descrizione delle tecnologie, mostrando i vincoli strutturali derivanti dalle caratteristiche peculiari del vetro e dando una approfondita idea di come siano costituiti i macchinari per la realizzazione automatica del taglio.

Infine il terzo paragrafo introduce i principali elementi che emergono in una realtà produttiva di trasformazione del vetro piano e che hanno impatto sui modelli del problema del Cutting Stock, con particolare riferimento alle dimensioni dei pezzi da estrarre. In particolare si descrive un modo originale per tenere conto di importanti aspetti pratici nei modelli del problema, per poi concludere con alcune considerazioni sui costi di realizzazione dei tagli.

2.1 Classificazione dei vetri piani

I vetri possono essere classificati secondo l'associazione alle relative tecnologie esistenti per la realizzazione del taglio, che riflettono le caratteristiche fisiche peculiari di ciascuna classe. In tal senso una semplice tassonomia prevede la distinzione di quattro classi di vetro, come rappresentato in Figura 2.1.

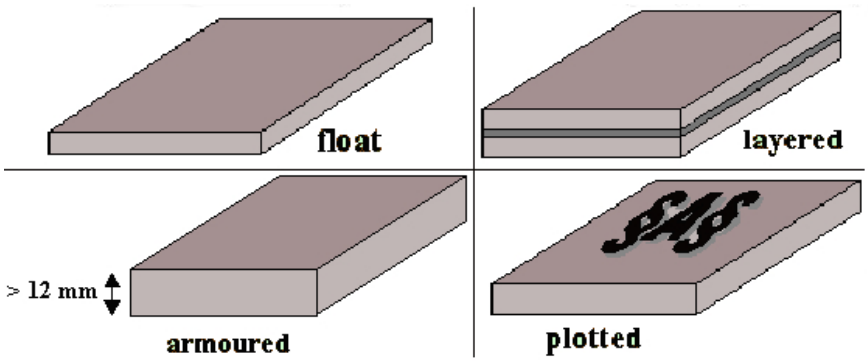


Figura 2.1 Tipi di vetro piano

Il vetro più semplice è quello di tipo *float*, così chiamato per il processo industriale con cui vengono prodotte le lastre grezze, ed in particolare in relazione al suo ultimo step produttivo, che consiste nel far scorrere il vetro fuso in un letto di ghisa posto perfettamente in orizzontale ed in cui il vetro si raffredda. Si dice allora che il vetro, mentre è fuso, fluttua sul letto di ghisa, da cui il termine "float".

Il vetro stratificato (*layered*), o laminato, è nel caso più semplice, ma anche più frequente, costituito da tre strati. I due strati esterni sono di vetro float e fra essi è interposto uno strato di materiale plastico (PVB, o Polinivil Butirrato). Questo vetro, se soggetto a violenti urti, si lesiona ma non si frantuma.

Un vetro, secondo la tassonomia che si sta illustrando, si definisce blindato (*armoured*) quando il suo spessore supera i 12 mm. Nella pratica, già gli spessori standard di 10 mm e 12 mm sono considerabili come intermedi fra la classe float e quella blindata, poiché nel processo di taglio subentrano vincoli di realizzazione legati allo spessore levato.

Infine c'è la classe di vetri detti stampati (*plotted*), generalmente per la presenza di una trama sulla superficie. Per tale classe di vetri è necessario estrarre i pezzi desiderati secondo un orientamento fissato.

2.2 Taglio del vetro

Questo paragrafo comincia con la descrizione del taglio del vetro float, che è il più semplice. Prosegue indicando gli elementi aggiuntivi che devono essere considerati per il taglio rispettivamente di vetri stratificati e di vetri blindati, sottolineandone anche l'impatto sulle tecnologie realizzative.

2.2.1 Taglio di vetri float

L'operazione di taglio (bidimensionale) di una lastra di vetro avviene in due fasi: l'*incisione* e l'*apertura*. L'incisione è effettuata con punte di diamante, la cui durezza conferisce loro la capacità di 'affondare' nel vetro per una certa profondità, piccola rispetto allo spessore totale della lastra. L'incisione è realizzata in genere sulla faccia superiore e genera di fatto una 'ferita' nel vetro, propedeutica alla successiva fase che determina la vera e propria separazione. Questa seconda fase è l'apertura ed è effettuata operando una pressione sulla faccia inferiore del vetro, ed in corrispondenza della linea di incisione. Tale pressione, insieme al peso delle due parti che si trovano ai lati dell'incisione, determina fisicamente una leva e permette di superare la soglia di elasticità del vetro, che dunque cede lungo la linea incisa. Le due fasi sono schematizzate nella seguente Figura 2.2.

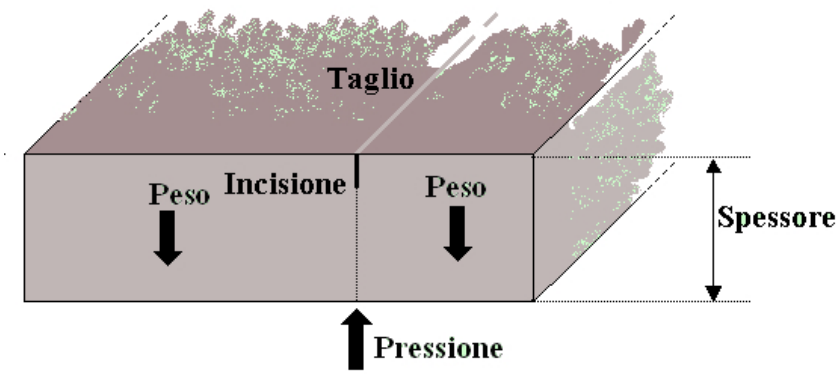


Figura 2.2 Incisione ed apertura

E' importante sottolineare che l'apertura può generare solo delle divisioni complete di una lastra in due parti e cioè lo spacco deve necessariamente avvenire fra due punti perimetrali del vetro. Si dice per questo che i tagli devono essere 'a ghigliottina'. Per le incisioni che non soddisfano tale proprietà, se si fa leva, lo spacco si allunga automaticamente lungo una traiettoria geometricamente casuale e che segue le direzioni microscopiche in cui la struttura fisica del vetro risulta più debole. Tale allungamento indesiderato è in gergo chiamato *finta*. Un esempio è riportato in Figura 2.3. L'esperienza mostra che maggiore è lo spessore del vetro, maggiore è il raggio di curvatura minimo delle finte.

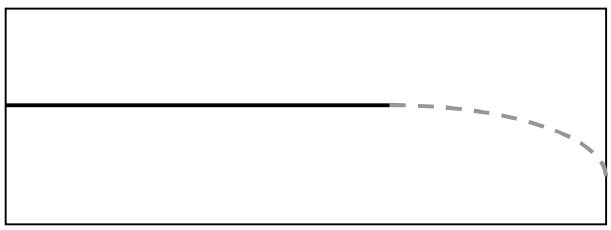


Figura 2.3 Finta per un'incisione non a ghigliottina.

In genere le incisioni vengono preliminarmente realizzate tutte insieme, con opportuna sequenza per minimizzare il tempo di esecuzione dell'operazione, legato al percorso che le punte di diamante devono seguire. Infatti questa operazione avviene quasi sempre in modo automatico, con il supporto di due bracci meccanici, uno per ciascuno dei due assi geometrici. Le aperture vengono effettuate successivamente, una per volta e nella sequenza che rispetta la logica del taglio a ghigliottina.

Un insieme di tagli a ghigliottina, tale da far ottenere i pezzi desiderati a partire da una lastra iniziale, è chiamato *schema di taglio*. In Figura 2.4 è riportato un semplice schema di taglio, con l'indicazione della successione delle aperture, attraverso un indice crescente.

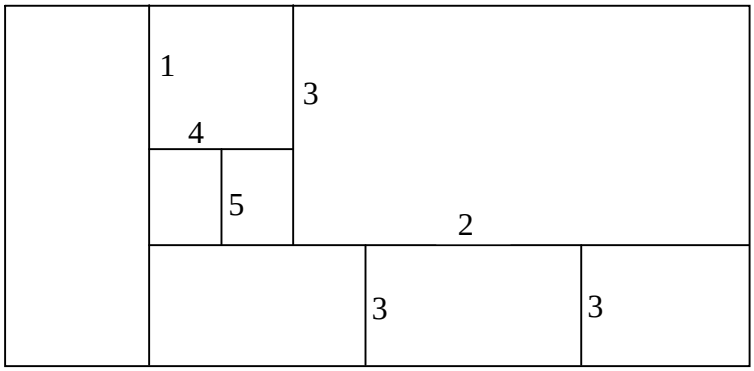


Figura 2.4 Successione delle aperture

Si noti che tagli diversi condividono lo stesso indice (in particolare l'indice 3), poiché tali tagli sono paralleli e le relative aperture possono essere realizzate indipendentemente l'una dall'altra.

Può accadere, per ottenere una riduzione dello *sfrido*, cioè della quantità di materiale scartato, che due incisioni risultino molto vicine, ma una delle due è prioritaria rispetto all'altra. Un esempio è dato dalla coppia di tagli con indici 1 e 3 nella Figura 2.5. La vicinanza fra i due tagli può far si che nella realizzazione dell'apertura del taglio 1, possa aprirsi anche il taglio 3, generando così una finta. Se l'operazione avviene manualmente, è cura dell'operatore porgere attenzione, fermo restando che esiste una probabilità non trascurabile che il fenomeno si verifichi. Si sottolinea pure che se i tagli fossero, oltre che paralleli, di uguale lunghezza, non vi sarebbe alcun pericolo. Ciò vorrebbe dire che i due tagli hanno la stessa priorità, come nel caso dei due tagli con indice 3 che in Figura 2.4 si trovano sotto il taglio di indice 2.

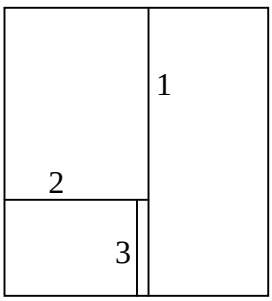


Figura 2.5 Tagli ravvicinati di diversa priorità

Un’altra considerazione riguarda la qualità del taglio. All’atto dell’apertura può avvenire che il taglio non sia perfettamente verticale, ma sia del tipo riportato schematicamente in Figura 2.6.

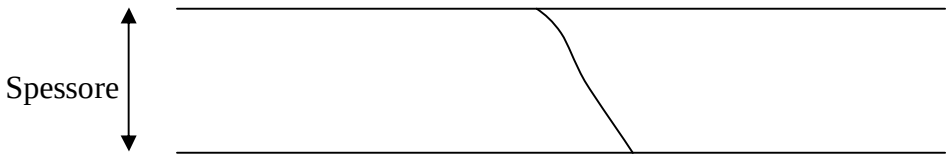


Figura 2.6 Taglio non perpendicolare

Ciò vuol dire che entrambe le parti presentano al bordo una regione da eliminare, la cui ampiezza è comunque dell’ordine dei millimetri. Può esserci l’esigenza di eliminare tale imperfezione e ciò è ottenuto attraverso un’apposita lavorazione di arrotatura che leviga i bordi del pezzo di vetro. Vi sono vari tipi di arrotatura e possono avere altre finalità oltre a quella di correggere l’imperfezione del taglio.

Al taglio del vetro è associato un parametro di *rifilo minimo* (o *minimum trim*), legato ad un vincolo fisico dell’operazione di apertura e corrispondente alla minima distanza che può esserci fra un taglio ed il bordo del pezzo di vetro su cui esso viene realizzato. Ciò perché, al di sotto di certi valori, risulta impossibile esercitare una leva che possa far spaccare il vetro. Tale dimensione cresce con lo spessore del vetro. Ad esempio, nel caso di Figura 2.5, si nota che, effettuati i tagli 1 e 2, l’apertura del taglio 3 può risultare difficoltosa data la vicinanza con il bordo destro della parte sinistra risultante dal taglio 1. La Figura 2.7 schematizza questo discorso.

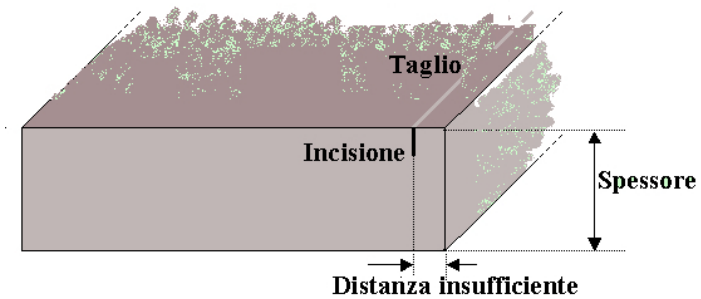


Figura 2.7 Rifilo minimo

Si descrive infine un fenomeno che si verifica nella pratica e soprattutto con gli spessori maggiori. Una volta che l’incisione è stata fatta, è opportuno non aspettare troppo tempo prima di eseguire l’apertura. In gergo si dice che il taglio si “raffredda” e può poi risultare difficile, se non impossibile, spaccare il vetro. Non si può infatti re-incidere il taglio, poiché tale operazione, anche a caldo, ha l’effetto di complicare l’apertura. Questa considerazione implica che, se sono presenti molte incisioni, è necessario operarne solo un opportuno sottoinsieme, aprire i corrispondenti tagli e passare ad un sottoinsieme successivo, iterando tale sequenza fino al completamento dei tagli. Tale strategia è adottata solo raramente nella pratica poiché per spessori crescenti le dimensioni dei pezzi desiderati è generalmente crescente, per cui sono necessari meno tagli per estrarre i pezzi da una lastra. La partizione dell’insieme delle incisioni avviene in generale secondo uno schema ad albero, operando cioè ulteriori incisioni sui pezzi ottenuti da un precedente insieme di incisioni. La divisione in sottoinsiemi comporta perdite di tempo laddove le incisioni debbano essere effettuate attraverso appositi macchinari che operano su un tavolo diverso da quello sul quale

avviene l'apertura, poiché sono necessari gli spostamenti per riportare le sotto-lastre residue sul tavolo di incisione.

2.2.2 Taglio di vetri stratificati

I vetri stratificati sono generalmente costituiti da due strati di vetro float intervallati da uno strato plastico (PVB). Per tale tipo di vetri, le fasi per la realizzazione di un taglio sono tre: *incisione*, *troncaggio* e *strappo*. La fase di strappo determina la rottura dello strato plastico e rappresenta l'apertura vera e propria, realizzata dopo la rottura dei due strati di vetro, qui indicata come troncaggio. In particolare lo strappo deve essere preceduto da una fase di surriscaldamento del plastico. Ciò può essere fatto manualmente bruciando piccole quantità di alcool riversate nella fessura del taglio, oppure in modo meccanizzato con resistenze elettriche che vengono riscaldate ed inserite lungo il taglio, come schematizzato in Figura 2.8. Questo riscaldamento ha impatto sui tempi di realizzazione, in modo più o meno marcato a seconda della temperatura ambientale.

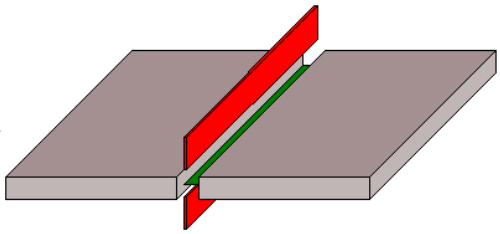
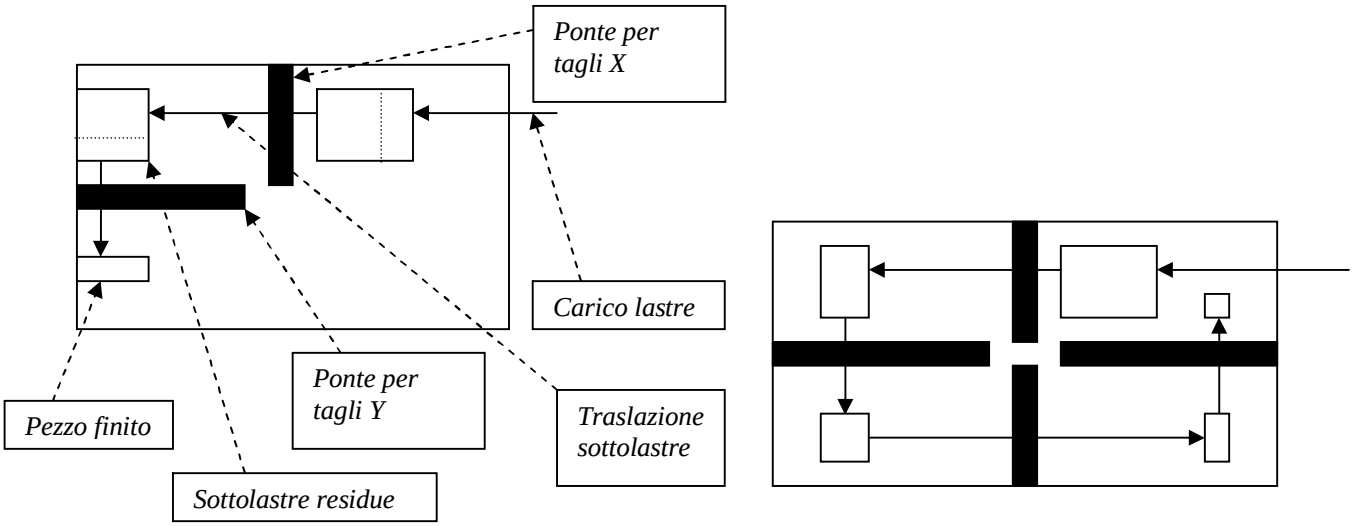


Figura 2.8 Strappo del PVB

Altra differenza importante è che l'incisione deve avvenire su entrambe le facce e cioè su entrambi gli strati di vetro, per cui la meccanizzazione dell'operazione prevede la presenza di due punte di diamante le cui posizioni sono sincronizzate. La complicazione maggiore risulta dal fatto che le incisioni non possono essere effettuate tutte insieme poiché in genere i macchinari che meccanizzano lo strappo esercitano pressioni tali che la presenza di altre incisioni, geometricamente incidenti su quella in fase di apertura, può generare delle finte. Tale vincolo ha una forte conseguenza pratica poiché determina la necessità di un taglio dopo ciascuna incisione.

Nel caso di vetri non stratificati, i macchinari hanno in genere due ponti di taglio mobili e sincronizzati, uno per ciascuno dei due assi del piano. E' infatti possibile operare anche incisioni curvilinee. Nel caso di vetri stratificati, la complessità meccanica derivante dalle considerazioni fatte sopra, implica la necessità di un apposito ponte di taglio per ciascun gruppo di tagli della stessa priorità. Poiché non esistono ancora macchine che consentono di operare in circolo sulle sottolastre residue, vincoli di spazio e vincoli economici fanno optare per la presenza di due o al più tre ponti di taglio. Questo può avere un forte impatto sulla percentuale di sfrido ottenibile, poiché limita fortemente l'insieme delle soluzioni adottabili per lo schema geometrico di taglio. Generalmente le lettere associate alle priorità dei tagli sono, nell'ordine, X, Y, Z, V, W per i primi cinque livelli. In Figura 2.9a è schematizzata la struttura di un macchinario reale, senza ciclo, ed in Figura 2.9b la struttura con ciclo.



a) caso con due ponti di taglio b) caso con ciclo a quattro ponti

Figura 2.9 Macchinario per il taglio di vetri stratificati

Alcuni macchinari prevedono la possibilità di riportare indietro i pezzi, dopo che essi siano stati opportunamente ruotati manualmente da un operatore, così da usare l'ultimo dei ponti per un ulteriore livello di priorità. Si può pensare di generalizzare tale opportunità ed utilizzare l'ultimo ponte per qualunque livello di priorità, supportando l'operazione con un'opportuna logica di "a stack" delle sottolastre residue. Tuttavia tale tipo di operazioni determina problemi di riallineamento dei pezzi poiché c'è l'intervento manuale degli operatori.

In Figura 2.10 si schematizza un esempio di intervento dell'operatore che deve ruotare il pezzo per potervi operare un taglio Z, cioè di terzo livello, con il ponte per tagli Y.

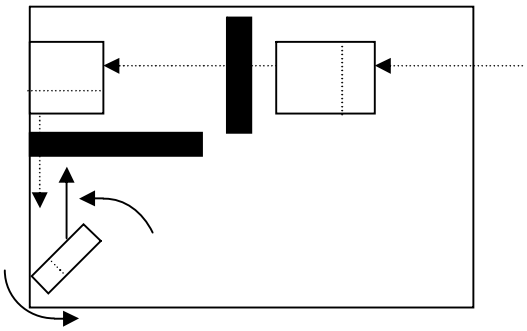


Figura 2.10 Rotazione del pezzo e riallineamento sotto il ponte Y

Esistono anche altri tipi di macchinari, più costosi, che anziché essere disposti in orizzontale, sono disposti in verticale e leggermente inclinati, così da facilitare l'appoggio dei vetri (Figura 2.11). Tali macchinari sono in grado di ruotare automaticamente le sottolastre residue. Attraverso una logica a stack ed anche un solo ponte di taglio, è possibile con tali macchine avere qualunque livello di priorità, pur di non eccedere nell'impegno di spazio dell'insieme delle sottolastre residue presenti in ciascun istante dell'esecuzione. Se ne riporta uno schema esemplificativo in Figura 2.11. Tale tipo di macchine presenta un grosso vantaggio anche in termini di spazio.

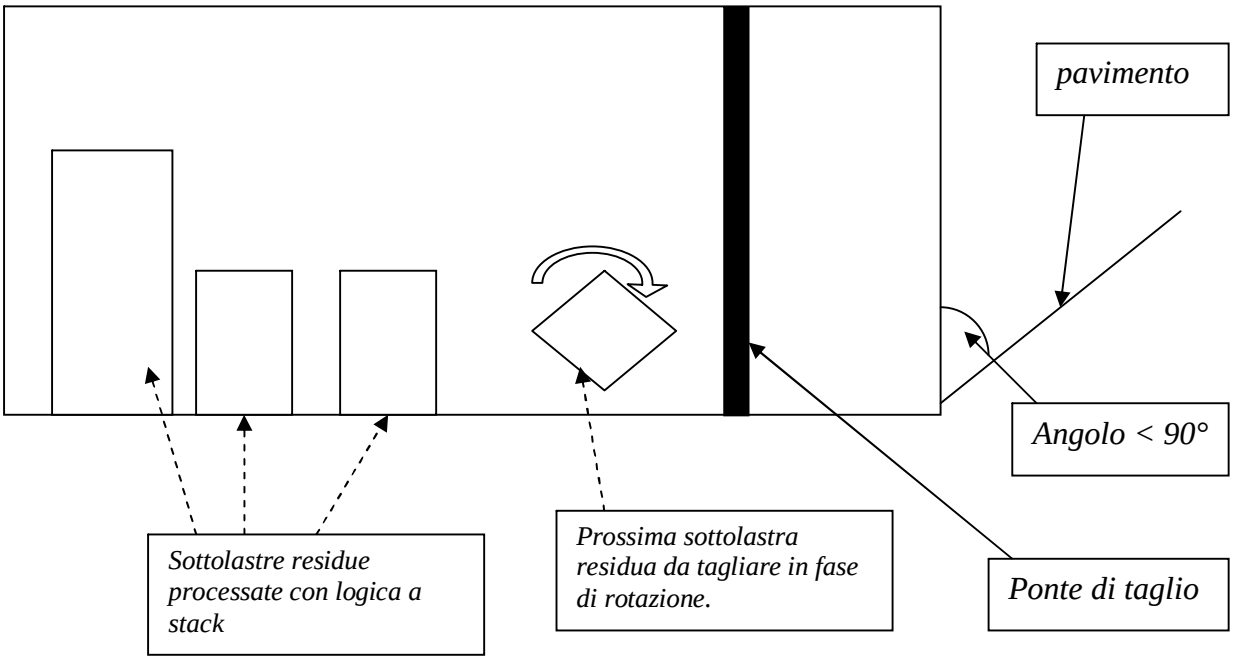


Figura 2.11 Macchina verticale per vetri stratificati

2.2.3 Taglio di vetri blindati

Una discussione particolare merita il tipo di taglio cosiddetto blindato. Quando infatti lo spessore del vetro è elevato, cioè da circa 15 mm in su, il taglio va eseguito con tecniche diverse poiché risulta fisicamente troppo difficoltoso, se non impossibile, aprire il vetro dopo un'incisione. In tal caso il taglio è simile a quello di una lastra di marmo e si utilizza una ruota diamantata, che incide il vetro consumandolo per tutto il suo spessore, come schematizzato in Figura 2.12.

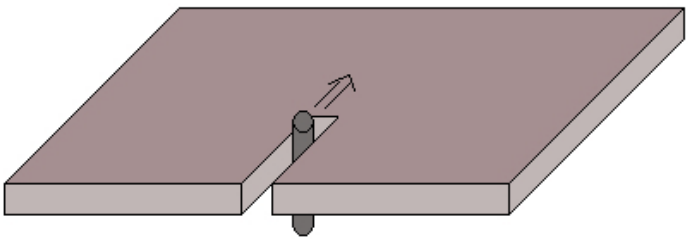


Figura 2.12 Taglio con consumo di materiale

Nella direzione perpendicolare al taglio ed appartenente alla superficie della lastra si genera un consumo di vetro dell'ordine dei quattro millimetri, di cui bisogna tener conto quando si preparano gli schemi di taglio, considerando lastre e pezzi con ciascuna dimensione in modo fittizio aumentata di una quantità pari al consumo.

2.3 Aspetti reali correlati al taglio

La lavorazione di taglio ha degli aspetti dipendenti da vari altri fattori che emergono nella realtà di un ambiente produttivo per la trasformazione del vetro piano. Di seguito viene descritto uno dei più importanti, descrivendo poi anche un modo originale per tenerne conto nei modelli.

2.3.1 Arrotatura, rifilo minimo e tolleranze

La lavorazione di arrotatura consiste nel 'limare' un vetro lungo i bordi, consumando in genere due millimetri lungo tutto il perimetro. E' possibile anche aumentare la quantità di vetro abrasa, non oltre certi limiti, dipendenti dal macchinario utilizzato. La funzione fisica dell'arrotatura è in genere quella di irrobustire il vetro, operando sui suoi bordi.

Essa risulta necessaria quando un vetro deve, ad esempio, essere temprato, poiché altrimenti il danno microscopico generato dall'operazione di taglio può compromettere l'integrità dell'intero vetro quando questo viene portato ad elevata temperatura in un forno di tempra.

Risulta altresì necessaria in presenza di tipologie di vetro con spessore elevato. Infatti, sia che venga essere inciso e successivamente aperto, sia che venga tagliato come un blindato, i bordi risultano non ben definiti e dunque vanno levigati. Nel primo caso la motivazione è il fenomeno già schematizzato in Figura 2.6. Nel caso di taglio blindato, i bordi risultano irregolari per la modalità con cui viene operato il taglio, come schematizzato in Figura 2.13.

Figura 2.13 Bordi irregolari

La necessità dell'arrotatura può anche dipendere da scelte operate nella generazione degli schemi di taglio. In particolare la dipendenza è legata alla possibilità di ottenere (dal taglio) dei vetri leggermente più grandi del dovuto e quindi di utilizzare la lavorazione di arrotatura anche se non esplicitamente richiesta. Questa scelta può, in alcuni casi, determinare un migliore utilizzo delle lastre di partenza, poiché l'allungamento delle dimensioni dei vetri da ottenere può "coprire" scarti che avrebbero avuto dimensione inferiore al rifilo minimo e che dunque non sarebbero stati apribili. Un tale caso è prospettabile anche quando la lavorazione di arrotatura sia già prevista, ma si scelga di aumentare le dimensioni del vetro più del necessario e sempre con il fine di evitare quel tipo di scarti. Tali aumenti sono comunque limitati dalla grandezza del rifilo minimo, al raggiungimento del quale, gli scarti diventano apribili ed il problema viene superato. Quando l'aumento scelto è elevato, è possibile operare più cicli di arrotatura.

Va infine sottolineato che in genere le dimensioni richieste per i vetri hanno di per sé una tolleranza, quasi sempre limitata entro ±1 mm. Ciò implica che a volte il problema del rifilo minimo può essere superato sfruttando la tolleranza propria del vetro, piuttosto che l'arrotatura. In Figura 2.14 è schematizzato il caso in cui, nonostante la tolleranza, l'ausilio dell'arrotatura è necessario a superare il vincolo del rifilo minimo.

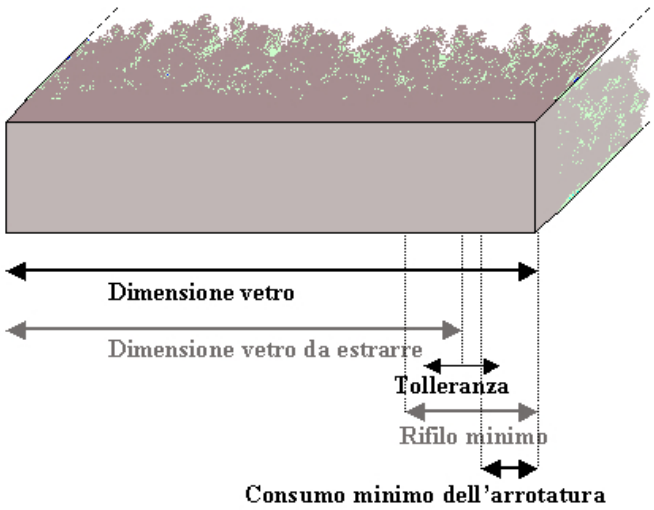


Figura 2.14 Supporto dell'arrotatura

Come si può notare, nonostante la tolleranza, non è né possibile evitare il taglio, né realizzarne uno in accordo con il vincolo del rifilo minimo. La prima evenienza sarebbe possibile se la dimensione massima del pezzo da estrarre fosse almeno pari a quella del vetro, mentre la seconda lo sarebbe se la dimensione minima non superasse la differenza fra la dimensione del vetro ed il rifilo minimo. Se però il consumo minimo dell'arrotatura, come nell'esempio e del resto anche nella pratica, è inferiore al rifilo minimo, è possibile non realizzare il taglio. Nell'esempio di figura si può sfruttare il consumo minimo ed avvalersi della tolleranza sulla dimensione del pezzo.

2.3.2 Estensione dei modelli di Cutting Stock

Al fine di sviluppare metodologie utili nell'ambito della produzione e trasformazione del vetro, sulla base di quanto esposto, è necessario tener conto degli aspetti evidenziati finora in questo Capitolo. E' dunque opportuno introdurre nei modelli classici di Cutting Stock bidimensionale alcuni elementi, con particolare riferimento alla realizzazione dei tagli ed alle dimensioni dei vetri, influenzate dalle tolleranze, dal rifilo minimo e dal legame con l'arrotatura.

Per quanto riguarda il rifilo minimo, la letteratura appare poco presente. Tale aspetto è stato trattato nei lavori di Farley (1983a, 1983c). Per quanto riguarda la tolleranza sulle dimensioni dei pezzi, si indicano i lavori di Scheithauer (1995), che però riguarda problemi mono-dimensionali, e di Vasko, Wolf e Stott (1989). Si sottolinea inoltre un interessante lavoro di Scheithauer (1993), con considerazioni molto utili per tener conto del raffreddamento dei tagli descritto in relazione ai vetri blindati.

In termini di limiti al nesting dei tagli, cioè al livello di priorità, la letteratura appare invece molto ricca. E' infatti elevato il numero di lavori che trattano del Cutting Stock con soli due o tre livelli di taglio. In particolare si parla di *stages*, pensando a differenti fasi del taglio, e ciò si sposa bene al caso del taglio di vetro stratificato, come ampiamente descritto in precedenza. Si citano, ad esempio, i lavori di Valério de Carvalho e Guimarães Rodriguez (1995), Riehme, Scheithauer e Terno (1996), Morabito e Garcia (1998), Hifi (2001), Hifi e Roucairol (2001), Puchinger, Raidl e Koller (2004), Hifi e M'Hallah (2005, 2006), Belov e Scheithauer (2006), Morabito e Belluzzo (2007) e Cui e Zhang (2007).

Per quanto riguarda l'arrotatura, in relazione anche alle tolleranze dimensionali ed al rifilo minimo, si descrive nel seguito una proposta originale di estensione ai modelli di Cutting Stock. In particolare l'elemento chiave consiste nell'esplodere ciascun tipo di pezzi in un insieme di sottotipi, la cui domanda è condivisa. Ciascun sottotipo è associato ad un prefissato numero di cicli di arrotatura. Il numero di sottotipi è limitato dal fatto che, come sottolineato in precedenza, al di sopra del rifilo minimo non vi sono altri vincoli. A ciascun sottotipo può essere associato un profitto diverso, calcolato a partire dal profitto del tipo iniziale ed alterato a seconda del costo aggiuntivo introdotto dal corrispondente numero di cicli di arrotatura. La Figura 2.15 schematizza la divisione in sottotipi secondo le dimensioni in gioco, la tolleranza, l'ampiezza del rifilo minimo ed il range entro cui può cadere, senza l'aggiunta di particolari costi di setup dei relativi macchinari, il consumo dell'arrotatura.

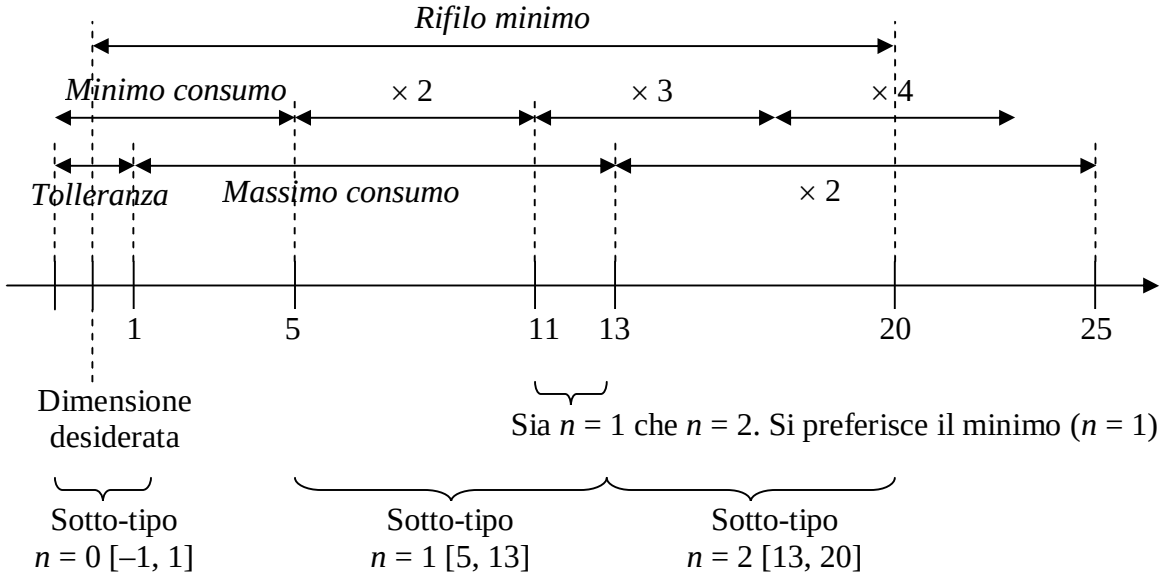


Figura 2.15 Sotto-tipi per numero di cicli di arrotatura

Nell'esempio di figura si suppone di avere una tolleranza di ±1, un rifilo minimo di 20, ed un consumo di arrotatura da un minimo di 6 ad un massimo di 12. Senza cicli di arrotatura, bisogna ottenere una dimensione di ±1 rispetto alla dimensione desiderata. Dunque per il sotto-tipo di indice 0 bisogna rispettare questo range. Con un ciclo di arrotatura, la dimensione del pezzo può essere, rispetto a quella base, da un minimo di 5 in più, in corrispondenza del consumo minimo aggiunto rispetto alla massima tolleranza in negativo, ad un massimo di 13, in corrispondenza del consumo massimo rispetto alla massima tolleranza in positivo. Con due cicli di arrotatura, si potrebbe andare da un minimo di 11 ad un massimo di 25. Tuttavia, da 11 a 13, il range si sovrappone al caso di un solo ciclo di arrotatura.

In casi di sovrapposizione come questo, è preferibile il minimo numero di cicli di arrotatura, che in questo caso corrisponde ad uno. Inoltre, poiché il rifilo minimo vale 20, non c'è bisogno di ulteriori sotto-tipi, ed anzi si può dire che, per il sotto-tipo associato a due cicli, il range va da 13 a 20. Si fa notare che, se il rifilo minimo fosse maggiore, il range virtualmente utile per il caso di tre cicli sarebbe da 17 a 37, ma per sovrapposizione con il caso di due cicli, si restringerebbe all'intervallo da 25 a 37, la cui ampiezza è pari proprio al consumo massimo. Per i sotto-tipi associati ad un numero maggiore di cicli di arrotatura succede la stessa cosa per cui basta considerare a questo punto solo il consumo massimo senza tener conto del consumo minimo. Questo fatto è vero in generale, nel senso che, qualunque siano le dimensioni in gioco, esiste un numero di cicli di arrotatura a partire dal

quale l'intervallo associato è adiacente a quello relativo al numero immediatamente precedente e di ampiezza pari al consumo massimo.

Come approfondimento c'è da dire che il rifilo minimo non è in realtà costante, nel senso che esso varia al variare della lunghezza del relativo taglio. Infatti per tagli piccoli può essere meno difficoltoso aprire tagli che siano abbastanza vicini ai bordi del vetro.

Un modello, che rispecchia bene la realtà, per l'andamento del rifilo minimo al variare della lunghezza del taglio, è riportato in Figura 2.16 e si tratta di un andamento a soglia, con valore massimo al di sotto di una certa lunghezza, un valore minimo al di sopra di una lunghezza maggiore, ed un andamento linearmente decrescente nella zona intermedia.

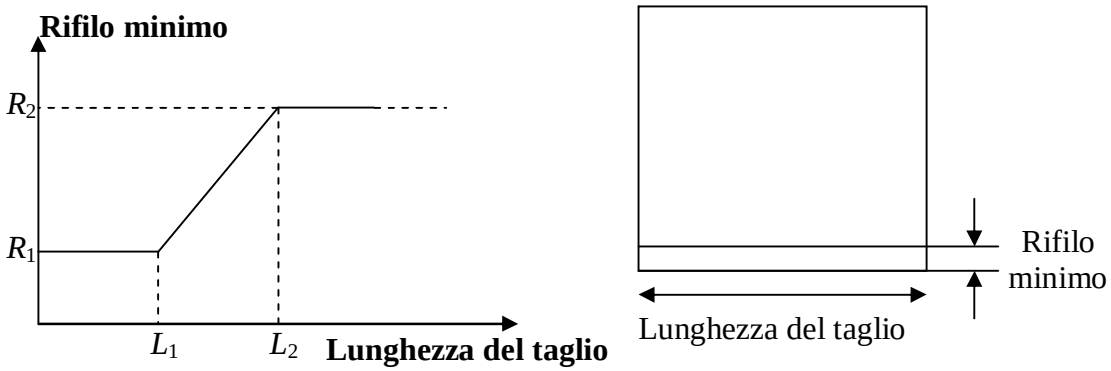


Figura 2.16 Rifilo minimo al variare della lunghezza del taglio

2.3.3 Costi di realizzazione dei tagli

Un ultimo interessante aspetto pratico qui riportato, con possibili ricadute sul modello del problema di Cutting Stock, riguarda il costo associato alla realizzazione dei tagli. Si fa particolare riferimento ai seguenti aspetti, in particolare per i vetri float. Il primo è legato alla lunghezza dei tagli. Infatti tagli più lunghi, soprattutto se si realizzano le aperture manualmente, necessitano di tempi di esecuzione maggiori. Infatti la pressione lungo l'incisione (sulla superficie opposta ad essa) avviene attraverso l'innalzamento, comandato da pedale, di barre di legno presenti sul tavolo di lavoro. Ciò vuol dire che bisogna posizionare la lastra in maniera tale che vi sia corrispondenza fra una barra e l'incisione. Sebbene la movimentazione delle lastre è coadiuvata da soffi di aria che permettono, di fatto, alla lastra di alleggerirsi, è comunque vero che per tagli lunghi questa operazione non è semplice. Un altro motivo che allunga i tempi di esecuzione è che, per tagli lunghi, vi è una probabilità non nulla che possa sorgere una finta, nel senso che il taglio può prendere una direzione diversa da quella dell'incisione, a causa dell'elevato peso delle due parti in cui bisogna scindere il vetro ed a causa di eventuali disomogeneità nella struttura microscopica del vetro. Di conseguenza è necessario porre più attenzione nell'eseguire l'apertura.

Un altro aspetto che ha effetti sui tempi di realizzazione dei tagli è relativa all'esecuzione, soprattutto se automatica, dell'insieme di incisioni. Come accennato in precedenza, è possibile scegliere diversi "percorsi" perché siano realizzati dai bracci meccanici che supportano il movimento delle punte diamantate. In linea di massima si tratta di un problema di ottimizzazione che ricalca il problema del commesso viaggiatore, con alcune particolarità.

La prima consiste nel fatto che, se si modella il problema con una struttura a grafo in cui i nodi rappresentano gli inizi e le fini delle incisioni, con l'aggiunta del punto di fermo dei bracci, da cui bisogna partire ed a cui bisogna tornare, allora i rami corrispondenti alle

incisioni debbono necessariamente far parte del circuito hamiltoniano scelto, dato che le relative incisioni devono necessariamente essere realizzate.

Un'altra caratteristica prescinde dai metodi risolutivi di questo tipo di problema e consiste nel fatto che il calcolo delle distanze fra i punti per cui passare non è quella euclidea classica. Infatti i due bracci meccanici associati ai due assi possono muoversi indipendentemente, per cui la distanza fra due punti del piano (x_1, y_1) ed (x_2, y_2) è pari a $\max\{|x_2 - x_1|, |y_2 - y_1|\}$.

Inoltre i bracci meccanici hanno delle accelerazioni e delle decelerazioni rispettivamente quando cominciano e quando terminano un segmento. L'andamento della velocità su ciascun asse può essere modellato in modo lineare durante queste fasi, mentre c'è velocità di regime pressoché costante nella fase intermedia di uno spostamento, purché il segmento sia sufficientemente lungo. Infine, quando si comincia (o termina) un taglio, c'è un tempo di abbassamento (o innalzamento) delle punte diamantate. Queste ultime due considerazioni implicano che non è possibile, a rigore, associare un costo di movimentazione costante a ciascun ramo. Si supponga ad esempio di avere tre incisioni, *a*, *b* e *c*, sufficientemente corte, allineate ed adiacenti. Se, nel circuito scelto, i rami corrispondenti ad *a*, *b* e *c* si trovano l'uno dopo l'altro, il passaggio per *c* risente del vantaggio di poter sfruttare una velocità dei bracci di regime, poiché l'accelerazione è avvenuta già durante il passaggio per i rami associati ad *a* e *b*, mentre se i rami associati a *b* ed a *c* sono consecutivi, ma il ramo associato ad *a* non precede quello associato a *b*, allora il ramo associato a *c* viene cominciato con una velocità non di regime, ma inferiore.

Si sottolinea che l'ottimizzazione dei costi che sorge da queste considerazioni ha un'importanza minore rispetto ad altre voci di costo, come quella legata alla materia prima. Ciò permette, in generale, di poter ottimizzare il percorso a valle della risoluzione del problema di Cutting Stock, cioè senza tenerne conto durante la costruzione degli schemi di taglio. Inoltre la soluzione può essere sub-ottima, utilizzando modelli semplificati in cui, ad esempio, in presenza di incisioni corte, allineate ed adiacenti, si considerano macro-rami sufficientemente lunghi, generati dal loro accorpamento, come se vi fosse un'unica incisione che le comprende tutte.

L'ultima considerazione riguarda il fatto che, in termini di costo di realizzazione dei tagli, in presenza di vetri con spessore elevato, bisogna tener conto delle movimentazioni delle lastre residue di cui si è parlato in relazione al raffreddamento dei tagli.

PARTE II

Un nuovo approccio algoritmico e nuovi upper bounds

Capitolo 3

Un nuovo approccio algoritmico per il Cutting Stock bidimensionale

I problemi di Cutting Stock vengono ampiamente studiati in letteratura, per l'elevato numero di applicazioni, dall'informatica all'ingegneria industriale, dalla logistica ai processi produttivi. Alcuni esempi sono dati dalla produzione industriale di vetro, carta, legno e metallo. Altre applicazioni sono relative alla schedulazione di attività, anche in sistemi multiprocessori (dove una dimensione rappresenta la risorsa a disposizione e l'altra rappresenta il tempo).

Due lavori introduttivi sull'argomento sono stati scritti da Kantorovich (1939) e da Brooks, Smith, Stone e Tutte (1940). Utili classificazioni si trovano in Fayard, Hifi e Zissimopoulos (1998) ed in Wäscher, Hauβner e Schumann (2007). Per l'importanza del problema, molte riviste hanno pubblicato appositi numeri speciali (Wang e Wäscher, 2002; Oliveira e Wäscher, 2007).

In questo Capitolo, viene considerato il problema del Cutting Stock bidimensionale (Two-Dimensional Cutting Stock problem, o TDC). Esso consiste nel trovare il miglior insieme di pezzi estraibili da un fissato stock (o lastra) rettangolare. In particolare, viene trattata la versione del problema che permette solo tagli a ghigliottina, che cioè cominciano e terminano sui bordi del rettangolo da essi tagliato. Il primo paragrafo dà un definizione del problema e delle sue varianti. Esso dà inoltre una breve descrizione degli approcci risolutivi classici, *top-down* e *bottom-up*, nonché delle *strip heuristics*. Vi sono inoltre introdotti alcuni concetti preliminari. Il secondo paragrafo presenta l'innovativa idea di *boundary disposition* e spiega la struttura di una ricerca euristica ad albero basata su di essa. Il terzo paragrafo spiega una strategia di ricerca per le boundary dispositions. Nel quarto paragrafo sono descritti i relativi upper e lower bounds appositamente utilizzati. Il quinto paragrafo presenta le strategie di eliminazione. Il sesto paragrafo presenta risultati computazionali ottenuti nel contesto reale di una industria di trasformazione del vetro del Sud Italia, nel quale si è sperimentalmente verificato che il nuovo algoritmo produce buone soluzioni in un tempo di calcolo basso e flessibile. Il settimo paragrafo presenta brevemente delle conclusioni.

3.1 Definizione del problema ed approcci risolutivi

Il problema del Cutting Stock bidimensionale (TDC) prevede i seguenti dati di partenza:
- Un rettangolo iniziale *S*, chiamato stock, di materiale con dimensioni intere (L_S, W_S),
- un insieme di *n* quadruple $T = \{t_i : i = 1, \ldots, n\} = \{(l_1, w_1, d_1, \pi_1), \ldots, (l_n, w_n, d_n, \pi_n)\}$ che rappresentano *n* tipi di pezzo rettangolari. Ciascun tipo t_i ha lunghezza l_i, ampiezza w_i, area $l_i \cdot w_i$, domanda d_i e peso (o profitto) π_i, tutti interi. Si suppone, per ciascun tipo t_i, che $l_i \leq L_S$ e $w_i \leq W_S$.

Il problema consiste nel trovare come estrarre dallo stock *S* un insieme di pezzi, che non si sovrappongono e sono posizionati in modo da avere i lati paralleli a quelli dello stock, massimizzando una delle seguenti funzioni obiettivo:
- la somma delle aree dei pezzi estratti (variante non pesata, o *un-weighted*),
- la somma dei pesi dei pezzi estratti (variante pesata o *weighted*).

Nel seguito π_i sarà utilizzato per denotare tanto il profitto quanto l'area, secondo la funzione obiettivo.

Sulla base delle precedenti definizioni, per ciascun tipo t_i, il valore $D_i = \left\lfloor \frac{L_S}{l_i} \right\rfloor \cdot \left\lfloor \frac{W_S}{w_i} \right\rfloor$ è il massimo numero di pezzi di tipo t_i estraibili dallo stock *S*. Quando la domanda d_i di ciascun tipo è non minore della corrispondente D_i, il problema viene classificato come non vincolato, o *un-constrained*, altrimenti come vincolato, o *constrained*.

Inoltre, può esservi la necessità di sezionare lo stock solo attraverso tagli a ghigliottina. In tal caso, il problema è classificato come *guillotine*, e *free* in caso contrario.

Un ulteriore elemento di classificazione è relativo alla possibilità di estrarre i pezzi in entrambi gli orientamenti (problema *free orientation*) oppure no (problema *fixed orientation*).

Infine, per la versione a ghigliottina, i tagli paralleli realizzati sullo stock iniziali sono definiti tagli *first-stage*. I tagli realizzati sui rettangoli così ottenuti sono definiti *second-stage*, perpendicolari a quelli *first-stage*, e così via. Esistono versioni del problema con un limitato numero di *stages*, con applicazioni relative a specifiche tecnologie di realizzazione dei tagli, come descritto nel secondo Capitolo.

Il problema TDC è NP-hard. Infatti, quando $W_s = w_1 = w_2 = \ldots = w_n$, esso diventa equivalente al problema dello zaino mono-dimensionale, che è noto essere NP-hard.

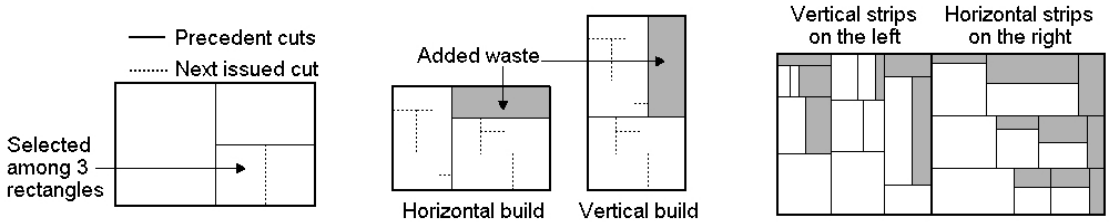
In questo Capitolo viene trattata la versione del problema a ghigliottina. L'approccio proposto è adatto alla vesione *fixed orientation*, sia *weighted* che *un-weighted*. Inoltre, è facilmente adattabile al caso *free orientation*, come spiegato in coda al Capitolo.

In letteratura sono stati proposti differenti approcci per il problema TDC, ad esempio in termini di schemi di programmazione dinamica (Gilmore e Gomory, 1966; Hifi e Zissimopoulos, 1996), di meta-euristiche (Alvarez-Valdés, Parajon e Tamarit, 2002; Puchinger, 2005), di programmazione lineare (Lodi e Monaci, 2003; Belov, 2003). Molti lavori scientifici propongono algoritmi ad hoc, classificabili, rispetto alla versione a ghigliottina, in tre classi: *top-down*, *bottom-up* e *strip heuristics*.

L'approccio *top-down* taglia ricorsivamente lo stock iniziale ed i susseguenti rettangoli in due parti, fino ad ottenere i pezzi finali. Questo approccio segue uno schema di ricerca ad albero, dove il branching consiste nel sezionare un rettangolo selezionato fra quelli ottenuti con i tagli precedenti, come mostrato in Figura 3.1a, generando un *branch* per ciascun possibile taglio e per entrambi i possibili orientamenti. Lavori relativi a tale approccio sono Christofides e Whitlock (1977), Beasley (1985), Morabito, Arenales e Arcaro (1992), Morabito e Arenales (1996), Hifi e Zissimopoulos (1997), Hifi (2004).

L'approccio *bottom-up* segue una strategia del tutto opposta, "fondendo" rettangoli (chiamati *builds* o *meta-rettangoli*) contenenti pezzi. Esso dunque genera rettangoli sempre più grandi fino a riempire al meglio lo stock iniziale. Questa procedura inizia da un insieme di *builds*, corrispondenti ai tipi di pezzi t_i. Vengono utilizzati due modi (*orizzontale* e *verticale*) per fondere una coppia di *builds*, come mostrato in Figura 3.1b, così da soddisfare il taglio a ghigliottina. Lavori relativi a tale approccio sono Wang (1983), Vasko (1989), Oliveira e Ferreira (1990), Viswanathan e Bagchi (1993), Hifi (1997b), Cung, Hifi e Le Cun (2000), G, Seong e Kang (2003).

Gli approcci *strip heuristic* concentrano la loro attenzione su un sottoinsieme di soluzioni, identificato da un predefinito numero di stages, in genere non più di quattro. La Figura 3.1c mostra una soluzione che ha due componenti principali, con orientamento delle strip (cioè delle strisce di pezzi) opposto. Lavori relativi a tale approccio sono Fayard e Zissimopoulos (1995), Fayard, Hifi e Zissimopoulos (1998), Hifi e M'Hallah (2006).



a) *top-down* b) *bottom-up* c) *strip-heuristics*

Figura 3.1 Approcci classici

3.1.1 Alcuni preliminari

Questo paragrafo riassume due concetti chiave, così come noti in letteratura, necessari per supportare la descrizione dell'euristica proposta.

Christofides e Withlock (1977) hanno introdotto le soluzioni normalizzate, definite come soluzioni in cui nessun pezzo può essere traslato verso sinistra o verso il basso, rispettando il taglio a ghigliottina, e senza sovrapporsi ad altri pezzi. E' facile mostrare che è sufficiente cercare solo soluzioni normalizzate, dato che qualunque altra soluzione può essere trasformata in una soluzione normalizzata, attraverso un opportuna traslazione dei suoi pezzi. L'insieme delle possibili coordinate dei tagli in soluzioni normalizzate solo chiamate coordinate normalizzate.

Gilmore e Gomory (1966) hanno introdotto la cosiddetta *knapsack functions*, che fornisce il valore della soluzione ottima per problemi con domanda illimitata. Nel seguito si denota con $f(l, w)$ il valore della knapsack function associata ad un generico rettangolo con dimensioni (l, w). Questo valore può essere usato come upper bound in problemi constrained. Per un rettangolo con dimensioni (l, w), sia $f_0(l, w)$ il valore della migliore soluzione contenente un solo pezzo. Dunque $f(l, w)$ è calcolato attraverso la seguente relazione ricorsiva, come espressa in Hifi (1997a):

$$f(l,w)=\begin{cases} 0 & \text{if} \quad l=0 \quad or \quad w=0 \\ \max\left\{ f_0(l,w),\ \max\limits_{\substack{x_1,x_2:\\ x_1+x_2\le l\\ 0<x_1\le x_2}}\left\{f(x_1,w)+f(x_2,w)\right\},\ \max\limits_{\substack{y_1,y_2:\\ y_1+y_2\le w\\ 0<y_1\le y_2}}\left\{f(l,y_1)+f(l,y_2)\right\}\right\} & else \end{cases} \qquad (3.1)$$

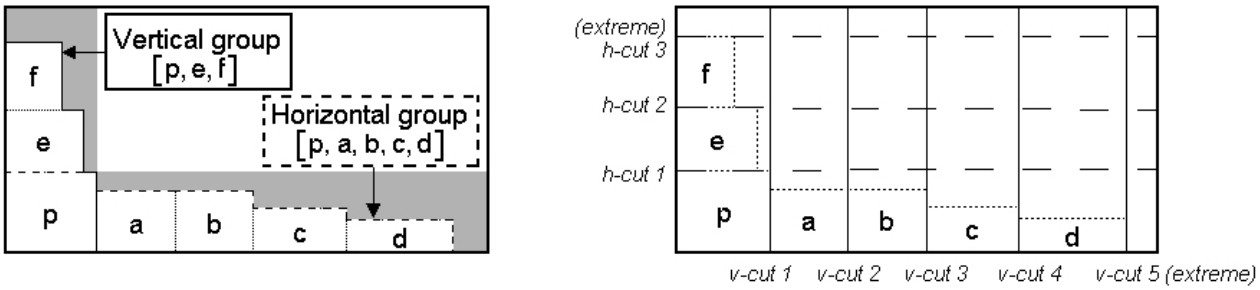
$$\text{where} \quad 0\le l\le L_S \quad , \quad 0\le w\le W_S$$

3.2 L'approccio Boundary Disposition

La strategia top-down non segue alcun criterio per la scelta del taglio da realizzare dopo quelli già aggiunti allo schema di taglio. Quindi gli algoritmi branch and bound con ricerca in profondità basati su questo approccio rischiano di esplorare rami con soluzioni scadenti. L'approccio bottom-up rischia invece di generare troppi builds inutili durante gli step iniziali, senza tener conto delle dimensioni dello stock. A causa di questi potenziali difetti, le due strategie non danno la garanzia di raggiungere buone soluzioni in tempi brevi, se le dimensioni del problema sono elevate. Potrebbe essere possibile utilizzare strategie strip heuristic, le quali hanno però dei limiti sul massimo numero di stage. Per superare tali limitazioni, si propone un nuovo approccio, descritto nel seguito.

3.2.1 L'idea del nuovo approccio

L'idea di fondo consiste nel riempire subito e bene i lati inferiore e sinistro dello stock. Questo può essere fatto scegliendo un pezzo da porre nell'angolo in basso a sinistra, chiamato *corner piece* e denotato con *p* nella Figura 3.2a, e costruendo due *gruppi* (*orizzontale* e *verticale*) di pezzi, che condividono il corner piece e riempiono al meglio la regione a forma di elle evidenziata nella Figura 3.2a. L'unione di questi due gruppi è chiamata *boundary disposition* (disposizione di frontiera) o semplicemente *disposition*. Il tipo di pezzo corrispondente al corner piece è chiamato *corner type*.



a) boundary disposition b) disposition cuts

Figura 3.2 Boundary dispositions

Una boundary disposition genera due insiemi di tagli, definiti *disposition cuts* (Figura 3.2b). Il gruppo orizzontale di pezzi genera un insieme di tagli verticali. L'*i*-mo taglio verticale, da sinistra a destra, è identificato, rispetto al vertice in basso a sinistra dello stock, dalla sua coordinata orizzontale, data dalla somma delle lunghezze dei pezzi posti alla sua sinistra ed appartenenti al gruppo orizzontale. Il gruppo verticale di pezzi genera un insieme di tagli orizzontali. L'*i*-mo taglio orizzontale, dal basso verso l'alto, è identificato dalla sua coordinata verticale, data dalla somma delle ampiezze dei pezzi posti sotto di esso ed appartenenti al gruppo verticale. Tutti i disposition cuts hanno, per costruzione, coordinate normalizzate.

Si noti che in Figura 3.2a i pezzi del gruppo orizzontale sono ordinati, da sinistra a destra, per ampiezza non crescente e che i pezzi del gruppo verticale sono ordinati, dal basso verso l'alto, per lunghezza non crescente. Questo tipo di ordinamento è imposto al fine di ottenere compatibilità fra i tagli relativi ad un gruppo ed i pezzi relativi all'altro gruppo, come evidenziato in Figura 3.2b. La dimensione così associata ad un gruppo è denominata *sorting*

dimension. L'altra dimensione è invece indicata con il nome di *filling dimension*. Un gruppo, inoltre, è definito

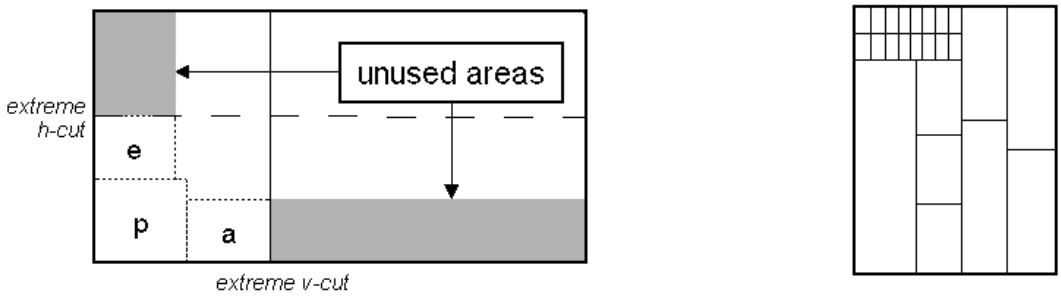
- *omogeneo* quando tutti i suoi pezzi (compreso il corner piece) hanno la stessa sorting dimension,
- *dominato* quando altri pezzi potrebbero essere aggiunti al gruppo senza superare i limiti dello stock,
- *pieno* quando la somma delle filling dimension dei suoi pezzi è pari alla dimensione del rettangolo in cui è costruita la disposizione,
- *perfetto* quando è pieno ed omogeneo,
- *dummy* quando è costituito dal solo corner piece.

Ciascun disposition cut genera due rettangoli che ereditano una boundary disposition parziale. Uno di essi eredita una boundary disposition, che è completa per esso, con lo stesso corner piece (a sinistra per tagli verticali e sotto per tagli orizzontali). L'altro rettangolo eredita solo un gruppo (quello orizzontale per tagli verticali e quello verticale per tagli orizzontali) ed una boundary disposition può essere completata a partire da esso e costruendo l'altro gruppo. Il gruppo ereditato è mantenuto inalterato, anche se è dummy.

Supponendo entrambi i gruppi iniziali non pieni, un'eccezione avviene per due tagli, quello verticale più a destra e quello orizzontale più in alto, che lasciano l'intera boundary disposition solo da un lato. Questi due tagli sono chiamati *extreme disposition cuts*. I pezzi ad essi vicini sono chiamati *extreme disposition pieces*.

Scegliere una boundary disposition corrisponde a scegliere un sottoinsieme di soluzioni. Una boundary disposition può contenere pochi pezzi (a limite uno solo) e lasciare aree non utilizzate nelle zone in basso a destra ed in alto a sinistra dello stock (Figura 3.3a). Le aree inutilizzate sono contenute ciascuna nel rettangolo vuoto ottenuto con il relativo extreme disposition cuts. In tali rettangoli vuoti è possibile costruire ex-novo una boundary disposition, e ciò allo scopo di ottenere soluzioni differenti da quelle ottenibili con boundary dispositions che hanno gruppi non dominati. In Figura 3.3b è mostrata una soluzione ottenibile con gruppi dominati.

Nel seguito del Capitolo, per ogni boundary disposition *bd*, si denoterà con $\Pi(bd)$ la somma dei profitti dei pezzi che costituiscono *bd*.



a) boundary disposition con aree inutilizzate b) soluzione con gruppi dominati

Figura 3.3. Aree non utilizzate e gruppi dominati

3.2.2 La struttura dell'euristica proposta

Si presentano nel seguito gli elementi principali della struttura dell'algoritmo. Si tratta di una ricerca euristica ad albero basata sulla definizione di boundary disposition precedentemente data. Si ricordi che si vuole ottenere una soluzione che massimizzi la somma dei profitti dei pezzi estratti.

3.2.2.1 *Nodo radice e branching*

Nel nodo radice dell'albero di ricerca, vengono calcolati un upper bound ed un lower bound, rispetto allo stock *S*. L'upper bound è calcolato utilizzando due tipi di rilassamenti. Il primo è un rilassamento sia mono-dimensionale che continuo. Il secondo tipo di rilassamento considera domande illimitate di pezzi.

Il lower bound è ottenuto con una euristica greedy che costruisce una soluzione fattibile che ha una struttura a strip. I dettagli di questi bound verranno dati nel paragrafo 3.4. Se l'upper ed il lower bounds sono uguali, l'algoritmo termina poiché ciò vuol dire che è stato trovato l'ottimo, altrimenti viene realizzata una fase di partizionamento.

L'insieme di tutte le possibili soluzioni, corrispondente al nodo radice, può essere partizionato rispetto a tutte le possibili boundary dispositions, realizzando così un *level*-1 *branching*, come mostrato in Figura 3.4. Ciascun sottoinsieme di tale partizione contiene le soluzioni consistenti con la boundary disposition associata. In effetti, dal punto di vista computazionale, non è pratico generare tutte le boundary dispositions. Per tale ragione, l'euristica proposta esplora un limitato numero di sottoinsiemi di soluzioni, adottando una strategia di ricerca *depth-first*.

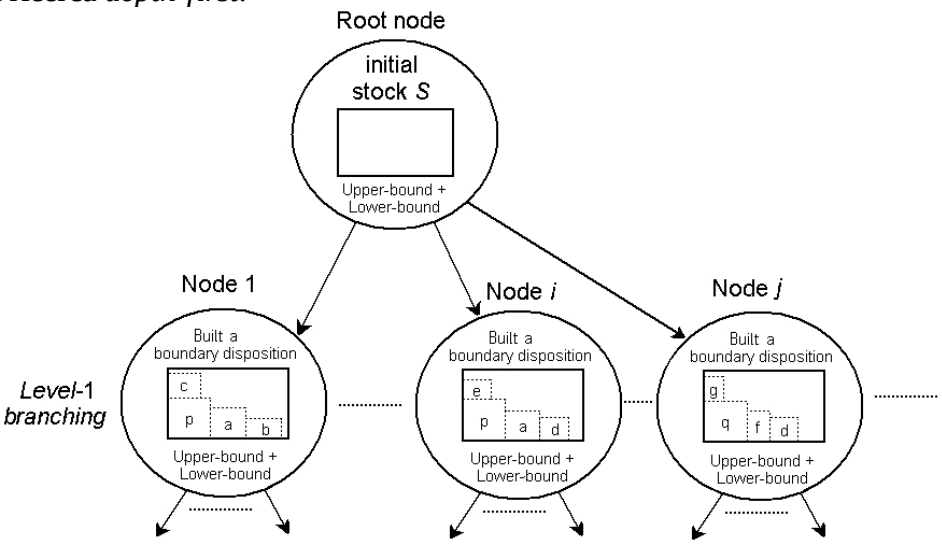


Figura 3.4 Partizionamento nel nodo radice

Per ciascun nodo (e quindi ciascun sottoinsieme di soluzioni), gli stessi due precedenti (upper e lower) bounds sono calcolati, tenendo conto della struttura geometrica della corrente boundary disposition. Questi bounds possono determinare la chiusura del nodo corrente secondo le regole standard degli algoritmi di ricerca ad albero. Infatti, se l'upper bound non è maggiore del valore del miglior corrente lower bound, cioè del valore della migliore soluzione fattibile correntemente nota, il nodo è chiuso, altrimenti viene calcolato il lower bound. Se il lower bound è maggiore del miglior corrente lower bound, esso viene aggiornato. Se l'upper bound ed il lower bound sono uguali, il nodo è chiuso poiché la migliore soluzione del sottoinsieme associato è stata trovata, altrimenti esso deve essere esplorato.

Per fare questo, si sfrutta un nuovo tipo di partizionamento, realizzato rispetto ai disposition cuts associati alla boundary disposition corrente, realizzando così un *level*-2 *branching*. La Figura 3.5 mostra la partizione del nodo 1 rispetto ai cinque disposition cuts, tre verticali e due orizzontali. Ciascun sottoinsieme di questa nuova partizione corrisponde alle soluzioni coerenti con il relativo disposition cut.

Una nuova coppia (upper e lower) di bounds è calcolata per ciascun sottoinsieme ed i test precedentemente descritti vengono applicati. La ricerca fra questi nodi, questa volta, segue una logica *best-bound*. Si noti che la Figura 3.5 mostra entrambi i livelli di branching definiti e gli indici associati ai nodi corrispondono all'ordine in cui essi vengono costruiti.

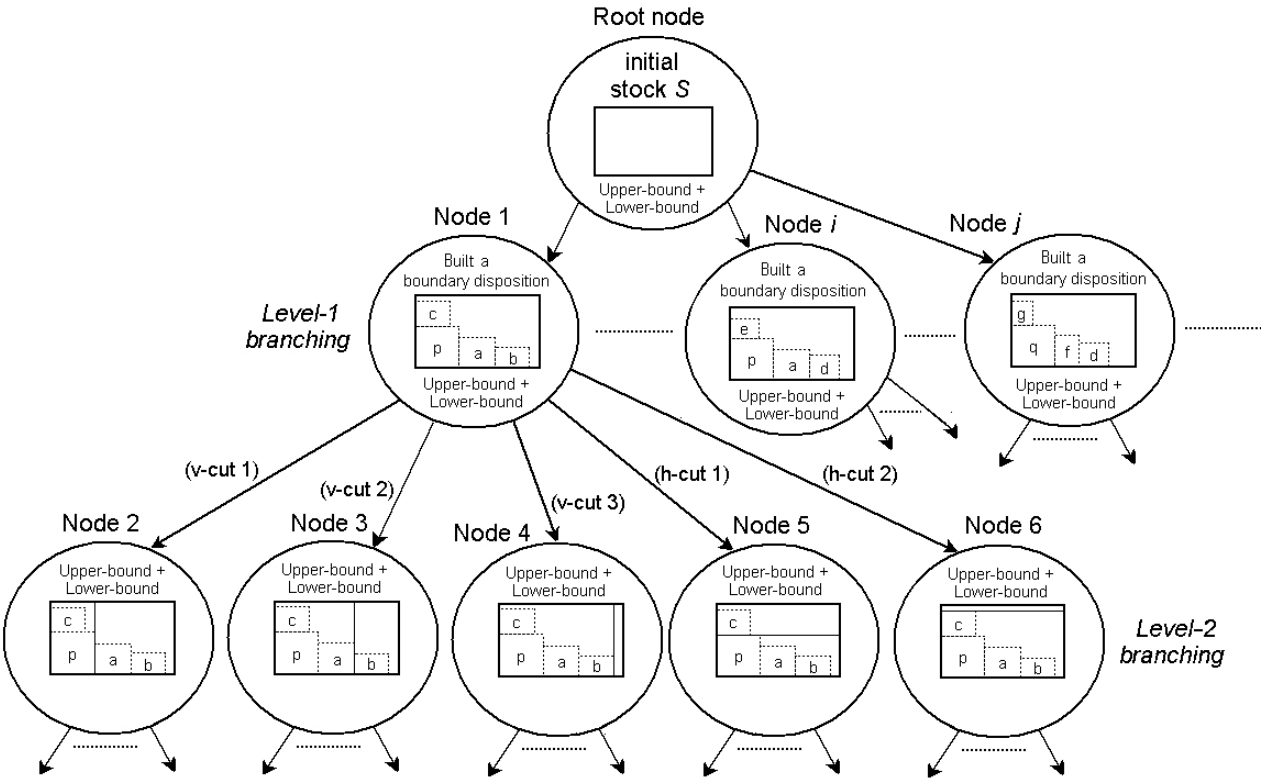


Figura 3.5. Level-2 branching

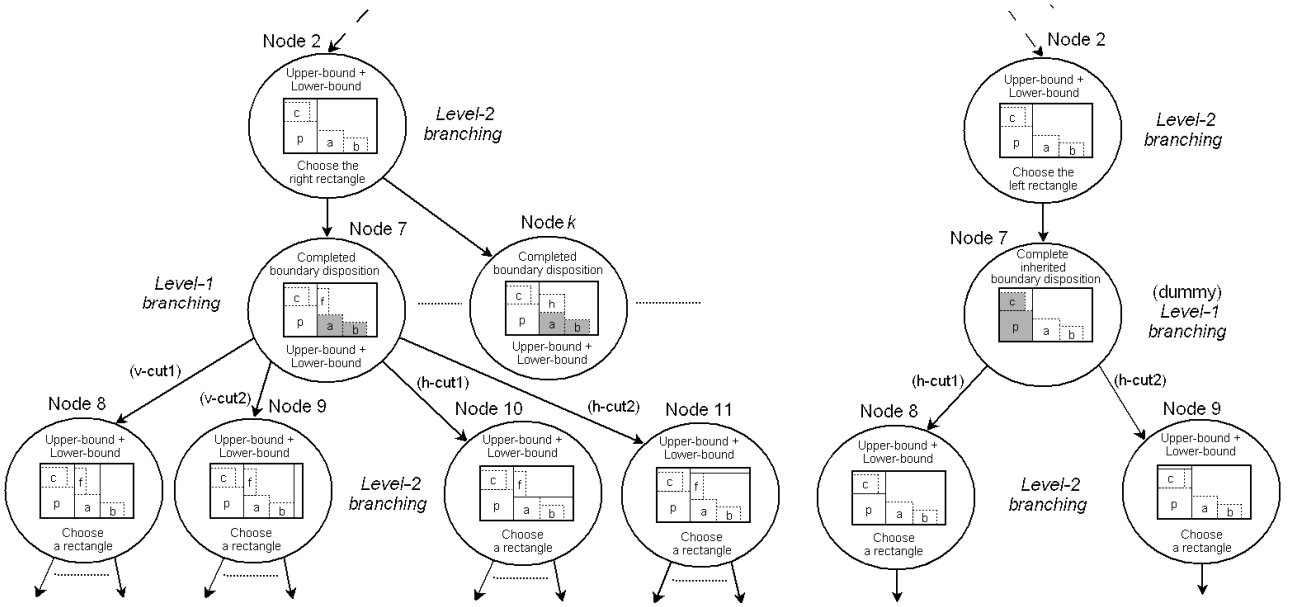
3.2.2.2 *Nodi interni e branching*

Per esplorare uno dei nodi da 2 a 6 della Figura 3.5, l'algoritmo sceglie in modo esclusivo e definitivo uno dei due rettangoli generati dal corrispondente disposition cut. Per tale motivo, essi sono chiamati *rettangoli candidati*. La scelta fra i rettangoli candidati sarà spiegata più avanti nel paragrafo 3.2.2.4.

Si ricordi che ciascun candidato, come spiegato prima, eredita alcuni pezzi, che costituiscono una boundary disposition completa oppure un gruppo da integrare con un nuovo gruppo al fine di ottenere una nuova boundary disposition. Una volta scelto un rettangolo candidato, l'algoritmo realizza un level-1 branching, come fatto nel nodo radice, rispetto a tutte le possibili boundary dispositions coerenti con i pezzi ereditati. Le Figure 3.6a e 3.6b si riferiscono al partizionamento nel nodo 2. La Figura 3.6a fa riferimento al caso in cui si scelga il rettangolo destro, mentre la Figura 3.6b fa riferimento al caso in cui si scelga il rettangolo sinistro.

Se viene scelto il rettangolo destro, è ereditato il gruppo orizzontale e l'algoritmo costruisce un gruppo verticale per ottenere una boundary disposition completa, in ciascuno dei nodi figli generati (nodo 7, …, nodo *k*, …), come mostrato in Figura 3.6a. La partizione del nodo 7, rispetto ai quattro disposition cuts, genera i nodi da 8 a 11. In pratica, come per i primi due livelli di nodi che seguono il nodo radice, tutto l'albero è costruito alternando i due

branching levels, uno (level-1) rispetto alle boundary dispositions e l'altro (level-2) rispetto ai disposition cuts.



a) rettangolo destro scelto nel nodo 2

b) rettangolo sinistro scelto nel nodo 2 (dummy level-1 branching)

Figura 3.6 Partizionamento nei nodi interni

Se invece viene scelto il rettangolo sinistro, è disponibile una boundary disposition completamente ereditata (Figura 3.6b). Quindi viene realizzato un dummy level-1 branching, che genera solo il nodo 7 di Figura 3.6b, dato che l'unica boundary disposition coerente con i pezzi ereditati è proprio quella ereditata. In questo caso non viene aggiunto nessun pezzo, per cui i bounds non vengono ricalcolati, dato che avrebbero lo stesso valore di quelli associati al nodo 2. La partizione del nodo 7, rispetto ai due possibili disposition cuts, generano i nodi 8 e 9 di Figura 3.6b.

Si sottolinea infine che il rettangolo candidato non scelto resta fra quelli candidati per cui verrà scelto in qualche nodo di livello più profondo.

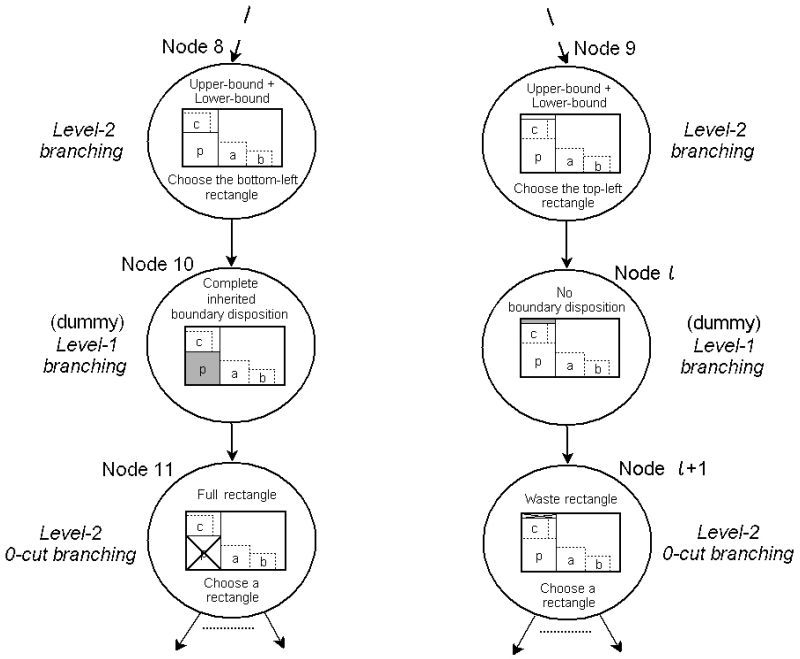
3.2.2.3 *Due casi speciali di branching*

In entrambi i nodi 8 e 9 di Figura 3.6b, risultano generati tre rettangoli candidati. Quando uno di questi nodi deve essere esplorato, uno dei tre rettangoli deve essere scelto, in accordo con quanto detto in precedenza. Si supponga che, per il nodo 8, sia scelto quello in basso a sinistra, contenente il pezzo *p*.

Come per il nodo 2, viene ereditata una boundary disposition completa e solo un nodo viene generato (il nodo 10 in Figura 3.7a). Inoltre, nel nodo 10, il pezzo *p* coincide con il rettangolo scelto e non esiste alcun disposition cut. In questo primo caso speciale, viene realizzato un particolare level-2 branching, che genera il nodo 11. Questo level-2 branching consiste nell'associare il rettangolo al pezzo *p*. E' l'analogo dello 0-cut branching utilizzato nello schema classico del top-down, e per questo qui lo si denota con lo stesso nome. Si noti che, con la nuova strategia, l'associazione con il pezzo è automaticamente ottenuta.

Nell'esplorazione del nodo 9 di Figura 3.6b, se viene scelto il rettangolo candidato vuoto in alto a sinistra (Figura 3.7b), avviene un secondo caso speciale di 0-cut branching. Infatti si suppone che tale rettangolo non possa contenere alcun altro pezzo per cui nessuna boundary disposition può essere associata al nodo *l* e lo 0-cut branching consiste nel fissare tale rettangolo come uno scarto.

Si noti che dopo un qualsiasi 0-cut branching, i valori dei due (upper e lower) bounds restano inalterati, come nel caso di una boundary disposition completamente ereditata. Inoltre, dopo uno 0-cut branching, il rettangolo scelto è eliminato dall'insieme dei candidati. Quindi, nell'esempio di Figura 3.7a, restano solo due rettangoli candidati per il successivo level-1 branching da realizzare nel nodo 11. La stessa cosa succede nel nodo *l* + 1 della Figura 3.7b. Se, dopo uno 0-cut branching, non resta alcun rettangolo candidato, e cioè è stata raggiunta una foglia dell'albero, viene realizzata una operazione di backtracking. La ricerca ad albero termina se nessun altro nodo resta da esplorare.



a) esplorazione del nodo 8 b) esplorazione del nodo 9

Figura 3.7 0-cut branching

3.2.2.4 *Scelta fra i rettangoli candidati*

Come visto in precedenza, per ogni nodo dell'albero c'è un insieme associato di rettangoli candidati. Per gestirli viene utilizzata una semplice strategia. Infatti, viene costruita una lista L di tali rettangoli.

Nel nodo radice la lista L contiene solo lo stock iniziale *S*. Quando un taglio viene realizzato su *S*, esso viene rimosso dalla lista L e rimpiazzato dai due sotto-rettangoli così generati, che vengono ordinati secondo una regola che sarà spiegata nel paragrafo 3.4.3.2.

In ciascun nodo interno in cui si debba realizzare un level-1 branching, viene scelto il primo rettangolo *R* della lista L. Come per lo stock iniziale *S*, se un taglio è realizzato su *R*, esso viene rimosso dalla lista L e rimpiazzato dai due sotto-rettangoli così generati, ponendoli

come primo e secondo nella lista L, secondo un ordine basato sulla stessa regola che vale per i due sotto-rettangoli generati dalla divisione di *S*.

Quando, dopo uno 0-cut branching, il rettangolo *R* scelto è rimosso dall'insieme dei rettangoli candidati, esso viene rimosso pure dalla testa della lista L. Negli step di backtracking, vengono realizzate le operazioni opposte, ponendo cioè il rettangolo *R* in testa alla lista L se esso era stato rimosso o rimpiazzando i primi due rettangoli della lista L con il rettangolo *R* da cui essi erano stati estratti attraverso un taglio.

3.3 Costruzione delle boundary disposition

In tale paragrafo viene spiegato in che modo ed in quale sequenza l'algoritmo genera le boundary dispositions. Si ripete che un level-1 branching esaustivo, rispetto a tutte le possibili boundary dispositions, potrebbe essere troppo costoso dal punto di vista computazionale. Per tale motivo, viene selezionato un numero limitato di boundary dispositions.

Si ricordi che il primo passo per costruire una boundary disposition consiste nella scelta del suo corner piece, e ciò equivale a scegliere il corrispondente tipo di pezzo, che quindi, è il corner type, come definito nel paragrafo 3.2.1. Nel seguente paragrafo 3.3.1 viene spiegata la scelta del corner type. Nel paragrafo 3.3.2 si descrive una procedura per enumerare tutte le boundary dispositions che hanno lo stesso corner type. Infine, nel paragrafo 3.3.3, viene spiegato come realizzare la selezione fra le boundary dispositions associate a tutti i possibili corner types.

3.3.1 Scelta del corner type

Per scegliere il corner type, è importante calcolare un valore di performance per ciascun tipo t_i candidato per questo ruolo. Per lo stock iniziale S si calcola il valore $UB_c(S, i)$, che rappresenta un upper bound per il valore di funzione obiettivo ottenibile utilizzando un pezzo p di tipo t_i come corner piece. Tale valore è la somma di due componenti, la prima delle quali è π_i, cioè il profitto del tipo t_i che ha dimensioni (l_i, w_i). La seconda componente è $U_S(l_i, w_i)$, upper bound per il valore di profitto totale dell'insieme di pezzi che può essere estratto da $S - p$, cioè dalla parte di S non riempita da p.

L'upper bound $U_S(l_i, w_i)$ è selezionato come minimo fra due alternative. Il primo valore possibile, denotato con $U_S'(l_i, w_i)$, è l'upper bound proposto da Viswanathan e Bagchi (1993) per l'approccio bottom-up. In particolare, sia r un rettangolo con dimensioni (l, w), posto all'angolo in basso a sinistra dello stock S, che ha dimensioni (L_S, W_S). I valori di $U_S'(l, w)$, al variare di l e w, sono calcolati attraverso la seguente relazione ricorsiva, che utilizza i valori della knapsack function $f(l, w)$:

$$U_S'(l,w) = \begin{cases} 0 & \text{if} \quad l = L_S \quad \text{and} \quad w = W_S \\ \max\{h_1(l,w), h_2(l,w)\} & \text{else} \end{cases} \tag{3.2}$$

dove

$$h_1(l,w) = \begin{cases} 0 & \text{if} \quad l = L_S \\ \max\limits_{0<u\le L_S - l} \{U_S'(l+u,w) + f(u,w)\} & \text{else} \end{cases} \tag{3.3}$$

e

$$h_2(l,w) = \begin{cases} 0 & \text{if} \quad w = W_S \\ \max\limits_{0<z\le W_S - w} \{U_S'(l,w+z) + f(l,z)\} & \text{else} \end{cases} \tag{3.4}$$

La seconda alternativa, denotata con $U_S''(l_i, w_i)$, è calcolata assumendo due tipi di rilassamenti. Infatti, per calcolare $U_S''(l_i, w_i)$, sono considerate solo le aree dei pezzi e l'area della superficie $S - p$ (rilassamento mono-dimensionale) ed il numero di pezzi utilizzato per ciascun tipo può essere frazionario (rilassamento continuo).

Il valore $U_S''(l_i, w_i)$ può essere calcolato attraverso la seguente procedura greedy. I tipi t_j vengono ordinati per rapporto non crescente $\pi_j / (l_j \cdot w_j)$ fra profitto ed area di ciascun tipo.

Secondo questo criterio di ordinamento, i pezzi sono caricati fino a riempire $S - p$, se possibile, anche con un numero frazionario di pezzi. Questa procedura carica, per ciascun tipo t_j, il massimo numero di pezzi tenendo conto della domanda d_j del tipo e dell'area residua di $S - p$ non riempita dai pezzi dei tipi precedenti.

Si sottolinea che, per calcolare $U_S''(l_i, w_i)$, non vengono utilizzati tutti i tipi t_j, ma solo quelli che posso essere estratti da $S - p$, cioè tali che almeno una delle due seguenti coppie di relazioni sia valida:

$$l_j \le L_S - l_i,\ w_j \le W_S \qquad \text{oppure} \qquad l_j \le L_S,\ w_j \le W_S - w_i$$

Si noti anche che, per problemi un-weighted, se $J_S(l_i, w_i)$ denota l'insieme di indici di tipi usati, vale la seguente relazione:

$$U_S''(l_i, w_i) = \min\left\{ L_S \cdot W_S - l_i \cdot w_i,\ \sum_{j \in J_S(l_i, w_i)} \left[l_j \cdot w_j \cdot \left(d_j - \delta_{ij} \right) \right] \right\} \tag{3.5}$$

L'utilizzo del simbolo di Kronecker serva a ridurre di uno il numero di pezzi nel caso $j = i$, dato che un pezzo di tipo i è già usato come corner piece.

Inoltre, per un sotto-rettangolo R dello stock S, un analogo upper bound $U_R''(l, w)$ può essere definito, utilizzando, per ciascun tipo t_j, al posto dj, il numero b_j di pezzi disponibili.

Infine, per lo stock S, la seguente espressione è adottata per valutare l'upper bound associato al tipo di pezzi t_i.

$$UB_c(S, i) = \pi_i + \min\{U_S'(l_i, w_i),\ U_S''(l_i, w_i)\} \tag{3.6}$$

Per lo stock iniziale S i tipi (corner) t_i sono ordinati secondo valori non crescenti dei corrispondenti bound $UB_c(S, i)$.

Per un generico rettangolo R con dimensioni (L_R, W_R), sono considerati solo i tipi (corner) che possono essere estratti da R, cioè solo i tipi t_i tali che $l_i \le L_R$ e $w_i \le W_R$. Essi vengono ordinati con un analogo criterio, utilizzando il valore di performance:

$$UB_c(R, i) = \pi_i + \min\{U_R'(l_i, w_i),\ U_R''(l_i, w_i)\} \tag{3.7}$$

dove $U_R'(l_i, w_i) = f(L_R, W_R) - f(l_i, w_i)$. La validità di $U_R'(l_i, w_i)$ come upper bound è facile da mostrare secondo ragionamenti analoghi a quelli più complessi del paragrafo 3.4.2.1.

Se per due tipi vi è lo stesso valore di performance, la procedura utilizza prima il tipo con area maggiore o, a parità, quello la cui dimensione maggiore corrisponde alla dimensione maggiore di R.

3.3.2 Enumerazione delle boundary disposition per un fissato corner type

In questo paragrafo è descritta una strategia di enumerazione delle boundary disposition per un determinato corner type. Fissato un corner type, è possible costruire, in genere, diverse boundary dispositions, cioè, ricordando le definizioni date nel paragrafo 3.2.1, diverse coppie di gruppi (orizzontale e verticale).

Se, per un rettangolo R con dimensioni (L_R, W_R), si fissa un corner type t_c con dimensioni (l_c, w_c), allora $L_R - l_c$ è la dimensione della lunghezza residua di R. Analogamente, $W_R - w_c$ è la dimensione dell'ampiezza residua di R. Il gruppo costruito per primo è quello corrispondente alla minima dimensione residua di R oppure, in caso di uguaglianza, quello orizzontale.

Costruire un gruppo orizzontale (verticale) significa porre un insieme di pezzi a destra del (sopra il) corner piece, con lunghezza (ampiezza) totale, cioè filling dimension totale, non maggiore della lunghezza residua $L_R - l_c$ (ampiezza $W_R - w_c$). Una procedura per enumerare

tutti i possibili gruppi orizzontali o verticali è data in Suliman (2001) e riportata nel seguente Box 3.1.

//Groups building procedure
STEP 1. //Build the first group
$\delta = \Omega - \omega_c$; $\Pi = 0$;
For $i = 1$ **to** n **do**
Begin
$m_i = \min\left\{b_i, \left\lfloor \frac{\delta}{\omega_i} \right\rfloor\right\}$;
$\delta = \delta - m_i \cdot \omega_i$;
$\Pi = \Pi + m_i \cdot \pi_i$
End
STEP 2. //Build the other groups
While $m_1 + m_2 + \ldots + m_n > 0$ **Do**
Begin
STEP 2a. //Build a dominated group
$q = \max\{i : m_i > 0\}$;
$\delta = \delta + \omega_q$; $m_q = m_q - 1$;
$\Pi = \Pi - \pi_q$;
STEP 2b. //Build a not dominated group
For $i = q + 1$ **To** n **Do**
Begin
$m_i = \min\left\{b_i, \left\lfloor \frac{\delta}{\omega_i} \right\rfloor\right\}$;
$\delta = \delta - m_i \cdot \omega_i$;
$\Pi = \Pi + m_i \cdot \pi_i$
End
End

Box 3.1 Enumerazione dei gruppi

Per il gruppo orizzontale, Ω sta per L_R ed ω_c per l_c. Per ogni tipo t_i, ω_i sta per l_i. Per il gruppo verticale, Ω sta per W_R, ω_c per w_c ed ω_i per w_i. Inoltre δ denota la corrente dimensione disponibile di R, m_i il numero di pezzi di tipo t_i utilizzati per costruire il gruppo e Π il profitto totale.

Il parametro b_i rappresenta il numero di pezzi di tipo t_i ancora disponibili. In realtà b_i è settato a zero se la sorting dimension del corrispondente tipo t_i è maggiore di quella del corner piece. Questa posizione è fatta in accordo alla struttura delle boundary dispositions con sorting dimensions non crescenti, come precedentemente descritto.

La procedura di costruzione dei grippi genera il primo gruppo (non dominato) con lo step 1 e gli altri con lo step 2. Lo step 2a genera gruppi dominati e lo step 2b genera gruppi non dominati.

Per ottenere prima gruppi che riempiono meglio il corrispondente lato della regione "ad L", la procedura ordina i tipi per valore non crescente di un parametro associato a ciascun tipo t_i e definito come $\theta_i = [\pi_i / (l_i \cdot w_i)] \cdot (\varphi_i / \omega_i)$, dove φ_i è la sorting dimension, perpendicolare alla filling dimension ω_i. Si noti che $l_i \cdot w_i = \varphi_i \cdot \omega_i$ per cui $\theta_i = \pi_i / \omega_i^2$.

Si supponga di costruire prima il gruppo orizzontale e poi quello verticale. Per costruire tutte le possibili boundary dispositions associate al corner type fissato, bisogna

costruire tutti i possibili gruppi verticali compatibili con ciascun gruppo orizzontale costruito dalla precedente procedura. Per ciascun gruppo orizzontale, il corrispondente insieme di gruppi verticali può essere costruito ancora dalla stessa procedura, utilizzando valori ridotti per i b_i, ottenuti tenendo conto dei pezzi già utilizzati per la costruzione del corrente gruppo orizzontale.

Per ciascun gruppo (orizzontale e verticale) di una boundary disposition, i pezzi devono essere ordinati per sorting dimensions non crescente, così da soddisfare la struttura geometrica delle boundary disposition.

Un insieme di pezzi nel gruppo orizzontale (verticale), escludendo il corner piece, potrebbe avere pezzi che condividono il valore della sorting dimension. In questo caso l'algoritmo genera un differente gruppo per ciascuna possible permutazione di questi pezzi, purché non ridondante (per la presenza di pezzi di uguali dimensioni). La prima permutazione nel caso del gruppo orizzontale (verticale), è quella che pone il pezzo più lungo (ampio), cioè quello con filling dimension maggiore, nella posizione aderente al corner piece ed il pezzo meno lungo (ampio) nella posizione più lontana dal corner piece, cioè come extreme piece.

Si ricordi infine che, quando una boundary disposition parziale ereditata va completata, va costruito solo un gruppo ed il corner type è ereditato.

3.3.3 Enumerazione e selezione delle boundary disposition

La strategia descritta nel precedente paragrafo permette di enumerare tutte le possibili boundary dispositions per ciascun fissato corner type. La procedura globale potrebbe risultare troppo costosa, in termini di tempo computazionale, se ciò fosse fatto per ogni corner type. Quindi è necessaria una strategia per selezionare in modo intelligente le boundary dispositions. L'idea di base è selezionare le boundary dispositions in modo coerente con il criterio di performance stabilito per i tipi nel paragrafo 3.3.1. Nel seguito si spiegano i dettagli della strategia.

Si supponga che i tipi siano indicizzati in base al criterio di performance stabilito tramite i valori $UB_c(R, i)$ per il rettangolo corrente R. Come prima boundary disposition, l'euristica costruisce la prima di quelle associate al corner type t_1, cioè la prima costruita dalla procedura spiegata nel paragrafo 3.3.2 con un pezzo di tipo t_1 come corner piece. Essa è denotata con bd_1. Per una qualsiasi altra boundary disposition bd, sia $\Pi(bd)$ la somma dei profitti dei suoi pezzi, come già definito nel paragrafo 3.2.1.

Sia λ un parametro di regolazione. L'euristica costruisce boundary dispositions associate al corner type t_1 finché, per l'n-ma boundary disposition, denotata con $bd_{n(1)}$, la relazione di disuguaglianza $\Pi(bd_{n(1)}) \geq \lambda \cdot \Pi(bd_1)$ non è più soddisfatta. In tal caso l'euristica sospende la costruzione di boundary dispositions associate al corner type t_1 e comincia a costruire quelle associate al corner type t_2. Il valore adottato nei test sperimentali per il parametro λ è 0.95.

Un criterio generale, basato sulla relazione $\Pi(bd_{n(i)}) \cdot UB_c(R, i) \geq \lambda \cdot \Pi(bd_1) \cdot UB_c(R, 1)$, viene utilizzato per decidere quando passare da un corner type t_i al successivo corner type t_{i+1}, fino ad un prefissato numero totale di boundary dispositions, denotato con n_R. Se dopo aver generato le prime boundary dispositions associate all'ultimo corner type, per le quali la suddetta relazione sia soddisfatta, il numero nR non è stato raggiunto, la procedura riparte dalla prima boundary disposition, associata al corner type t_1, non ancora costruita. In tal caso, nella relazione alla base del criterio, il fattore λ è rimpiazzato da λ^k, dove k aumenta con passo 1 ad ogni ciclo. Il fattore $\Pi(bd_1)$ resta invece inalterato.

L'euristica limita il numero di boundary dispositions generate nel seguente modo. Sia n_{bd} un intero positivo. Sul rettangolo candidato scelto R con dimensioni (L_R, W_R), l'euristica

costruisce non più di $n_R = \left\lceil n_{bd}\left(\frac{L_R \cdot W_R}{L_S \cdot W_S}\right)\right\rceil$ boundary dispositions. In altri termini l'euristica costruisce un numero di boundary dispositions proporzionale alla grandezza di *R*. Diverse strategie possono essere realizzate secondo il modo di gestire il parametro n_{bd}, dato che esso influenza direttamente il numero di boundary dispositions generate.

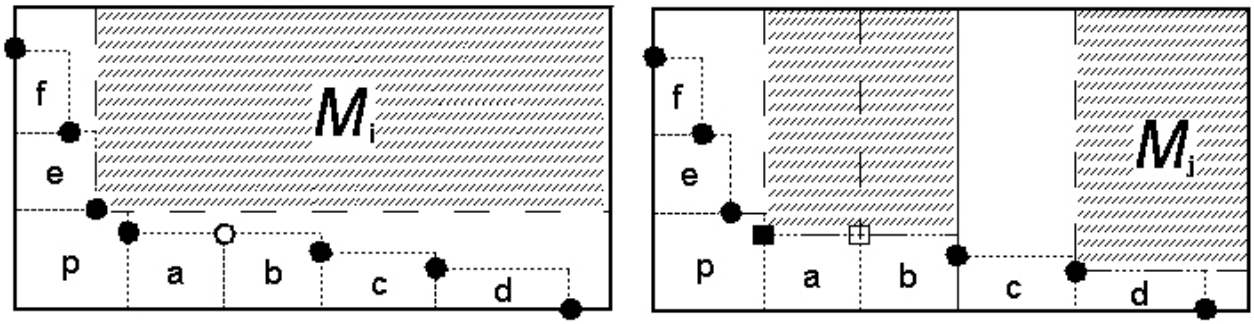
Per esempio, dato un prefissato tempo computazionale, l'euristica di ricerca ad albero può essere ripetuta più volte. Inizialmente il parametro n_{bd} è settato ad 1. Al *k*-mo run il suo valore è settato a *k*. La procedura termina quando un prefissato tempo computazionale è raggiunto. Essa termina pure se, nell'esplorazione di ciascun nodo nel *k*-mo run, nessuna boundary disposition è stata trascurata oppure se l'ottimo è stato trovato, secondo il valore del migliore (minimo) upper bound noto.

3.4 Bounds

In questo paragrafo vengono descritti i bounds utilizzati nell'euristica di ricerca ad albero. Il paragrafo 3.4.1 dà la definizione di rettangolo residuo massimo insieme ad altri concetti correlati. Gli upper e lower bounds sono descritti rispettivamente nei paragrafi 3.4.2 e 3.4.3.

3.4.1 Massimi rettangoli residui

Per una boundary disposition con n pezzi su un rettangolo R con dimensioni (L_R, W_R), possono esistere $n + 1$ *corner points* (Lodi, Martello e Monaci, 2002), evidenziati da cerchi pieni in Figura 3.8a. Essi sono le sole possibili posizioni per i vertici in basso a sinistra di altri pezzi eventualmente aggiunti e traslati il più possibile a sinistra e verso il basso. Per semplicità, sono considerati come corner points tutti gli $n + 1$ punti evidenziati sia da cerchi pieni che vuoti in Figura 3.8a. Indicizzandoli a partire da quello più in alto e terminando con quello più a destra, si denotano con (σ_i, η_i) le coordinate dell'i-mo corner point. Si noti che valgono i seguenti due insiemi di relazioni di ordinamento, vale a dire $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_{n+1}$ e $\eta_1 \geq \eta_2 \geq \ldots \geq \eta_{n+1}$, rispettivamente per le ascisse e per le ordinate.



a) per una boundary disposition completa b) dopo un taglio

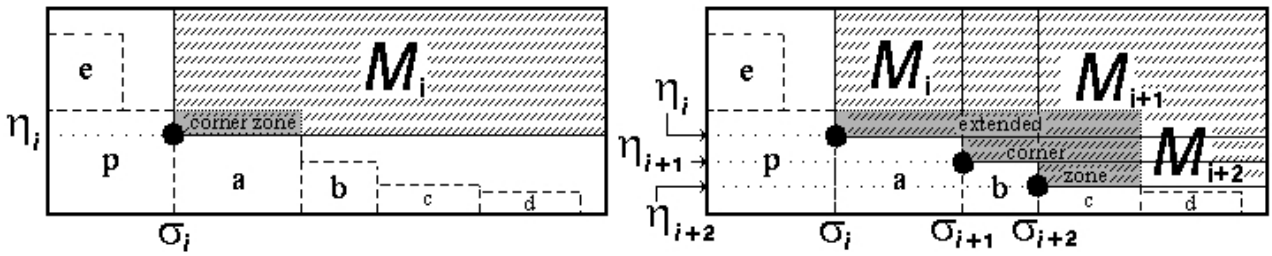
Figura 3.8 Massimi rettangoli residui

Il rettangolo con dimensioni ($L_R - \sigma_i$, $W_R - \eta_i$), che ha cioè l'i-mo corner point come vertice in basso a sinistra e che condivide il vertice in alto a destra con il rettangolo R, è definito *massimo rettangolo residuo* ed è denotato con M_i. Esempi di massimi rettangoli residui per una boundary disposition completa oppure per una parziale ereditata dopo un taglio sono mostrati rispettivamente in Figura 3.8a ed in Figura 3.8b.

Per una data boundary disposition, se $\eta_i = \eta_{i+1}$ allora M_i contiene completamente (domina) M_{i+1}. Ciò può avvenire solo se $i \geq n_v$, dove n_v è il numero di pezzi nel gruppo verticale. Analogamente se, per qualche $i \leq n_v$, vale $\sigma_i = \sigma_{i+1}$ allora M_{i+1} contiene completamente (domina) M_i. Nel seguito si usa questa proprietà per ridurre il numero di rettangoli M_i da considerare. Per esempio in Figura 3.8b c'è un rettangolo residuo massimo, corrispondente al corner point evidenziato da un quadratino vuoto, dominato da quello corrispondente al corner point evidenziato da un quadratino pieno.

Ciascun rettangolo M_i non dominato (Figura 3.9a) contiene, nel suo angolo in basso a sinistra, un'area rettangolare che è definita come *corner zone* e che non è contenuta in nessun altro rettangolo residuo massimo. Quando un rettangolo M_i non può contenere completamente nessun pezzo disponibile, la sua corner zone è inutile e può essere trascurata nel calcolo dei bounds. Si definisce $\mathsf{M}(bd, R)$ l'insieme di indici dei rettangoli M_i non dominati che risultano

utili, cioè che possono contenere completamente almeno un pezzo disponibile e la cui corner zone dunque non è inutile.



a) corner zones b) extended corner zones

Figura 3.9 (Extended) corner zones

Per un sottoinsieme di rettangoli residui massimi $\{M_i, M_{i+1}, \ldots, M_{i+k}\}$ (Figura 3.9b) con indici consecutivi, si può definire una *extended corner zone*, costituita dall'unione non solo delle rispettive corner zones ma anche di tutte le altre aree rettangolari contenute esclusivamente nell'unione di questi rettangoli. In Figura 3.9b sono mostrati tre rettangoli residui massimi (M_i, M_{i+1}, M_{i+2}), corrispondenti ai corner points evidenziati da cerchietti, e la relativa extended corner zone. Se ciascuno di tali rettangoli non è capace di contenere completamente alcun pezzo disponibile, l'euristica trascura non solo le relative corner zones, ma l'intera extended corner zone.

Si possono avere diversi sottoinsiemi di tali rettangoli con indici consecutivi, e con extended corner zones inutili. Definendo come area residua la parte di R non coperta dalla boundary disposition *bd*, si denota con $A(bd, R)$ la parte utile di tale area residua, ottenuta sottraendo da essa le aree delle (extended) corner zones inutili.

3.4.2 Upper bounds

Nel seguito si definiscono due upper bounds. Il primo è relativo al profitto totale, denotato con π_R, dell'insieme di pezzi che può essere estratto dall'area residua utile di un singolo rettangolo candidato R. Tale upper bound è denotato con $UB_1(R)$. Esso permette di calcolare un upper bound, denotato con UB_1, relativo al profitto dell'insieme di pezzi che può essere estratto dall'insieme di aree residue di tutti i rettangoli candidati. Il secondo bound è direttamente calcolato per l'insieme di rettangoli candidati ed è denotato con UB_2. In ciascun nodo dell'albero di ricerca, il minimo fra i due bounds UB_1 ed UB_2 è usato per calcolare un upper bound per il valore della funzione obiettivo ottenibile in quel nodo.

3.4.2.1 *Upper bound* UB_1

Data una boundary disposition *bd* su un singolo rettangolo R, con dimensioni (L_R, W_R), il bound $UB_1(R)$ è ottenuto come minimo fra due differenti upper bounds.

Il primo, denotato con $UB'_1(R)$, è basato su un rilassamento mono-dimensionale e continuo ed è calcolato con la procedura greedy già introdotta nel paragrafo 3.3.1 per il calcolo di $U_S''(l, w)$, utilizzando come dati l'area disponibile ed il massimo numero di pezzi disponibili per ciascun tipo t_i. L'area disponibile qui considerata è l'area residua utile $A(bd, R)$ prima definita. Il massimo numero di pezzi utilizzabili è calcolato, per ciascun tipo t_i, secondo l'area residua massima e secondo le dimensioni dei massimi rettangoli residui di R.

Infatti, per calcolare tale numero, si considerano tre valori. Il primo è b_i, disponibilità corrente per il tipo t_i. Essendo $\left\lfloor \frac{L_R - \sigma_j}{l_i} \right\rfloor . \left\lfloor \frac{W_R - \eta_j}{w_i} \right\rfloor$ il massimo numero di pezzi che può essere estratto dal j-mo rettangolo residuo massimo, allora la somma di questi valori, estesa all'insieme dei rettangoli residui massimi utili, dà un limite superiore al numero di pezzi che può essere estratto dall'area residua utile. Un altro limite superiore per il numero di pezzi è dato dal rapporto $\left\lfloor \frac{A(bd,R)}{l_i \cdot w_i} \right\rfloor$. Quindi, il massimo numero di pezzi utilizzabili di tipo t_i è dato dall'espressione:

$$\min \left\{ b_i , \sum_{j \in \mathsf{M}(bd,R)} \left\lfloor \frac{L_R - \sigma_j}{l_i} \right\rfloor . \left\lfloor \frac{W_R - \eta_j}{w_i} \right\rfloor , \left\lfloor \frac{A(bd,R)}{l_i \cdot w_i} \right\rfloor \right\} \tag{3.8}$$

Il secondo upper bound, denotato con $UB''_1(R)$, è basato su un rilassamento che considera domande di pezzi illimitate ed è calcolato utilizzando la knapsack function descritta nel paragrafo 3.1.1. Per spiegare in che modo, sono necessarie alcune definizioni preliminari.

Sia $R_{c,min}$ la minima parte rettangolare di R contenente l'area residua utile (Figura 3.10). Esso esiste se l'insieme $\mathsf{M}(bd, R)$ prima definito non è vuoto, ovvero se $A(bd, R) > 0$. In tal caso, siano α e β il minimo ed il massimo elemento di $\mathsf{M}(bd, R)$. Quindi $R_{c,min}$ ha $(\sigma_\alpha, \eta_\beta)$ come vertice in basso a sinistra ed ha dimensioni $L_R - \sigma_\alpha$ e $W_R - \eta_\beta$. Sia R_{ov} la parte di $R_{c,min}$ parzialmente sovrapposta da pezzi della boundary disposition. Il numero di tali pezzi è $(\beta - \alpha)$. La parte del k-mo pezzo sovrapposto è denotata con O_k e le sue dimensioni sono denotate con (u_k, z_k). In Figura 3.10 si può vedere R_{ov} costituita da parti di due pezzi, rispettivamente con dimensioni $(l_p - l_e, w_p - w_b)$ ed $(l_a, w_a - w_b)$.

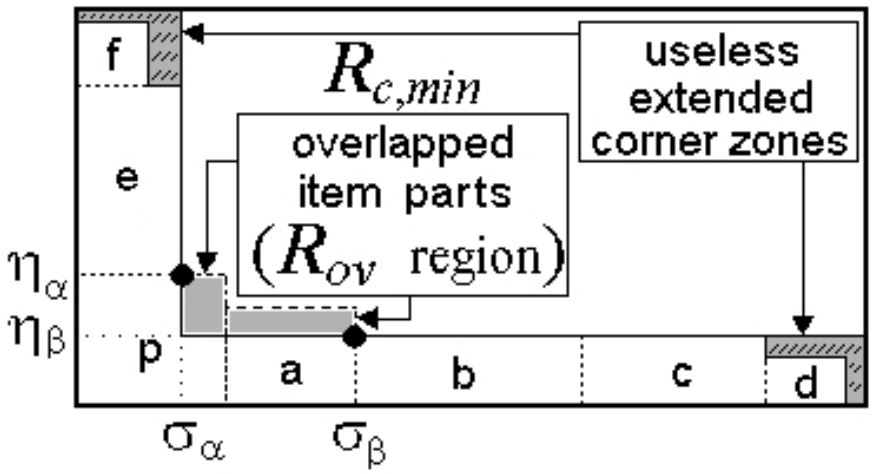


Figura 3.10 Upper bound UB''_1

Sopra si è denotato con π_R il miglior profitto totale per gli insiemi di pezzi estraibili dall'area residua utile di R attraverso un insieme di tagli a ghigliottina che includano i lati dei pezzi di bd. Si supponga ora di separare $R_{c,min}$ da R e si denoti con π'_R il miglior profitto totale per gli insiemi di pezzi estraibili dall'area residua di $R_{c,min}$ attraverso tagli a ghigliottina che includano i lati di ciascuna parte O_k. Si noti che $\pi_R \leq \pi'_R$ dato che tutta l'area residua utile di R è contenuta in $R_{c,min}$ e dato che i lati da includere nell'insieme di tagli a ghigliottina sono ridotti per numero e per lunghezza.

Si denoti ora con f'_R l'analogo di π'_R per il caso di domande di pezzi illimitate. Vale allora la disuguaglianza $\pi'_R \leq f'_R$. Ricordando che i valori della knapsack function $f(l, w)$ sono

calcolati utilizzando domande illimitate, il miglior profitto totale ottenibile per $R_{c,min}$ con le parti O_k, utilizzando domande illimitate, può essere ottenuto come somma dei migliori profitti parziali relativi alla diverse porzioni di $R_{c,min}$, cioè come $f'_R + \sum_{k=1}^{\beta-\alpha} f(u_k, z_k)$. Si noti anche che tale quantità non può essere maggiore del miglior profitto ottenibile, con domande illimitate, a partire dal rettangolo $R_{c,min}$ libero e corrispondente al valore $f(L_R - \sigma_\alpha, W_R - \eta_\beta)$. Vale dunque la seguente relazione:

$$f\left(L_R - \sigma_\alpha, W_R - \eta_\beta\right) \geq f'_R + \sum_{k=1}^{\beta-\alpha} f(u_k, z_k) \tag{3.9}$$

Si può concludere che $\pi_R \leq \pi'_R \leq f'_R \leq f\left(L_R - \sigma_\alpha, W_R - \eta_\beta\right) - \sum_{k=1}^{\beta-\alpha} f(u_k, z_k)$, e dunque adottare, come valore dell'upper bound $UB''_1(R)$, relativo a π_R, il valore dato dall'espressione $f\left(L_R - \sigma_\alpha, W_R - \eta_\beta\right) - \sum_{k=1}^{\beta-\alpha} f(u_k, z_k)$.

Si noti che, se $\mathsf{M}(bd, R)$ contiene un solo elemento, allora il valore di $UB''_1(R)$ diventa pari a $f(L_R - \sigma_\alpha, W_R - \eta_\alpha)$, poiché in tal caso $\alpha = \beta$ ed R_{ov} è vuoto. Si sottolinea anche che, quando $A(bd, R)$ è nulla, $UB''_1(R)$ è settato a zero poiché $R_{c,min}$ è vuoto. Naturalmente in tal caso anche $UB'_1(R)$ vale zero. E' facile verificare che ciascuno dei due bound $UB'_1(R)$ ed $UB''_1(R)$ non domina l'altro. Infine, come precedentemente indicato, si utilizza l'espressione $UB_1(R) = \min\{UB'_1(R), UB''_1(R)\}$.

Nel nodo radice dell'albero ed in quelli di livello immediatamente successivo, nel quale il rettangolo corrente è lo stock S, il bound $UB_1(S)$ è calcolato come appena spiegato. In ciascun nodo interno, per ciascun rettangolo candidato R della corrente lista L, il corrispondente upper bound $UB_1(R)$ è calcolato. L'upper bound UB_1 è ottenuto come somma dei bounds $UB_1(R)$, cioè $UB_1 = \sum_{R \in \mathsf{L}} UB_1(R)$.

3.4.2.2 *Upper bound UB*$_2$

Un secondo upper bound, denotato come UB_2, è calcolato nei nodi interni. Esso si basa sullo stesso rilassamento mono-dimensionale e continuo utilizzato per $UB'_1(R)$ ed è calcolato dalla stessa procedura greedy. Denotando con $bd(R)$ la boundary disposition (completa o parzialmente ereditata) presente in un rettangolo R, l'area disponibile usata per UB_2 è $\sum_{R \in \mathsf{L}} A(bd(R), R)$, cioè la somma delle aree residue utili di tutti i rettangoli candidati R della lista L.

Denotando con $(\sigma_j(R), \eta_j(R))$ le coordinate del j-mo corner point su un rettangolo R, il massimo numero di pezzi utilizzabili, per il tipo t_i, è calcolato come $\min\left\{b_i, \sum_{R \in \mathsf{L}} q(R)\right\}$, dove la somma è estesa a tutti i rettangoli candidati e

$$q(R) = \min\left\{ \sum_{j \in \mathsf{M}(bd(R), R)} \left\lfloor \frac{L_R - \sigma_j(R)}{l_i} \right\rfloor \cdot \left\lfloor \frac{W_R - \eta_j(R)}{w_i} \right\rfloor, \left\lfloor \frac{A(bd(R), R)}{l_i \cdot w_i} \right\rfloor \right\}$$

In particolare, si assume, per ciascun termine, lo stesso significato degli analoghi termini dell'espressione (3.8).

Nel nodo radice, il bound UB_2 è uguale ad $UB'_1(S)$ poichè la lista L contiene solo lo stock S.

3.4.2.3 *Upper bound globale*

Per un nodo dell'albero, sia L' la lista contenente tutti i rettangoli ottenuti dai tagli già realizzati, ma non ancora, a loro volta, tagliati, candidati oppure no. Si ricordi che i rettangoli non candidati sono quelli associati agli step di 0-cut branching realizzati, come descritto in precedenza. Un rettangolo non candidato *R* può essere vuoto o contenere un solo pezzo, che lo riempie completamente, secondo il tipo del corrispondente 0-cut branching.

In ciascun nodo, l'upper bound globale relativo al valore di funzione obiettivo è ottenuto aggiungendo il valore $\min\{UB_1, UB_2\}$ ai profitti dei pezzi utilizzati per costruire le boundary dispositions presenti sui rettangoli associati a quel nodo, cioè utilizzando l'espressione $\left[\min\{UB_1, UB_2\} + \sum_{R \in L'} \Pi(bd(R)) \right]$, dove $\Pi(bd(R))$ denota il profitto totale della boundary disposition $bd(R)$ presente su un rettangolo *R*. Nel caso di rettangolo *R* non candidato, la boundary disposition bd(R) ha un solo pezzo, ad esempio di tipo t_i, per cui $\Pi(bd(R))$ è uguale a π_i.

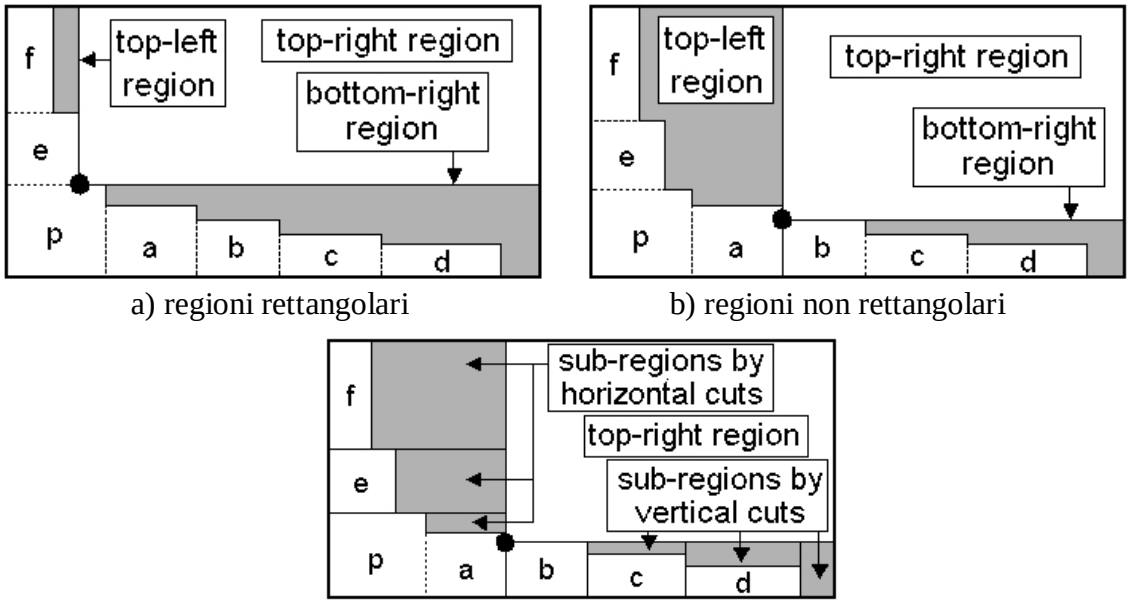
3.4.3 Lower bounds

Nel seguito si spiega come si calcola un lower bound "parziale" per un singolo rettangolo parzialmente riempito da una boundary disposition, sia essa completa oppure parzialmente ereditata. A tal fine, viene utilizzata una strip heuristic. I lower bounds parziali relativi ai rettangoli candidati sono poi combinati per ottenere il lower bound globale relativo al nodo corrente dell'albero di ricerca. Infatti, il lower bound di un nodo è ottenuto aggiungendo la somma dei lower bounds parziali ai profitti dei pezzi fissati tramite gli step di 0-cut branching precedentemente realizzati.

3.4.3.1 *Lower bound parziale per un singolo rettangolo candidato*

Si può osservare che è possibile associare una coppia di tagli perpendicolari a ciascun corner point di un rettangolo contenente una boundary disposition. In Figura 3.11a (Figura 3.11b) si può vedere che per un corner point del gruppo verticale (orizzontale), il primo taglio della coppia è un disposition cut orizzontale (verticale) ed il secondo è il taglio verticale più a sinistra (orizzontale più in basso) sopra il (a destra del) primo taglio.

Ciascuna coppia di tagli divide l'area residua in tre regioni. La regione in alto a destra (*top-right region*) ha sempre una forma rettangolare e può essere riempita con una strip heuristic, cioè generando una combinazione di strips orizzontali e/o verticali. Le altre due regioni (quella in alto a sinistra o *top-left region*, e quella in basso a destra, o *bottom-right region*) possono ciascuna essere rettangolare oppure no. La Figura 3.11a mostra un caso con una regione top-left rettangolare. La Figura 3.11b mostra un caso con entrambe le regioni non rettangolari. Nel secondo caso, utilizzando tagli verticali nella regione bottom-right e tagli orizzontali nella regione top-left, le regioni non rettangolari possono essere divise in sotto-regioni (*sub-regions*) rettangolari, come mostrato in Figura 3.11c.

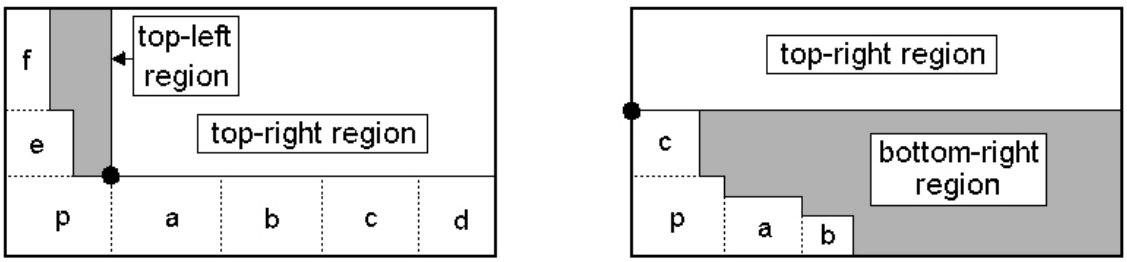


a) regioni rettangolari b) regioni non rettangolari

c) sub-regions

Figura 3.11 (Sub-)regions per lower bounds

Se un gruppo perfetto, cioè pieno ed omogeneo, è presente nella boundary disposition, una delle due regioni associate ad un qualsiasi corner point di quel gruppo è vuota, come nel caso del gruppo orizzontale in Figura 3.12a. Si noti anche che solo un taglio può essere associato al primo (ultimo) corner point, cioè il disposition cut orizzontale (verticale) estremo. Quindi si identificano solo due regioni, come mostrato in Figura 3.12b.



a) caso di gruppo perfetto b) regioni per il primo corner point

Figura 3.12 Casi con una regione vuota

Il lower bounds parziale per la regione top-right region e per ciascuna sub-region sono calcolati utilizzando la strip heuristic proposta da Hifi e Zissimopoulos (1997). All'inizio dell'algoritmo si calcolano le migliori strip per tutti i possibili rettangoli, assumendo che tutti i pezzi siano disponibili. In ciascun nodo dell'albero ed in ciascuna sub-region si utilizzano le strip ottenute da quelle migliori, eliminando i pezzi non più disponibili.

Per calcolare un lower bound parziale per la regione bottom-right o per la regione top-left, la strip heuristic è applicata a ciascuna relativa sub-region, iniziando da quella più vicina al corner piece, cioè da sinistra a destra per la regione bottom-right e dal basso verso l'alto per la regione top-left.

Le tre regioni sono riempite a partire da quella con superficie più piccola. Se due regioni hanno uguale superficie, la regione top-right è riempita come ultima. Fra le altre due regioni, la procedura riempie prima quella con massimo numero di sub-regions o, in caso di ulteriore parità, la regione associata alla dimensione minima del rettangolo corrente, considerando solo la parte residua rispetto a quanto occupato dal corner piece.

Se un rettangolo R è vuoto, cioè non ha alcuna boundary disposition, il lower bound parziale è calcolato come sopra per la regione top-right. Se invece R ha una boundary disposition *bd*, il lower bound parziale è calcolato come somma di due componenti. La prima è il profitto $\Pi(bd)$ associato a *bd*. La seconda è il massimo fra i valori ottenuti, per ciascun corner point, come somma dei lower bounds parziali associati alle tre corrispondenti regioni e calcolati nella sequenza descritta prima.

3.4.3.2 *Lower bound globale*

Il lower bound globale per lo stock S nel nodo radice è calcolato come descritto prima per un rettangolo vuoto. In un nodo interno il lower bound globale è calcolato aggiungendo i profitti dei pezzi fissati dagli step di 0-cut branching, ai lower bounds parziali relativi a tutti i rettangoli candidati e calcolati in sequenza come descritto nel seguito.

Si denoti con $bd(R)$ la boundary disposition (completa o parzialmente ereditata) presente in un rettangolo candidato R della lista L associata al nodo. L'euristica ordina i rettangoli candidati R secondo valori non decrescenti dell'area residua utile $A(bd(R), R)$ o, in casi di parità, secondo cardinalità non decrescente dell'insieme $\mathsf{M}(bd(R), R)$, cioè il numero di massimi rettangoli residui non dominati, come definito nel paragrafo 3.4.1. Se due rettangoli hanno parametri con lo stesso valore, si utilizza l'ordinamento nella lista L, introdotto nel paragrafo 3.2.2.4.

Nel paragrafo 3.2.2.4 si è ritardata la descrizione della regola di ordinamento per due nuovi rettangoli ottenuti da un taglio relativo ad un nodo generato da un level-2 branching. Ora si può indicare che vengono utilizzati i due parametri sopra menzionati, $A(bd(R), R)$ e $|\mathsf{M}(bd(R), R)|$, con la stessa regola di ordinamento, per decidere quale dei due rettangoli debba essere posto come primo nella lista L. Se essi hanno parametri con lo stesso valore, si utilizza la convenzione di porre come primo quello più vicino all'angolo in basso a sinistra dello stock S.

3.5 Strategie di eliminazione

Le strategie di eliminazione hanno lo scopo di evitare che si esplorino insiemi di soluzioni, ridondanti o dominati, nell'albero della ricerca euristica. L'euristica proposta si basa su due tipi di strategie di eliminazione, uno orientato ai disposition cuts e l'altro alle boundary dispositions. Per brevità, si descrivono solo alcune strategie atte ad evitare la generazione di nodi derivanti dall'aggiunta di disposition cuts. Le prime due strategie sono rispettivamente per tagli paralleli e per tagli perpendicolari. La terza strategia permette di considerare un solo disposition cut.

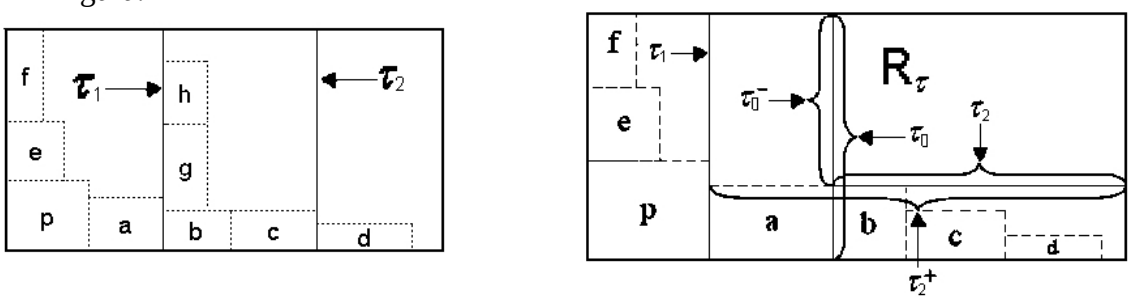
La prima strategia, denotata con *C*1, è simile alla strategia proposta da Christofides e Whitlock (1977). La Figura 3.13a schematizza il caso di tagli verticali. Si può vedere che è possibile aggiungere il taglio τ_1, poi completare sulla destra di τ_1 la boundary disposition ereditata aggiungendo i pezzi *g* ed *h*, ed infine aggiungere il taglio τ_2. Alternatiavmente si può cominciare aggiungendo τ_2, poi il taglio τ_1 ed infine completare la boundary disposition sulla destra di τ_1 con gli stessi pezzi *g* ed *h*. In altre parole, lo stesso posizionamento di pezzi è ottenibile in due differenti modi.

La strategia proposta ha lo scopo di evitare di costruire due volte le stesse soluzioni, evitando tagli verticali sulla destra di tagli verticali, e tagli orizzontali sopra tagli orizzontali. Ciò implica che, nell'esempio di Figura 3.13a, non si può aggiungere τ_2 dopo τ_1 e quindi solo tagli orizzontali possono essere aggiunti a destra di τ_1. Un'eccezione sorge nel caso di extreme disposition cuts, che generano un rettangolo senza alcuna boundary disposition ereditata.

La seconda strategia, denotata con *C*2, consiste nell'evitare coppie di tagli perpendicolati. Per spiegare tale strategia, si fa riferimento alla Figura 3.13b che mostra tre tagli, denotati con τ_0, τ_1 e τ_2. Nel gruppo orizzontale è presente una coppia di pezzi con uguale sorting dimension, cioè *a* e *b*. In questo caso è inutile avere contemporaneamente i tre tagli τ_0, τ_1 e τ_2.

Infatti, una qualunque soluzione che contiene l'insieme dei tre tagli $\{\tau_0, \tau_1, \tau_2\}$ è dominata da una soluzione che ha l'insieme di tagli $\{\tau_1, \tau_2^+\}$, dove τ_2^+ è il taglio τ_2 allungato a sinistra. La motivazione è che, aggiungendo anche il taglio τ_0^- (cioè il taglio τ_0 accorciato nella parte bassa) nel rettangolo top-right vuoto, generato sopra τ_2^+ e denotato con R_τ, si ottiene lo stesso schema parziale di tagli. Ciò vuol dire che la struttura delle soluzioni ottenibili utilizzando τ_1 e τ_2^+ è meno vincolata rispetto al caso in cui si utilizzino τ_0, τ_1 e τ_2, poiché R_τ risulta non ancora sezionato. Inoltre τ_0^- viene aggiunto solo se qualche boundary disposition su R_τ lo ha come disposition cut associato. In definitiva la strategia consiste in:

- verificare se uno o più pezzi di un gruppo hanno la stessa sorting dimension,
- in caso affermativo, qualora venga generato un taglio che li separa, come τ_0, non vanno aggiunti entrambi gli altri due tagli corrispondenti a τ_1 e τ_2 nell'esempio di figura.

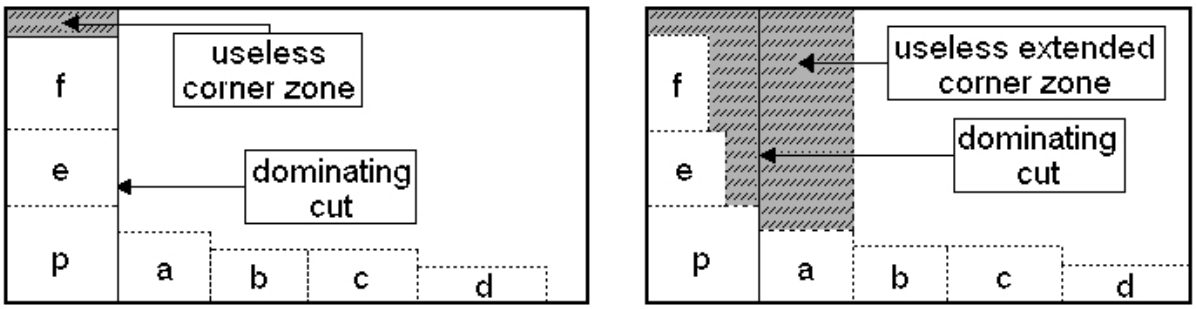


a) evitare tagli paralleli b) evitare tagli perpendicolari

Figura 3.13 Strategie anti-simmetria che evitano tagli

Una terza strategia, denotata con *C*3, può essere proposta quando una (extended) corner zone inutile è presente intorno ad un gruppo di una boundary disposition, nel senso che contiene la parte della regione "ad L" non coperata dai pezzi del gruppo. Negli esempi di Figura 3.14a e di Figura 3.14b una (extended) corner zone inutile si trova intorno al gruppo verticale. In questo caso, va generato solo il nodo associato al disposition cut verticale più a sinistra (*dominating cut* nelle Figure 3.14a,b), poiché nessun rettangolo residuo massimo utile viene da esso sezionato e quindi l'area residua utile resta intatta.

Si sottolinea che bisogna evitare di utilizzare la strategia *C*1 subito dopo la strategia *C*3, altrimenti i disposition cuts paralleli al taglio dominante ed associati alla boundary disposition parziale ereditata dal lato opposto rispetto al vertice in basso a sinistra, vengono erroneamente trascurati.



a) caso di corner zone inutile b) caso di extended corner zone inutile

Figura 3.14 Taglio dominante

3.7 Test sperimentali e risultati computazionali

L'approccio proposto, basato sulla nuova definizione di boundary disposition, è nata per il problema di Cutting single-stock. L'euristica di ricerca ad albero è stata ampiamente applicata ad istanze di test random ed a casi reali provenienti da una industria di trasformazione del vetro piano del Sud Italia. Tutti le istanze di test reali qui riportate sono un-weighted, constrained (cioè con domande di pezzi limitate) e con orientamento libero dei pezzi.

L'euristica è stata inoltre adattata al caso multi-stock, secondo gli scopi dell'industria, per soddisfare l'intera domanda di pezzi utilizzando il minimo numero di stock uguali. Per tale scopo è stata seguita la seguente procedura.

L'euristica descritta è inizialmente applicata per riempire un solo stock. Le domande associate ai tipi di pezzo sono quindi ridotte secondo il numero di pezzi ottenuti su tale primo stock e l'euristica è riapplicata per trovare una nuova soluzione single-stock, proseguendo allo stesso modo finché le domande di tutti i tipi di pezzo sono soddisfatte, in maniera similare a quanto fatto nel lavoro di Hinxman (1980). Per pilotare però la sequenza di soluzioni single-stock, i problemi single-stock vengono considerati con una variante weighted.

In particolare, per ciascun tipo t_i, con dimensioni (l_i , w_i), si denota con d_i la domanda ridotta da soddisfare sugli stock successivi. Per ciascun nuovo stock, il profitto π_i assegnato a ciascun tipo t_i spinge l'euristica a privilegiare i pezzi più grandi affinché siano utilizzati nei primi stock successivi, coerentemente con l'obiettivo di minimizzare il numero totale di stock necessari ad estrarre tutti i pezzi desiderati.

Nel dettaglio, si utilizza un parametro Γ per ciascuno stock, che viene calcolato secondo le seguenti considerazioni. Se l'area totale dei pezzi disponibili, cioè da estrarre sugli stock successivi, pari a $\sum_{i=1}^{n} l_i \cdot w_i \cdot d_i$, non supera l'area dello stock $L_S \cdot W_S$, il problema single-stock viene assunto un-weighted per cui ciascun π_i è posto pari all'area $l_i \cdot w_i$ dei pezzi di tipo t_i. Se invece quell'area totale è molto maggiore dell'area dello stock, il rapporto fra il profitto π_i e l'area $l_i \cdot w_i$ è imposta in modo tale che sia maggiore per i pezzi più grandi, secondo l'obiettivo dichiarato di voler privilegiare i pezzi grandi. Si propone dunque di imporre π_i come quadrato dell'area $l_i \cdot w_i$. Più in particolare, si assume per il parametro Γ l'espressione $\left(\sum_{i=1}^{n} l_i \cdot w_i \cdot d_i\right) \Big/ \left(L_S \cdot W_S\right)$ e per π_i l'espressione generale $\pi_i = \frac{1}{100}\left\lceil 100 \cdot \left(l_i \cdot w_i\right)^{2-1/\Gamma} \right\rceil$, approssimando cioè al centesimo il valore irrazionale $\left(l_i \cdot w_i\right)^{2-1/\Gamma}$.

Nel seguito si riportano i test computazionali ottenuti su quattro istanze di dimensioni medio-grandi con un codice basato sull'euristica multi-stock descritta. I test sono stati realizzati su un sistema Pentium IV 2.66 Ghz, 512 Mbytes RAM, Windows XP SP2. La Tabella 3.1 riporta i dati principali delle quattro istanze, i cui dettagli sono disponibili nelle Tabelle 3.2, 3.3, 3.4, 3.5.

Istanza	Numero di tipo (n)	Domanda totale $\left(\sum_{i=1}^{n} d_i\right)$	Dimensioni dello stock (L_S, W_S)	Area totale dei pezzi (numero di stock) $\frac{\left(\sum_{i=1}^{n} l_i \cdot w_i \cdot d_i\right)}{\left(L_S \cdot W_S\right)}$
BD1	13	508	(5985, 3195)	31.18
BD2	159	159	(5975, 3185)	5.79
BD3	84	465	(5975, 3185)	20.62
BD4	85	473	(5975, 3185)	20.90

Tabella 3.1 Dati delle istanze

Indice tipo	Lunghezza	Ampiezza	Domanda
1	1160	517	16
2	1279	1158	100
3	1162	517	100
4	1018	517	32
5	1279	1014	32
6	1391	637	6
7	1577	1387	6
8	1315	637	30
9	1577	1311	30
10	1196	1076	100
11	1495	1228	18
12	1196	1074	32
13	1495	1305	6

Tabella 3.2 Dettagli dell'istanza BD1

Indice tipo	Lunghezza	Ampiezza	Domanda	Indice tipo	Lunghezza	Ampiezza	Domanda
1	1267	307	1	31	1355	555	1
2	1267	510	1	32	2290	495	1
3	2182	450	1	33	2290	505	1
4	2182	452	1	34	745	525	1
5	2182	520	1	35	1185	440	1
6	1372	248	1	36	935	370	1
7	1252	603	1	37	494	328	1
8	655	495	1	38	630	260	1
9	725	655	1	39	640	275	1
10	945	655	1	40	935	505	1
11	1099	513	1	41	1235	530	1
12	975	655	1	42	2113	485	1
13	1061	636	1	43	2060	782	1
14	1892	560	1	44	375	215	1
15	1274	594	1	45	1210	515	1
16	735	500	1	46	1220	535	1
17	1170	403	1	47	1070	640	1
18	1200	516	1	48	2135	735	1
19	1200	617	1	49	1465	1150	1
20	1200	670	1	50	2135	835	1
21	1370	516	1	51	2185	680	1
22	1370	617	1	52	2290	300	1
23	1370	670	1	53	1130	515	1
24	1370	816	1	54	1250	515	1
25	737	516	1	55	1210	425	1
26	737	617	1	56	2200	520	1
27	737	670	1	57	2175	525	1
28	816	737	1	58	2190	520	1
29	1370	525	1	59	2200	520	1
30	1355	550	1	60	2190	530	1

Tabella 3.3 Dettagli dell'istanza BD2

Indice tipo	Lunghezza	Ampiezza	Domanda	Indice tipo	Lunghezza	Ampiezza	Domanda
61	2075	680	1	111	830	599	1
62	1190	475	1	112	775	664	1
63	2090	475	1	113	1059	381	1
64	1190	490	1	114	950	499	1
65	2090	490	1	115	1049	594	1
66	1065	295	1	116	1144	664	1
67	1525	298	1	117	1685	537	1
68	1455	445	1	118	527	437	1
69	1280	370	1	119	772	417	1
70	1265	500	1	120	772	722	1
71	2120	445	1	121	845	425	1
72	2100	495	1	122	1535	750	1
73	2100	500	1	123	650	210	1
74	2100	510	1	124	550	385	1
75	2080	590	1	125	550	440	1
76	2340	460	1	126	620	460	1
77	1240	440	1	127	999	440	1
78	720	545	1	128	887	437	1
79	1230	390	1	129	1978	440	1
80	1310	190	1	130	1310	382	1
81	1230	510	1	131	1305	391	1
82	1110	650	1	132	1315	389	1
83	1740	320	1	133	1399	412	1
84	1460	510	1	134	1310	529	1
85	2290	510	1	135	1452	404	1
86	2290	515	1	136	2091	231	1
87	1535	455	1	137	2091	331	1
88	2345	550	1	138	2181	391	1
89	405	305	1	139	2269	411	1
90	550	445	1	140	1320	350	1
91	645	400	1	141	1314	364	1
92	950	220	1	142	1305	382	1
93	970	220	1	143	1310	465	1
94	950	590	1	144	3110	529	1
95	970	590	1	145	2186	475	1
96	1265	400	1	146	1412	552	1
97	965	740	1	147	1340	390	1
98	2040	400	1	148	2186	465	1
99	2015	510	1	149	1395	1170	1
100	2015	600	1	150	1855	510	1
101	2035	620	1	151	1965	618	1
102	2020	670	1	152	1302	402	1
103	288	288	1	153	1302	667	1
104	553	188	1	154	1302	742	1
105	462	408	1	155	1302	747	1
106	518	441	1	156	2094	402	1
107	775	381	1	157	1192	362	1
108	594	575	1	158	1302	390	1
109	743	529	1	159	1322	395	1
110	863	429	1				

Tabella 3.3 Dettagli dell'istanza BD2

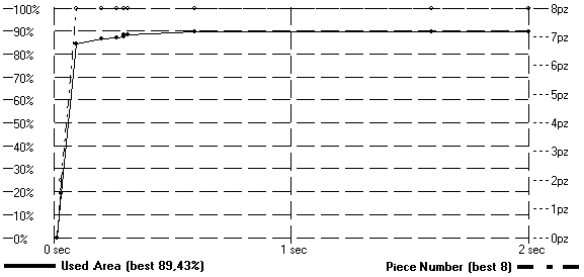
Indice tipo	Lunghezza	Ampiezza	Domanda	Indice tipo	Lunghezza	Ampiezza	Domanda
1	665	425	8	43	1089	370	2
2	624	272	8	44	1230	387	4
3	686	292	4	45	1230	394	2
4	686	384	4	46	1230	494	2
5	686	390	8	47	1230	269	1
6	564	564	4	48	1094	522	2
7	869	392	4	49	1194	522	2
8	939	327	8	50	1194	622	2
9	664	664	2	51	2094	522	6
10	1200	292	4	52	2194	724	1
11	1299	520	4	53	2281	764	1
12	1474	384	4	54	2281	811	1
13	1474	388	4	55	2174	950	2
14	1474	390	16	56	2174	1088	6
15	1472	392	4	57	1602	427	4
16	1474	396	4	58	2413	427	24
17	1384	510	32	59	2403	580	2
18	1515	390	8	60	997	582	1
19	1528	384	4	61	1349	660	2
20	1386	600	2	62	1274	867	2
21	686	390	2	63	1274	906	3
22	686	396	2	64	1244	935	10
23	1503	396	2	65	1224	983	6
24	1515	390	2	66	1274	935	22
25	1449	372	4	67	1274	983	22
26	1942	407	4	68	1350	931	4
27	2102	502	4	69	1244	1063	2
28	1174	722	4	70	1274	1034	2
29	1019	470	4	71	1574	749	1
30	1057	416	6	72	1274	1063	2
31	1057	480	1	73	1199	588	2
32	1788	364	2	74	1199	688	2
33	1788	607	2	75	1199	792	3
34	771	474	4	76	1199	830	2
35	771	476	2	77	1169	860	15
36	771	594	2	78	1199	856	4
37	1104	344	4	79	1149	908	9
38	1104	348	8	80	1199	860	33
39	1125	476	2	81	1199	901	3
40	1135	474	4	82	1199	908	33
41	1125	594	2	83	1169	988	7
42	1104	594	1	84	1199	988	3

Tabella 3.4 Dettagli dell'istanza BD3

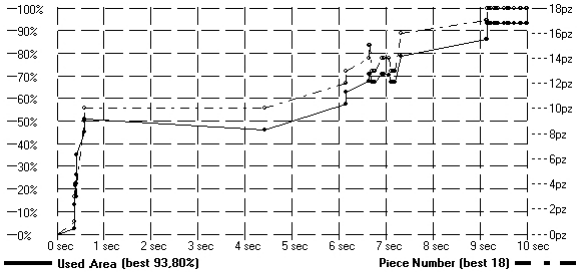
Indice tipo	Lunghezza	Ampiezza	Domanda	Indice tipo	Lunghezza	Ampiezza	Domanda
1	665	425	8	44	1089	370	2
2	624	272	8	45	1230	387	4
3	686	292	4	46	1230	394	2
4	686	384	4	47	1230	494	2
5	686	390	8	48	1230	269	1
6	564	564	4	49	1094	522	2
7	869	392	4	50	1194	522	2
8	939	327	8	51	1194	622	2
9	664	664	2	52	2094	522	6
10	1200	292	4	53	2194	724	1
11	1299	520	4	54	2281	764	1
12	1344	488	8	55	2281	811	1
13	1474	384	4	56	2174	950	2
14	1474	388	4	57	2174	1088	6
15	1474	390	16	58	1602	427	4
16	1472	392	4	59	2413	427	24
17	1474	396	4	60	2403	580	2
18	1384	510	32	61	997	582	1
19	1515	390	8	62	1349	660	2
20	1328	384	4	63	1274	867	2
21	1386	600	4	64	1274	905	3
22	686	390	2	65	1244	935	10
23	686	396	2	66	1224	983	6
24	1503	396	2	67	1274	935	22
25	1515	390	2	68	1274	983	22
26	1449	372	4	69	1350	931	4
27	1942	407	4	70	1244	1063	2
28	2102	502	4	71	1274	1034	2
29	1174	722	4	72	1574	749	1
30	1019	470	4	73	1274	1063	2
31	1057	416	6	74	1199	588	2
32	1057	480	1	75	1199	688	2
33	1788	364	2	76	1199	792	3
34	1788	607	2	77	1199	830	2
35	771	474	4	78	1169	860	15
36	771	476	2	79	1199	856	4
37	771	594	2	80	1149	908	9
38	1104	344	4	81	1199	860	33
39	1104	348	8	82	1199	901	3
40	1125	476	2	83	1199	908	33
41	1135	474	4	84	1169	988	7
42	1125	594	2	85	1199	988	3
43	1104	594	1				

Tabella 3.5 Dettagli dell'istanza BD4

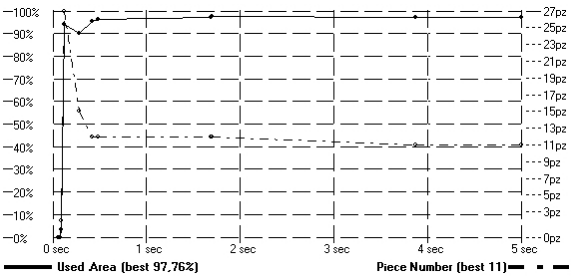
Nel seguito si riportano i risultati ottenuti per il caso single-stock, con riferimento al primo stock. Il tempo di run disponibile assume cinque diversi valori: 2 sec, 5 sec, 10 sec, 20 sec e 30 sec. Le Figure 3.15a, 3.15b, 3.15c e 3.15d riportano la relazione fra la percentuale usata per il primo stock nella migliore soluzione corrente, rispetto all'avanzamento del tempo di run. Esse riportano anche il numero di pezzi nella migliore soluzione corrente. La Figura 3.15a è relativa all'istanza BD1, con un massimo di 2 secondi per stock. Si noti che per le altre istanze a volte la percentuale dello stock utilizzata decresce, come nella Figura 3.15b, per l'istanza BD2, che ha molti tipi di pezzi con diverse dimensioni. In particolare si notano tre step decrescenti fra i 4 e gli 8 secondi. Questo fenomeno è dovuto al fatto che se si utilizzano meno pezzi, ma più grandi, la somma dei profitti dei pezzi può crescere. Un comportamento simile è presente in Figura 3.15c per l'istanza BD3, entro il primo secondo, in cui il numero di pezzi scende velocemente da 20 a 12. La Figura 3.15d mostra un comportamento analogo per l'istanza BD4 per la quale lo step di riduzione è addirittura maggiore, prima da 27 a 12 entro il primo secondo e poi ad 11 prima degli 8 secondi, con una percentuale di utilizzo finale pari al 96.92%. Una buona percentuale è raggiunta anche per le altre tre istanze, vale a dire 89.43% per BD1, 93.08% per BD2 e 97.76% per BD3.



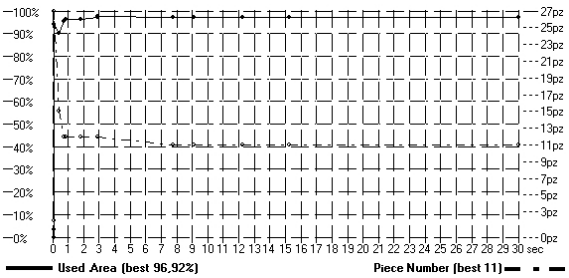
a) BD1 con 2 secondi per stock



b) BD2 con 10 secondi per stock



c) BD3 con 5 secondi per stock



d) BD4 con 30 secondi per stock

Figura 3.15 Migliore soluzione con lo scorrere del tempo di run

La Tabella 3.6 riporta i risultati ottenuti con riferimento al caso multi-stock, seguendo la procedura euristica greedy precedentemente descritta. Vengono analizzati due parametri di performance, vale a dire il numero di stock utilizzati e la percentuale di utilizzo sull'ultimo stock.

Istanza	2 sec	5 sec	10 sec	20 sec	30 sec
BD1	37 / 40.50%	36 / 10.75%	36 / 47.25%	36 / 6.75%	36 / 20.25%
BD2	7 / 85.54%	7 / 47.27%	7 / 27.88%	7 / 27.88%	7 / 15.22%
BD3	23 / 7.45%	23 / 5.51%	23 / 3.85%	22 / 54.03%	22 / 52.19%
BD4	23 / 31.26%	23 / 51.40%	23 / 32.65%	23 / 18.73%	23 / 11.22%

Tabella 3.6 Risultati per il caso multi-stock

Per l'istanza BD1 si può notare una relazione non monotòna fra il numero di stock utilizzati (tenendo conto anche della percentuale sull'ultimo stock) ed il tempo massimo di run disponibile per ciascuno stock. Questo è dovuto all'approccio greedy che ottimizza uno stock alla volta. Infatti una migliore soluzione sugli stock iniziali non implica una migliore soluzione complessiva per il multi-stock, sebbene l'euristica utilizzata sia pilotata con profitti variabili stock per stock per evitare questo fenomeno. Inoltre si può notare che, per tutte e quattro le istanze, sono ottenuti buoni risultati già con soli 2 secondi per stock. Ad esempio, per le istanze BD2 e BD4 il numero di stock utilizzati è lo stesso indipendentemente dal tempo di run massimo per stock, da 2 a 30 secondi. Per l'istanza BD1 il miglior numero (cioè minimo) di stock utilizzati è raggiunto già con 5 secondi. Solo l'istanza BD3 necessita di più di 10 secondi per stock per raggiungere il numero minimo di stock utilizzati, ma si noti che la percentuale di utilizzo dell'ultimo stock è molto bassa (7.45% con 2 secondi, 5.51% con 5 secondi e 3.85% con 10 secondi), il che vuol dire che c'è ugualmente poca differenza fra i risultati relativi ai diversi tempi massimi di run per singolo stock.

Si riporta inoltre che, per 21 delle 36 differenti soluzioni single-stock trovate con 5 secondi per l'istanza BD1, la soluzione finale è stata trovata entro il primo secondo. Inoltre, per i primi 10 stock di tale istanza, la percentuale utilizzata è fra l'88% ed il 97%. Quindi si può affermare che, per tutti i test descritti, l'euristica ha prodotto buone soluzioni fattibili in un tempo computazionale basso.

La Figura 3.16 riporta lo schema di taglio di una soluzione per il primo stock per l'istanza BD2 con un massimo di 5 secondi per stock. Essa ha solo il 6.44% di scarto. Si può notare la struttura tipica delle boundary dispositions. Si noti inoltre che in questa istanza ciascun tipo t_i ha domanda $d_i = 1$ ed i 159 pezzi hanno dimensioni molto diverse fra loro.

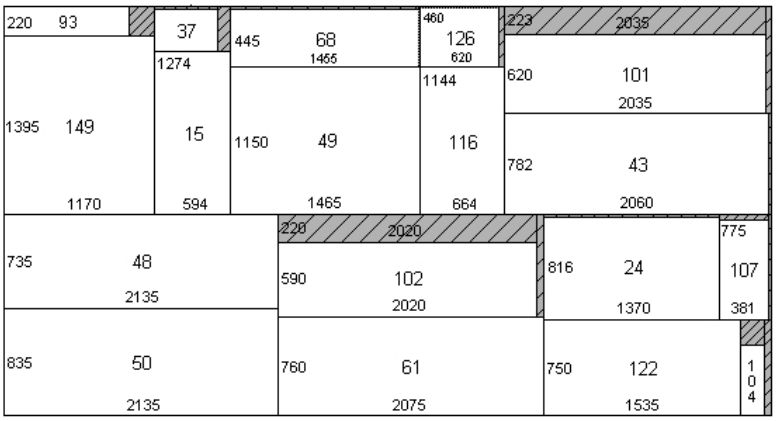


Figura 3.16. Il primo stock per l'istanza BD2 in 5 secondi

Si sottolinea infine che la tecnica utilizzata per passare dal caso single-stock al caso multi-stock ha il pregio di riuscire a gestire fin dal primo stock un riadattamento dei valori associati ai singoli pezzi, al contrario di altre tecniche classiche, come quelle di Hinxman (1980) e di Belov (2003).

3.7 Conclusioni e prospettive

E' stato descritto un approccio euristico di ricerca ad albero, basato sulla nuova idea di boundary disposition. L'euristica proposta può essere vista come un ibrido fra gli approcci top-down e bottom-up. Infatti, i gruppi di pezzi sono costruiti come nell'approccio bottom-up, mentre la struttura della ricerca è ispirata all'approccio top-down, con il taglio ricorsivo dello stock iniziale e dei susseguenti sotto-rettangoli. Il modo di scegliere i tagli è basato sui pezzi della boundary disposition. C'è anche una similarità con l'approccio strip heuristic approach poiché i pezzi di una boundary disposition possono produrre buone soluzioni a strisce se si utilizza l'intero insieme di disposition cuts verticali o l'intero insieme di disposition cuts orizzontali.

L'algoritmo privilegia le boundary dispositions che utilizzano al meglio entrambi le dimensioni del rettangolo corrente, pensalizzando così la ricerca di soluzioni ottenibili con boundary disposition costituite da gruppi dominati.

L'euristica è stata soddisfacentemente applicata nell'ambito produttivo di una delle più grandi industrie di trasformazione del vetro piano del Sud Italia, incoraggiando dunque ulteriori investigazioni sul nuovo metodo risolutivo proposto.

Ad esempio, essa è stata anche adattata, secondo una metodologia da assestare, alla variante del problema in cui il riutilizzo futuro degli scarti prodotti influisce sulla funzione obiettivo, che però diventa non lineare e può essere in prima battuta espressa come

$$\frac{2+\left(S\cdot N_S\right)^k+\gamma\cdot B}{1+\left(\frac{P}{1+N_P}\right)^k}$$

dove S è la porzione dello stock, da 0 ad 1, di *soft waste*, cioè di scarto riutilizzabile, B è la porzione di *bad waste*, cioè di scarto non riutilizzabile, N_S è il numero di rettangoli di scarto riutilizzabile, P è la porzione di stock utilizzata per i pezzi, N_P è il numero di pezzi estratti, γ e k sono parametri inizialmente settati secondo le quantità e le aree dei pezzi iniziali, e la funzione va minimizzata.

Si riesce a trattare la non linearità della funzione obiettivo utilizzata, grazie all'approccio costruttivo su cui si basa il metodo proposto.

Studi su questa ed altre interessanti varianti del problema, per tener conto di aspetti reali, si trovano in diversi recenti lavori di letteratura, fra cui Whitwell (2004) ed Arenales, Cherri e Yanasse (2007). Dunque, interessanti prospettive riguardano l'adattamento alle varianti del problema studiate in questi ed altri lavori.

3.7.1 Adattamento alla versione con orientamento libero

Per la versione del problema con orientamento libero, vanno adottate le seguenti differenze. Per ciascun tipo t_i, con dimensioni (l_i, w_i), si denoti con t'_i un tipo di pezzi ruotato, cioè con dimensioni (w_i, l_i).

Nel calcolo dei valori di knapsack function, basati su domande illimitate, tutti i tipi t_i e tutti i tipi t'_i vengono considerati.

Per la procedura di costruzione delle boundary disposition, ricordando che n è il numero di differenti tipi t_i, $2\cdot n$ diversi possibili corner types devono essere considerati, cioè sia i tipi iniziali t_i sia quelli t'_i ottenuti con la rotazione, per ogni i. Nella procedura di

costruzione dei gruppi di pezzi, per ciascuna coppia di tipi t_i e t'_i, bisogna considerare una domanda condivisa d_i pari a quella inizialmente associata al tipo t_i.

Per tutti gli upper bounds basati su rilassamento mono-dimensionale e continuo, bisogna considerare una doppia possibilità di contenere i pezzi in un qualsiasi dato rettangolo, secondo i due possibili orientamenti. La stessa regola va applicata per verificare l'utilità dei massimi rettangoli residui, determinando così l'eventuale riduzione del numero di (extended) corner zones inutili.

Per i lower bounds, le migliori strips vengono costruite considerando entrambe le rotazioni e con una domanda condivisa per ciascun coppia di tipi t_i e t'_i.

Capitolo 4

Nuovi upper bounds

Da molti anni, la comunità di ricerca operativa ha studiato ampiamente i problemi di Cutting Stock, dato l'elevato numero di applicazioni, nei campi dell'ingegneria industriale, della logistica, dei processi produttivi, dell'informatica. Fra gli altri, interessanti esempi si trovano nella produzione di vetro, carta, legno, metalli.

I lavori di Kantorovich (1939) e Brooks, Smith, Stone e Tutte (1940) possono essere considerati storicamente introduttivi. Utili classificazioni dei problemi di Cutting si trovano in Dyckhoff (1990), Fayard, Hifi e Zissimopoulos (1998) e recentemente in Wäscher, Haußner e Schumann (2007).

In questo Capitolo si affronta il problema del Cutting Stock bidimensionale a ghigliottina, denotato in letteratura anche con l'acronimo TDGC. Esso consiste nel trovare il miglior insieme di oggetti rettangolari piccoli, o pezzi, estraibile da un dato oggetto rettangolare grande, o stock, utilizzando solo tagli a ghigliottina, che cioè abbiano inizio e termine sui bordi del rettangolo tagliato.

Una certa varietà di approcci risolutivi sono stati proposti in letteratura per il problema TDGC, ad esempio in termini di programmazione lineare (Lodi e Monaci, 2003; Belov, 2003) oppure con schemi di programmazione dinamica (Hifi e Zissimopoulos, 1996) o tramite meta-euristiche (Alvarez-Valdés, Parajon e Tamarit, 2002; Puchinger, 2005).

Inoltre, esistono anche approcci ad hoc, che possono essere distinti in due classi. Gli approcci *top-down* tagliano ricorsivamente lo stock iniziale ed i susseguenti sottorettangoli in due parti, fino ad ottenere i pezzi finali. Questo approccio segue uno schema di ricerca ad albero, dove il *branching* consiste nel tagliare un selezionato sotto-rettangolo dello stock, generando un *branch* per ogni possibile taglio, in entrambi gli orientamenti, cioè orizzontale e verticale. Lavori relativi a tale approccio sono Herz (1972), Christofides e Whitlock (1977), Beasley (1985), Hifi e Zissimopoulos (1997) ed Hifi (2004).

Gli approcci *bottom-up* seguono una strategia opposta, attraverso un *merge* dei rettangoli (chiamati *builds* o *meta-rettangoli*) contenenti pezzi. Quindi essi generano meta-rettangoli sempre più grandi fino a riempire al meglio lo stock iniziale. Questo tipo di procedura inizia con un insieme di meta-rettangoli corrispondenti ai pezzi iniziali da estrarre. Utilizza inoltre solo due modi (*orizzontale* o *verticale*) per eseguire il merge di una coppia di essi, così da soddisfare il taglio a ghigliottina. Lavori relativi a tale approccio sono Wang (1983), Viswanathan e Bagchi (1993), Hifi (1997b), Cung, Hifi e Le Cun (2000) e León, Miranda, Rodríguez e Segura (2007).

Gli approcci ad hoc usano gli upper bounds allo scopo di accelerare le procedure di enumerazione implicita. In questo Capitolo, si descrivono miglioramenti originali per gli upper bounds basati sulla cosiddetta *knapsack function*. Tali miglioramenti sono ottenuti sfruttando le *coordinate normalizzate*. Il nuovo metodo è stato testato su istanze di letteratura, mostrando incoraggianti risultati in termini di valore degli upper bounds, in particolare per istanze di piccola e media dimensione e nel caso non pesato.

Il primo paragrafo dà la definizione del problema e delle sue varianti. Il secondo paragrafo descrive le coordinate normalizzate e la knapsack function, così come date in letteratura. Il terzo paragrafo riassume gli upper bounds usualmente utilizzati in letteratura, dettagliando miglioramenti originali. Il quarto paragrafo mostra risultati computazionali, mentre nel quinto paragrafo sono date le conclusioni.

4.1 Definizione del problema

Nel problema del Cutting Stock bidimensionale a ghigliottina (TDGC), i dati di partenza consistono di:

- un rettangolo iniziale S, detto stock, con dimensioni intere (L_S, W_S),
- un insieme di n quadruple $T = \{t_i : i = 1, \ldots, n\} = \{(l_1, w_1, d_1, \pi_1), \ldots, (l_n, w_n, d_n, \pi_n)\}$ che rappresentano n tipi di pezzi rettangolari. Ciascun tipo t_i ha lunghezza l_i, ampiezza w_i , area $l_i \cdot w_i$, domanda d_i e peso (o profitto) π_i, tutti interi positivi, e tali che valgano le due relazioni $l_i \le L_S$ e $w_i \le W_S$, cioè che almeno un relativo pezzo è estraibile dallo stock.

Un *cutting pattern* è un vettore $B = (b_1, \ldots, b_n)$, di n elementi, che rappresenta un insieme di pezzi contenente, per ciascun tipo t_i, esattamente b_i pezzi. Un cutting pattern è *feasible* (o fattibile) quando le due seguenti condizioni sono soddisfatte:

- esso è *bounded* (o limitato), vale a dire che, per ciascun tipo t_i, il numero b_i di pezzi non è superiore alla corrispondente domanda d_i, ovvero che valga la relazione $b_i \le d_i$ per ogni $i = 1, \ldots, n$,
- esso è *consistent* (o coerente), vale a dire che il corrispondente insieme di pezzi sia estraibile dallo stock, utilizzando un insieme di tagli a ghigliottina verticali ed orizzontali. L'insieme di tagli associato è anche chiamato *cutting scheme* (o schema di taglio). Lo schema di taglio potrebbe essere non univoco, ma questo è poco importante ai fini dei ragionamenti descritti in questo Capitolo.

Sia $\Pi(B)$ il valore di funzione obiettivo associato ad un cutting pattern B. In genere $\Pi(B)$ rappresenta uno dei due seguenti valori:

- la somma delle aree dei pezzi estratti (variante *un-weighted* o non pesata),
- la somma dei profitti dei pezzi estratti (variante *weighted* o pesata).

Nel seguito, π_i denoterà o il profitto del tipo t_i , se il problema è pesato, o la sua area in caso contrario, cioè secondo la definizione della funzione obiettivo. Di conseguenza, il profitto totale associato ad un cutting pattern può essere sempre scritto come $\Pi(B) = \sum_{i=1}^{n} (b_i \cdot \pi_i)$.

In definitiva, il problema TDGC può essere formalizzato nel seguente modo:

massimizzare	$\Pi(B)$
al variare dei	*feasible cutting patterns B*

In accordo alle definizioni date in precedenza, per ciascun tipo t_i, il valore $D_i = \lfloor L_S / l_i \rfloor \cdot \lfloor W_S / w_i \rfloor$ è il massimo numero di pezzi di tipo t_i estraibili dallo stock S. Se la domanda d_i è strettamente inferiore al corrispondente valore D_i, per almeno un tipo t_i , allora il problema è classificato come *constrained* (o vincolato), ed *un-constrained* (o non vincolato) se, al contrario, la relazione $d_i \ge D_i$ vale per ogni $i = 1, \ldots, n$. Si può dunque supporre che $d_i \le D_i$ valga per ogni tipo t_i, limitando dunque, senza perdita di generalità, il caso un-constrained alla condizione $\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} D_i$.

Inoltre, si possono distinguere due diverse varianti del problema, secondo la possibilità di estrarre i pezzi ruotati di 90 gradi (*free orientation*) oppure no (*fixed orientation*).

Il problema TDGC è NP-hard poiché, se $W_S = w_1 = w_2 = \ldots = w_n$, esso equivale al problema dello zaino mono-dimensionale, che è noto essere NP-hard.

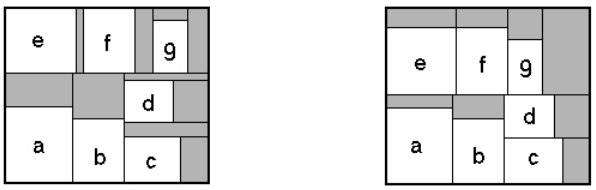
In questo Capitolo, sono descritti dei miglioramenti originali rispetto ai classici upper bounds utilizzati per la versione constrained del problema. Essi sono adatti alle versioni ad orientamento fisso, sia pesato che non pesato. I miglioramenti sono basati sullo sfruttamento delle coordinate normalizzate, che vengono riassunte nel dettaglio nel prossimo paragrafo. Essi hanno lo scopo di ottenere, per gli upper bounds basati sulla knapsack function, dei valori ridotti rispetto ai metodi classici usati per calcolarli.

4.2 Coordinate normalizzate e knapsack function

In questo paragrafo, si riassumono le coordinate normalizzate, che sono state proposte da Herz (1972) e da Christofides e Withlock (1977). Successivamente si riporta la definizione di knapsack function, dandone anche la formulazione proposta da Gilmore e Gomory (1966).

4.2.1 Coordinate normalizzate

Christofides e Withlock (1977) hanno definito le soluzioni normalizzate. Si tratta di soluzioni in cui nessun pezzo e nessun taglio può essere traslato verso sinistra o verso il basso, rispettando il taglio a ghigliottina e senza sovrapporsi ad altri pezzi (Figura 4.1). E' sufficiente cercare, nella risoluzione di un problema, solo soluzioni normalizzate. Infatti, è facile mostrare che ogni altra soluzione può essere trasformata in una soluzione normalizzata attraverso un opportuna traslazione dei suoi pezzi e dei suoi tagli, come mostrato in Figura 4.1. Sulla base di tale proprietà, nel seguito si considereranno implicitamente solo coordinate normalizzate come se tutte le altre non esistessero.



a) soluzione non normalizzata b) soluzione normalizzata

Figura 4.1 Soluzioni normalizzate

Ciascun taglio è identificabile dal suo orientamento e dalla sua coordinata rispetto al vertice in basso a sinistra del rettangolo da esso tagliato. Le possibili coordinate orizzontali per tagli verticali in soluzioni normalizzate sono chiamate coordinate normalizzate orizzontali, e le possibili coordinate verticali per i tagli orizzontali sono chiamate coordinate normalizzate verticali.

Le coordinate normalizzate possono essere calcolate da una procedura di programmazione dinamica. Si supponga di indicizzare i tipi di pezzo secondo ampiezza non decrescente. Sia inoltre $w = g_{v,i}(x)$ la minima ampiezza tale che, utilizzando solo pezzi di tipi con indice non superiore ad i e con ampiezza non superiore a w, il valore di coordinata orizzontale x per possibili tagli verticali sia ottenibile sommando le lunghezze di un sotto-insieme di questi pezzi. La funzione $g_{v,i}(x)$ può essere calcolata attraverso la seguente relazione ricorsiva, dove, per ciascun i, l_i e w_i sono le dimensioni del tipo t_i, e d_i è la sua domanda, come definito nel paragrafo 4.1 (Hifi, 2004):

$$g_{v,i}(x) = \begin{cases} 0 & \text{if} \quad i = 0 \quad \text{and} \quad x = 0 \\ \infty & \text{if} \quad i = 0 \quad \text{and} \quad x > 0 \\ g_{v,i-1}(x) & \text{if} \quad i > 0 \quad \text{and} \quad x < l_i \\ \min\left\{ g_{v,i-1}(x), \max\left\{ w_i, \min_{1 \le k \le \min\left\{ d_i, \left\lfloor \frac{x}{l_i} \right\rfloor \right\}} \left\{ g_{v,i-1}(x - k \cdot l_i) \right\} \right\} \right\} & \text{else} \end{cases}$$

Si noti che per ogni fissato i, e per ogni possibile valore x, il valore $g_{v,i}(x)$ appartiene certamente all'insieme $\{w_0, w_1, \ldots, w_i, \infty\}$ dove $w_0 = 0$. Se $g_{v,i}(x) = w_i$, allora $g_{v,j}(x) = w_i$ per ogni $j > i$, e se $g_{v,j}(x) = w_i < w_j$, allora $i < j$ e $g_{v,i}(x) = w_i$. Tali proprietà valgono poiché $j > i$ implica $w_j \geq w_i$.

Sia $N_v(l, w)$ l'insieme delle coordinate normalizzate orizzontali per i possibili tagli verticali su un rettangolo R contenuto nello stock S, con lunghezza l ed ampiezza w e denotato nel seguito anche con $R(l, w)$. Come conseguenza delle definizioni introdotte, si ha che $N_v(l, w) = \{x \leq l : g_{v,n}(x) \leq w\}$. Si noti che si è considerato anche il valore l fra i possibili valori di x. Ciò è necessario per gli scopi di questo Capitolo. Infatti, l'insieme $N_v(l, w)$ è considerato sia come insieme di coordinate di tagli verticali, sia come insieme di lunghezze.

Un'analoga formulazione è utilizzata per i tagli orizzontali, indicizzando i tipi di pezzo per lunghezza non decrescente, determinando così la definizione sia della funzione $g_{h,i}(y)$ sia dell'insieme $N_h(l, w)$ di coordinate normalizzate verticali per i possibili tagli orizzontali, ottenibile formalmente come $\{y \leq w : g_{h,n}(y) \leq l\}$.

Tutti gli insiemi $N_v(l, w)$ ed $N_h(l, w)$ contengono lo zero. Si definisce inoltre come l_{min} il minimo elemento positivo di $N_v(L_S, W_S)$ e come w_{min} il minimo elemento positivo di $N_h(L_S, W_S)$, dove L_S e W_S sono le dimensioni dello stock. Dunque, $l_{\min} = \min_{i=1,\ldots,n} \{l_i\}$ ed $w_{\min} = \min_{i=1,\ldots,n} \{w_i\}$, cioè l_{min} e w_{min} sono rispettivamente pari alla minima fra le lunghezze dei tipi, ed alla minima fra le ampiezze dei tipi.

4.2.2 Knapsack function

Gilmore e Gomory (1966) hanno dettagliatamente studiato le cosiddette knapsack functions, che essi stessi hanno introdotto. Si denota nel seguito con $f(l, w)$ il valore associato dalla knapsack function ad un rettangolo generico $R(l, w)$.

Sia $f_0(l, w)$ il valore della migliore soluzione parziale per R contenente non più di un pezzo. Dunque, $f_0(l, w)$ è uguale a $\max_{i=1,\ldots,n} \{\pi_i : l_i \leq l \text{ and } w_i \leq w\}$ se almeno un pezzo di qualsiasi tipo t_i può essere ottenuto da R, e 0 altrimenti. Quindi i valori $f(l, w)$, al variare di l e w, sono calcolabili con la seguente equazione ricorsiva (Hifi, 1997a):

$$f(l,w) = \begin{cases} 0 & \text{if } l = 0 \text{ or } w = 0 \\ \max\left\{ f_0(l,w),\ \max_{\substack{x_1,x_2: \\ x_1+x_2 \leq l \\ 0<x_1 \leq x_2}} \{f(x_1,w) + f(x_2,w)\},\ \max_{\substack{y_1,y_2: \\ y_1+y_2 \leq w \\ 0<y_1 \leq y_2}} \{f(l,y_1) + f(l,y_2)\} \right\} & \text{else } \begin{bmatrix} 1 \leq l \leq L_S \\ 1 \leq w \leq W_S \end{bmatrix} \end{cases} \quad (4.1)$$

Ciascun valore $f(l, w)$ è ottenuto come massimo fra tre termini. Il primo è $f_0(l, w)$. Gli altri due termini sono relativi rispettivamente ai tagli verticali ed ai tagli orizzontali. Il termine per i tagli verticali corrisponde al massimo fra i valori associati a tutti i modi possibili di tagliare con tagli verticali il rettangolo R in due o tre sotto-rettangoli, uno di dimensioni (x_1, w), il secondo di dimensioni (x_2, w) ed un terzo possibile sotto-rettangolo vuoto di dimensioni $(l - x_1 - x_2, w)$. Una definizione simile vale per il termine relativo ai tagli orizzontali.

Sulla base di quanto detto, $f(l, w)$ rappresenta il valore della soluzione ottima per il problema con domanda illimitata (versione un-constrained) relativo al rettangolo $R(l, w)$. Tale valore è generalmente utilizzato come upper bound, per il rettangolo R, nella versione constrained del problema. Quindi $f(L_S, W_S)$ è un upper bound per il problema constrained sullo stock S.

Si sottolinea, comunque, che per ciascuna possibile versione del problema, è possibile definire una corrispondente knapsack function f che associa ad ogni coppia di dimensioni (l, w) il valore della soluzione fattibile ottima per $R(l, w)$. L'idea su cui si basa il miglioramento degli upper bounds consiste nel calcolare un upper bound ridotto come valore della soluzione fattibile ottima per R con domanda illimitata, ma restringendo l'attenzione alle soluzioni il cui schema di taglio non appartenga ad un determinato insieme di schemi di taglio. Tale insieme è costruito in modo tale che gli schemi in esso contenuti non corrispondano a nessuna soluzione normalizzata che sia fattibile rispetto alla versione con domanda limitata. Più grande è l'insieme di schemi esclusi, minore può essere il valore dell'upper bound.

Tale idea è utilizzata per ottenere upper bound di valore ridotto attraverso un affinamento della knapsack function, i cui valori, per semplicità, continueranno ad essere denotati con $f(l, w)$, continuando inoltre ad utilizzare il nome di "knapsack function".

4.3 Upper bounds migliorati

In letteratura, per la versione constrained del problema, cioè con domande limitate di pezzi, si utilizzano due tipi di upper bound. Il primo è basato sul rilassamento che consiste nel trascurare i vincoli di domanda, e si parla dunque di rilassamento di domanda illimitata. Tale upper bound si ottiene come valore della knapsack function associato al relativo rettangolo, come descritto nel paragrafo 4.2.2. Il secondo upper bound è basato su un rilassamento mono-dimensionale, nel senso che vengono considerate solo le aree dei pezzi e quella dello stock S, come se si trattasse di un problema mono-dimensionale.

Si denoterà nel seguito con $f_{1D}(l, w)$ il valore dell'upper bound associato al generico rettangolo $R(l, w)$, ottenuto con il rilassamento di mono-dimensionalità, assumendo che, per ciascun tipo di pezzi t_i, il numero di pezzi disponibile sia uguale alla domanda d_i , supposta nel paragrafo 4.1 non maggiore del prodotto $\lfloor L_S / l_i \rfloor \cdot \lfloor W_S / w_i \rfloor$. Il valore $f_{1D}(l, w)$ dipende solo dall'area $A = (l \cdot w)$ del rettangolo e può dunque essere denotato anche con $f_{1D}(A)$. Una lista di valori per $f_{1D}(A)$, quando il valore del parametro A va da 0 al valore dell'area dello stock, è ottenibile con una procedura di programmazione dinamica (León, Miranda, Rodríguez e Segura, 2007).

Per un rettangolo $R(l, w)$, un upper bound $UB(l, w)$ è in genere ottenuto come minimo fra il valore di knapsack function $f(l, w)$ e l'upper bound mono-dimensionale $f_{1D}(l, w)$, vale a dire utilizzando l'espressione $UB(l, w) = \min\{ f(l, w), f_{1D}(l, w) \}$.

Entrambi i tipi di approcci risolutivi ad hoc, quello top-down e quello bottom-up, già brevemente descritti, utilizzano upper bounds non solo per l'intero stock, ma anche per ogni possibile sottorettangolo contenuto nello stock (Hifi, 2004; León, Miranda, Rodríguez e Segura, 2007) per cui è di grande utilità migliorare tutti i valori $UB(l, w)$.

Nel seguito si espongono i dettagli dei miglioramenti originali proposti. Essi sono organizzati logicamente in cinque passi, che sono dunque spiegati in cinque diversi sotto-paragrafi. Al primo passo si spiega come sfruttare le coordinate normalizzate per ridurre i valori degli upper bound, attraverso l'affinamento della knapsack function.

Nei successivi tre passi si raffina ulteriormente l'idea del primo passo. Infatti, il secondo passo consiste nel considerare solo coordinate normalizzate orizzontali compatibili con la corrente coordinate normalizzata verticale e viceversa. Al terzo passo, ciascuna dimensione di un rettangolo è divisa solo in coppie di valori corrispondenti a coordinate normalizzate compatibili con l'altra dimensione del rettangolo. Il quarto passo introduce il concetto di *coordinate normalizzate raggiungibili*. Si sottolinea che ciascun passo costituisce un miglioramento rispetto al precedente, come dimostrato anche nel quarto sotto-paragrafo con un opportuno esempio, e che la suddivisione in passi ha l'obiettivo di semplificare e rendere più chiara la comprensione.

Infine, il quinto passo migliora l'upper bound mono-dimensionale associato all'intero stock, secondo le definizioni date al primo passo.

4.3.1 Knapsack function con coordinate normalizzate

Herz (1972) e Beasley (1985) hanno proposto due approcci esatti per risolvere la versione un-constrained del problema, mostrando come ridurre il corrispondente sforzo computazionale per calcolare il solo valore $f(L_S, W_S)$, attraverso l'utilizzo delle coordinate normalizzate associate alla domanda illimitata di pezzi. La stessa idea può però essere sfruttata per affinare la relazione (4.1), applicando cioè un miglioramento per il caso di domanda limitata.

A tal fine, è necessario introdurre alcune definizioni. Per un rettangolo $R(l, w)$, si denotano rispettivamente con $f_v(l, w)$ il termine parziale della knapsack function associato ai

tagli verticali, e con $f_h(l, w)$ il termine parziale associato ai possibili tagli orizzontali. Inoltre, per una qualsiasi lunghezza l, si denota con l^* il massimo elemento di $N_v(L_S, W_S)$ che non sia maggiore di l, e, per un qualsiasi valore di w, si denota con w^* il massimo elemento di $N_h(L_S, W_S)$ non maggiore di w.

La formulazione rivisitata che segue calcola i valori di knapsack function solo per le coppie (l, w) di coordinate normalizzate, cioè tali che $l \in N_v(L_S, W_S)$ e che $w \in N_h(L_S, W_S)$, utilizzando solo tagli con coordinate normalizzate, escludendo così schemi di taglio non associati a nessuna soluzione normalizzata fattibile rispetto alla domanda limitata di pezzi.

$$f(l,w) = \begin{cases} 0 & \textit{if} \quad l = 0 \quad \text{or} \quad w = 0 \\ \max\{f_0(l,w),\, f_v(l,w),\, f_h(l,w)\} & \textit{else} \quad [l \geq l_{\min} \quad \text{and} \quad w \geq w_{\min}] \end{cases} \tag{4.2}$$

$$f_v(l,w) = \begin{cases} \max\limits_{\substack{x \in N_v(W_S, L_S): \\ 0 < x \leq \lfloor l/2 \rfloor}} \{f(x,w) + f^*(l-x,w)\} & \textit{if} \quad l \geq 2 \cdot l_{\min} \\ 0 & \textit{else} \quad [l < 2 \cdot l_{\min}] \end{cases} \tag{4.3}$$

$$f_h(l,w) = \begin{cases} \max\limits_{\substack{y \in N_h(W_S, L_S): \\ 0 < y \leq \lfloor w/2 \rfloor}} \{f(l,y) + f^*(l,w-y)\} & \textit{if} \quad w \geq 2 \cdot w_{\min} \\ 0 & \textit{else} \quad [w < 2 \cdot w_{\min}] \end{cases} \tag{4.4}$$

dove

$$l \in N_v(W_S, L_S)\ ,\ w \in N_h(W_S, L_S)$$

ed

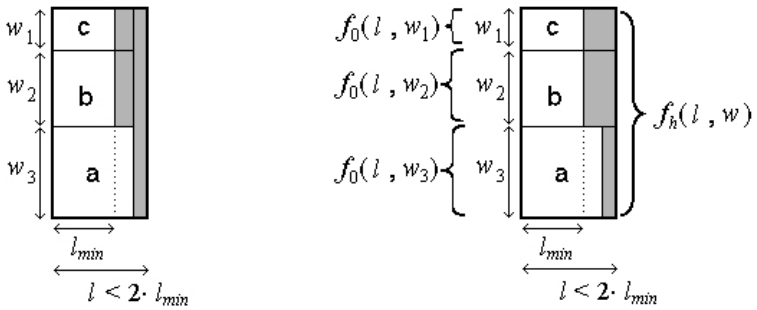
$$f^*(l,w) = f(l^*, w^*) \tag{4.5}$$

La relazione (4.2) corrisponde alla relazione (4.1). Nella relazione (4.3), se $l \geq 2 \cdot l_{min}$, allora sono considerati tutti i possibili modi di tagliare un rettangolo $R(l, w)$ con l'utilizzo di un taglio verticale alla coordinata orizzontale normalizzata x, in due sotto-rettangoli $R_1(x, w)$ ed $R_2(l - x, w)$. Il valore $(l - x)$ potrebbe non essere normalizzato, cioè potrebbe non corrispondere ad una lunghezza normalizzata, per cui va utilizzata la notazione f^* al posto di f, dove $f^*(l - x, w)$, secondo la definizione (4.5), è pari al valore di f associato al massimo sotto-rettangolo contenuto in R_2 e con dimensioni corrispondenti a coordinate normalizzate. Infatti, se $(l - x)$ non è normalizzata, allora $(l - x)^* < (l - x)$ ed il miglior valore associato al rettangolo $R_2(l - x, w)$ corrisponde ad una soluzione che ha un taglio verticale alla coordinata orizzontale normalizzata $(l - x)^*$, che lascia alla sua destra un terzo sotto-rettangolo vuoto. Dunque, il miglior valore, fra quelli associati agli schemi di taglio per $R(l, w)$ con un taglio verticale alla coordinata x, è ottenibile sommando al valore $f(x, w)$ il valore $f((l - x)^*, w)$ che, secondo la definizione di f^*, è uguale ad $f^*(l - x, w)$, dato che w è normalizzata e quindi $w^* = w$.

La condizione $x \leq \lfloor l/2 \rfloor$ ha lo scopo di evitare differenti ma simmetrici modi di ottenere il valore $f_v(l, w)$. L'utilizzo di tale condizione ha implicato la necessità di ipotizzare che valesse la relazione $l \geq 2 \cdot l_{min}$, altrimenti non esisterebbe alcun possibile x ed il valore di $f_v(l, w)$ viene allora settato a 0.

Per rettangoli $R(l, w)$ tali che $l < 2 \cdot l_{min}$, la formulazione esclude soluzioni come quelle mostrate in Figura 4.2a, dove a destra di un taglio verticale, che taglia l'intero rettangolo, c'è un sotto-rettangolo vuoto. Infatti, se $l < 2 \cdot l_{min}$, il valore $f_v(l, w)$ viene settato a zero. Tuttavia, ciò non genera incompletezza poiché i seguenti due sottocasi sono possibili:

- $w < 2 \cdot w_{min}$: in tal caso, non più di un pezzo può esssere estratto dal rettangolo R, per cui il valore $f_0(l, w)$ utilizzato nella relazione (4.2) garantisce la completezza della formulazione,
- $w \geq 2 \cdot w_{min}$: in questo caso, se la soluzione esclusa ha più di un pezzo, allora una equivalente soluzione normalizzata esiste per R, con tagli normalizzati orizzontali che identificano sotto-rettangoli da cui solo un pezzo viene estratto, come nel caso di Figura 4.2b. Quindi, il valore $f_h(l, w)$ utilizzato nella relazione (4.2) garantisce la completezza.



a) soluzione trascurata b) soluzione considerata

Figura 4.2 Knapsack function e soluzioni normalizzate

Considerazioni analoghe valgono per la relazione (4.4).

Come conseguenza della nuova formulazione, il valore $f^*(l, w)$ rappresenta un upper bound per il problema parziale associato ad $R(l, w)$, anche se l e w sono coordinate non normalizzate. Quindi il valore $f^*(L_S, W_S)$ rappresenta un upper bound per il problema associato allo stock S ed è pari, per definizione di f^*, ad $f(L^*_S, W^*_S)$, dove L^*_S e W^*_S sono i massimi elementi rispettivamente di $N_v(L_S, W_S)$ e di $N_h(L_S, W_S)$.

Un esempio del miglioramento è dato attraverso la seguente piccola istanza: uno stock quadrato con $L_S = W_S = 4$ e solo un tipo di pezzo, anch'esso quadrato, con $l_1 = w_1 = 2$, $d_1 = 1$ e $\pi_1 = 2$. E' chiaro che, utilizzando la classica knapsack function, si ottiene per l'intero stock il valore di upper bound $f(4, 4) = 8$, corrispondente ad una soluzione con quattro pezzi. Gli insiemi di coordinate normalizzate associate alla domanda limitata sono $N_v(4, 4) = N_h(4, 4) = \{2\}$ per cui, utilizzando le relazioni (4.2)-(4.5), si ottiene $f^*(4, 4) = 2$, corrispondente ad una soluzione che permette di estrarre un solo pezzo, ed è anche la soluzione fattibile ottima per la data istanza.

Si noti che in tal caso $f_{1D}(4, 4) = 2$, vale a dire $UB(4, 4)$ non cambia sebbene $f(4, 4)$ si riduca. Se però si aggiunge un secondo tipo di pezzo t_2 quadrato, con $l_2 = w_2 = 3$, $d_2 = 1$ e $\pi_2 = 1$, si ottiene $f_{1D}(4, 4) = 3$ per cui il valore dell'upper bound $UB(4, 4)$, pari al minimo fra $f(4, 4)$ ed $f_{1D}(4, 4)$, viene migliorato da 3 a 2. Una cosa simile avviene per gli esempi che verranno mostrati nei seguenti sotto-paragrafi 4.3.2, 4.3.3 e 4.3.4. Anch'essi possono essere facilmente e leggermente modificati per ottenere una istanza per la quale il valore $UB(L_S, W_S)$ è ridotto, mentre per la corrispondente istanza iniziale migliora solo il valore $f^*(L_S, W_S)$. Per brevità, si mostrerà esplicitamente come modificare l'istanza iniziale solo nell'ultimo esempio, mostrato nel paragrafo 4.3.4.

Si sottolinea che se, per ciascun tipo t_i, vale $d_i \geq \max\{\lfloor L_S / l_i \rfloor, \lfloor W_S / w_i \rfloor\}$, allora non vi è alcun miglioramento poiché, in tal caso, le coordinate normalizzate corrispondono a quelle del caso di domanda illimitata. Questa limitazione permarrà anche per i miglioramenti spiegati nei paragrafi che seguono.

4.3.2 Coordinate normalizzate compatibili

Un secondo passo può essere introdotto al fine di incrementare l'insieme di schemi di taglio esclusi, attraverso la riduzione, per ogni coppia (l, w) di coordinate normalizzate, del numero di somme realizzate nella computazione dei termini parziali $f_v(l, w)$ ed $f_h(l, w)$.

L'idea consiste, nel calcolo di $f_v(l, w)$ ($f_h(l, w)$), nell'utilizzare non l'intero insieme $N_v(L_S, W_S)$ ($N_h(L_S, W_S)$) bensì un insieme ridotto che tenga conto dei valori di l e w. Informalmente, ciò può essere ottenuto, per esempio per $f_v(l, w)$, utilizzando solo coordinate normalizzate orizzontali non maggiori di l e, contemporaneamente, compatibili con l'altra dimensione w, cioè ottenibili come somma di lunghezze dei pezzi disponibili con ampiezza non maggiore di w. Inoltre, se la stessa lunghezza l non è ottenibile attraverso una somma di questo tipo, $f_v(l, w)$ dovrebbe essere settata pari a zero. Tale obiettivo può essere raggiunto utilizzando l'insieme $N_v(l, w) = \{x \le l : g_{v,n}(x) \le w\}$ definito nel paragrafo 4.2.1. La definizione di coordinate normalizzate viene allora ad essere raffinata, associando a ciascun $w < W_S$ un insieme ridotto di coordinate per i possibili tagli verticali e per le possibili lunghezze, secondo l'insieme di tipi di pezzo che hanno ampiezza non maggiore di w, e, a ciascun $l < L_S$, un insieme ridotto di coordinate per i possibili tagli orizzontali e per le possibili ampiezze, secondo l'insieme di tipi di pezzo che hanno lunghezza non maggiore di l.

Per esempio, si consideri un caso con uno stock avente dimensioni $L_S = 6$ e $W_S = 2$, e due tipi di pezzo. Il tipo t_1 ha dimensioni $l_1 = 4$, $w_1 = 2$, domanda $d_1 = 1$ e profitto $\pi_1 = 3$ ed il tipo t_2 ha dimensioni $l_2 = 2$, $w_2 = 1$, domanda $d_2 = 1$ e profitto $\pi_2 = 1$. L'insieme di coordinate normalizzate orizzontali $N_v(6, 2)$ associate all'intero stock é {2, 4, 6}. Tuttavia è chiaro che, per un qualsiasi rettangolo R con ampiezza 1, la massima lunghezza utilizzabile vale 2, poiché solo un pezzo del secondo tipo può essere contenuto in R. Analogamente, l'insieme di coordinate verticali normalizzate $N_h(6, 2)$ associato all'intero stock è {1, 2}, ma, per un qualsiasi rettangolo R con lunghezza 2, la massima ampiezza utilizzabile è 1, poiché solo un pezzo del secondo tipo può essere contenuto in R.

Per tali motivi, utilizzando le relazioni (4.2)-(4.5), si ottiene $f(2, 2) = f_h(2, 2) = 2$ attraverso la somma "verticale" $f(2, 1) + f(2, 1)$, dove $f(2, 1) = f_0(2, 1) = 1$, ed $f(4, 1) = f_v(4, 1) = 2$ attraverso la equivalente somma "orizzontale" $f(2, 1) + f(2, 1)$. Si ottiene dunque $f(4, 2) = f_v(4, 2) = f_h(4, 2) = 4$, maggiore di $f_0(4, 2) = 3$, dove $f_v(4, 2)$ ed $f_h(4, 2)$ sono rispettivamente ottenuti con la somma $f(2, 2) + f(2, 2)$ e con la somma $f(4, 1) + f(4, 1)$. Si ottiene anche $f(6, 1) = f_v(6, 1) = 3$ con la somma $f(2, 1) + f(4, 1)$. Per l'intero stock, si ottiene $f(6, 2) = f_v(6, 2) = f_h(6, 2) = 6$, corrispondente ad una soluzione con sei pezzi del secondo tipo, dove $f_v(6, 2)$ ed $f_h(6, 2)$ sono rispettivamente ottenuti con la somma $f(2, 2) + f(4, 2)$ e con la somma $f(6, 1) + f(6, 1)$.

Al contrario, utilizzando gli insiemi ridotti di coordinate normalizzate compatibili, la somma "orizzontale" $f(2, 1) + f(2, 1)$ e le somme "verticali" $f(2, 1) + f(2, 1)$ ed $f(2, 1) + f(4, 1)$ non sono più realizzate e quindi si ottiene $f(2, 2) = f_0(2, 2) = 1$ ed $f(6, 1) = f(4, 1) = f_0(4, 1) = 1$. Di conseguenza, si ottiene $f(4, 2) = f_0(4, 2) = 3$ e, per l'intero stock, $f(6, 2) = f_v(6, 2) = 4$ con la somma $f(2, 2) + f(4, 2)$, corrispondente ad una soluzione con un pezzo di entrambi i tipi, e cioè la soluzione fattibile ottima.

Per formalizzare il miglioramento, sono necessarie delle definizioni preliminari. Sia IW l'insieme di ampiezze dei tipi di pezzo, e siano $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_{|IW|}$ i suoi elementi in ordine crescente, dove $|IW|$, cardinalità dell'insieme IW, è minore di n se più tipi condividono la stessa ampiezza, ed uguale ad n se tutti i tipi hanno ampiezze differenti. E' necessario definire anche un valore fittizio, denotato con $\underline{w}_{|IW|+1}$ e pari a $W_S + 1$.

E' sufficiente calcolare solo gli insiemi $N_v(L_S, \underline{w}_j)$, con $j \in \{1, .., |IW|\}$. Infatti, per una qualunque coppia (l, w), vale la relazione $N_v(l, w) = N_v(l, \underline{w}_k)$, dove $k = \max\{j : \underline{w}_j \le w\}$. In

pratica, per qualsiasi coordinata orizzontale normalizzata l, per qualunque $j \in \{1, \ldots, |IW|\}$ e per qualunque coordinata verticale w contenuta nell'intervallo $[\underline{w}_j, \underline{w}_{j+1}[$, l'insieme $N_v(l, w)$ è uguale ad $N_v(l, \underline{w}_j)$ poiché, rispetto al caso di $N_v(l, \underline{w}_j)$, nessun ulteriore tipo può essere usato per combinazioni lineari di lunghezze.

E' facile mostrare che $N_v(l, \underline{w}_j) = \{x \le l : x \in N_v(L_S, \underline{w}_j)\}$. Inoltre vale la relazione $N_v(l, \underline{w}_j) \subseteq N_v(l, \underline{w}_{j+1})$ per ogni $j = 1, 2, \ldots, |IW| - 1$.

Simili definizioni dell'insieme IL di lunghezze dei tipi, e degli elementi $\underline{l}_1, \underline{l}_2, \ldots, \underline{l}_{|IL|}$, supportano la relazione $N_h(l, w) = \{y \le w : y \in N_h(\underline{l}_k, W_S)\}$, dove $k = \max\{j : \underline{l}_j \le l\}$, e la relazione $N_h(\underline{l}_j, w) \subseteq N_h(\underline{l}_{j+1}, w)$ per $j = 1, 2, \ldots, |IL| - 1$. Quindi le relazioni (4.2)-(4.4) possono essere sostituite dalla seguente formulazione rivisitata:

$$f(l,w) = \begin{cases} 0 & \text{if} \quad l = 0 \quad \text{or} \quad w = 0 \\ \max\left\{f_0(l,w),\, f_v(l,w),\, f_h(l,w),\, f^*(l-1,w),\, f^*(l,w-1)\right\} & \text{else} \end{cases} \tag{4.6}$$

$$f_v(l,w) = \begin{cases} 0 & \text{if} \quad l < 2 \cdot l^w_{\min} \quad \text{or} \quad l \notin N_v(L_S, w) \\ \max\limits_{\substack{x \in N_v(L_S,w) \\ x \le \lfloor l/2 \rfloor}} \left\{f(x,w) + f^*(l-x,w)\right\} & \text{else} \end{cases} \tag{4.7}$$

$$f_h(l,w) = \begin{cases} 0 & \text{if} \quad w < 2 \cdot w^l_{\min} \quad \text{or} \quad w \notin N_h(l, W_S) \\ \max\limits_{\substack{y \in N_h(l,W_S) \\ y \le \lfloor w/2 \rfloor}} \left\{f(l,y) + f^*(l,w-y)\right\} & \text{else} \end{cases} \tag{4.8}$$

$$\text{dove} \quad l \in N_v(W_S, L_S) \quad \text{ed} \quad w \in N_h(W_S, L_S)$$

I due valori l^w_{min} e w^l_{min} sono i minimi elementi positivi di $N_v(L_S, w)$ e di $N_h(l, W_S)$. Essi giocano un ruolo simile a quello di l_{min} e w_{min} nelle relazioni (4.3) e (4.4).

La posizione $f_v(l, w) = 0$ nella relazione (4.7) quando $l \notin N_v(L_S, w)$ è applicata poiché, in questo caso, nessuna soluzione normalizzata per il rettangolo $R(l, w)$ può pienamente sfruttare l'intera lunghezza l. Una considerazione simile vale per la relazione (4.8). Si noti tuttavia che le soluzioni normalizzate per R che non sfruttano pienamente la lunghezza l o l'ampiezza w, possono dover essere considerate per non trascurare soluzioni normalizzate per rettangoli che contengono R. Nell'esempio precedente valeva $f(2, 2) = f_0(2, 2) = f(2, 1) = 1$, poiché solo un pezzo era coinvolto. In generale, per calcolare un upper bound valido, è necessario imporre che valga la relazione $f(l, w) \ge \max\{ f^*(l - 1, w), f^*(l, w - 1) \}$. Quindi l'equazione (4.6) consta di due termini in più, rispetto alla relazione (4.2), vale a dire i due valori $f^*(l - 1, w)$ ed $f^*(l, w - 1)$.

Si noti inoltre che lo l'utilizzo di questi due valori permette di ridefinire $f_0(l, w)$ come il migliore (massimo) fra i valori delle soluzioni per $R(l, w)$ che estraggono un solo pezzo che abbia le stesse dimensioni di R, cioè senza utilizzare alcun taglio. In pratica, se almeno un tipo di pezzo ha dimensioni (l, w), allora vale $f_0(l, w) = \max\limits_{i=1,\ldots,n} \{\pi_i : l_i = l \text{ and } w_i = w\}$, altrimenti esso è pari a zero. Questa nuova definizione facilita la computazione del termine $f_0(l, w)$.

Nella relazione (4.7), la condizione $x \in N_v(L_S, w)$ viene utilizzata al posto della condizione $x \in N_v(l, w)$, poiché quando l'altra condizione $x \le \lfloor l/2 \rfloor$ è soddisfatta, le due diventano equivalenti, secondo la definizione dell'insieme $N_v(l, w)$ data sopra. Un ragionamento simile vale per la relazione (4.8).

4.3.3 Coppie di coordinate normalizzate compatibili

Il passo migliorativo descritto in questa sezione ha l'obiettivo di aumentare ulteriormente l'insieme di schemi di taglio esclusi, ancora attraverso la riduzione, per una qualunque coppia (l, w) di coordinate normalizzate, del numero di somme realizzate per il calcolo dei valori $f_v(l, w)$ ed $f_h(l, w)$.

L'idea consiste, per esempio nel calcolare $f_v(l, w)$, nel considerare solo le coordinate orizzontali normalizzate x tali che anche $l - x$ sia una coordinata orizzontale normalizzata compatibile con l'ampiezza w.

Per mostrare come questa idea possa determinare un miglioramento, si fornisce il seguente esempio. Si supponga di avere uno a stock con dimensioni $L_S = 7$, $W_S = 1$ e due tipi di pezzo, rispettivamente t_1 con $l_1 = 2$, $w_1 = 1$, $d_1 = 2$ e $\pi_1 = 1$, e t_2 con $l_2 = 7$, $w_2 = 1$, $d_2 = 1$, $\pi_2 = 1$. In tal caso, $N_v(7, 1) = \{2, 4, 7\}$ per cui, utilizzando le relazioni (4.6)-(4.8), si ottiene $f(2, 1) = f_0(2, 1) = 1$, $f(4, 1) = f_v(4, 1) = f(2, 1) + f(2, 1) = 2$ e, per l'intero stock, $f(7, 1) = f_v(7, 1) = f(2, 1) + f^*(5, 1) = f(2, 1) + f(4, 1) = 3$, corrispondente ad una soluzione non fattibile con tre pezzi del primo tipo.

Secondo la nuova idea, nel calcolo del termine $f_v(7, 1)$, non andrebbe realizzata la somma $f(2, 1) + f^*(5, 1)$, poiché 5 non è una coordinata orizzontale normalizzata, per cui nessuna soluzione fattibile normalizzata può pienamente sfruttare la lunghezza 5 a destra di un qualsiasi taglio verticale. Di conseguenza si ottiene, per l'intero stock, $f(7, 1) = f^*(6, 1) = f(4, 1) = 2$, corrispondente ad una soluzione con due soli pezzi, che è anche la soluzione fattibile ottima.

Per formalizzare il raffinamento delle relazioni che danno i valori di $f_v(l, w)$ ed $f_h(l, w)$, si definisce $N^+_v(l, w)$ come l'insieme di valori x positivi, appartenenti ad $N_v(l, w)$ e tali che anche $l - x$ è positivo ed appartiene ad $N_v(l, w)$, cioè $N^+_v(l, w) = \{x \in N_v(l, w) : 0 < x < l \text{ ed } l - x \in N_v(l, w)\}$.

Similmente, si definisce $N^+_h(l, w) = \{y \in N_h(l, w) : 0 < y < w \text{ e } w - y \in N_h(l, w)\}$.

Secondo le definizioni date nel precedente paragrafo, si ricavano le due uguaglianze $N^+_v(l, w) = N^+_v(l, \underline{w}_k) = \{x \in N_v(L_S, \underline{w}_k) : 0 < x < l \text{ ed } l - x \in N_v(L_S, \underline{w}_k)\}$, ed $N^+_h(l, w) = N^+_h(\underline{l}_k, w) = \{y \in N_h(\underline{l}_k, W_S) : 0 < y < w \text{ e } w - y \in N_v(\underline{l}_k, W_S)\}$, dove, per la prima, $k = \max\{j : \underline{w}_j \le w\}$ e, per la seconda, $k = \max\{j : \underline{l}_j \le l\}$. Inoltre vale $N^+_v(l, \underline{w}_j) \subseteq N^+_v(l, \underline{w}_{j+1})$ per $j = 1, 2, \ldots, |IW| - 1$ ed infine vale $N^+_h(\underline{l}_j, w) \subseteq N^+_h(\underline{l}_{j+1}, w)$ per $j = 1, 2, \ldots, |IL| - 1$.

Quindi, le relazioni (4.7) ed (4.8) possono essere sostituite dalle seguenti:

$$f_v(l, w) = \begin{cases} 0 & \text{if} \quad l < 2 \cdot l^w_{\min} \quad \text{or} \quad N^+_v(l, w) = \varnothing \\ \max\limits_{\substack{x \in N^+_v(l,w) \\ x \le \lfloor l/2 \rfloor}} \{f(x, w) + f(l - x, w)\} & \text{else} \end{cases} \tag{4.9}$$

$$f_h(l, w) = \begin{cases} 0 & \text{if} \quad w < 2 \cdot w^l_{\min} \quad \text{or} \quad N^+_h(l, w) = \varnothing \\ \max\limits_{\substack{y \in N^+_h(l,w) \\ y \le \lfloor w/2 \rfloor}} \{f(l, y) + f(l, w - y)\} & \text{else} \end{cases} \tag{4.10}$$

Si noti che, nella relazione (4.9), si è utilizzato $f(l - x, w)$ al posto di $f^*(l - x, w)$. Ciò è motivato dal fatto che anche $(l - x)$ appartiene ad $N_v(l, w)$ per cui i due valori sono uguali. Una analoga considerazione vale per il valore $f(l, w - y)$ nella relazione (4.10).

4.3.4 Coordinate normalizzate raggiungibili

L'ulteriore passo migliorativo descritto nel seguito è rivolto a ridurre ulteriormente l'insieme di schemi di taglio esclusi. Come nei passi dei precedenti paragrafi, ciò è ottenuto riducendo il numero di somme realizzate nella computazione dei due valori $f_v(l, w)$ ed $f_h(l, w)$ per le coppie (l, w) di coordinate normalizzate.

Per esempio, nel calcolare $f_v(l, w)$, si considerano solo coppie di lunghezze x ed $l - x$ tali che $0 < x < l$ e che esistano due cutting patterns $B^x = (b^x_1, \ldots, b^x_n)$ e $B^l = (b^l_1, \ldots, b^l_n)$, soddisfacenti le quattro seguenti condizioni:

- $x = \sum_{i=1}^{n} \left(b_i^x \cdot l_i \right)$
- $l = \sum_{i=1}^{n} \left(b_i^l \cdot l_i \right)$
- $b^x_i \le b^l_i \le d_i$ per $i = 1, \ldots, n$
- per ogni $i = 1, \ldots, n$, se $w_i > w$ allora $b^l_i = b^x_i = 0$

Quando valgono tutte queste quattro condizioni, allora significa che si può ottenere sia l come combinazione lineare di lunghezza dei tipi di pezzo, utilizzando solo i pezzi estraibili da un rettangolo $R(l, w)$ e tenendo conto dei vincoli di domanda, sia il valore x, positivo ed inferiore ad l, come somma delle lunghezze di un sottoinsieme di pezzi coinvolti nella combinazione lineare relativa al valore l.

In questo caso, si può dire che la coordinate normalizzata orizzontale l è *raggiungibile* dalla coordinata normalizzata orizzontale x, rispetto all'ampiezza w oppure, equivalentemente, che x è *osservabile* da l rispetto a w. Si noti che potrebbero esistere differenti coppie B^x, B^l per le quali siano soddisfatte le quattro suddette proprietà. Si definisce con $RN_v(w)$ l'insieme delle coppie (x, l) per le quali esista almeno una coppia buona di cutting patterns, rispetto all'ampiezza w. Analogamente, si definiscono le coordinate normalizzate verticali raggiungibili e gli insiemi $RN_h(l)$.

Si denota con $N^*_v(l, w)$ l'insieme di coordinate normalizzate orizzontali x appartenenti all'insieme $N_v(l, w)$ e tali che l sia raggiungibile da x rispetto a w. Ciò vuol dire che vale l'espressione $N^*_v(l, w) = \{x : (x, l) \in RN_v(w)\}$. Inoltre, secondo le definizioni date, è chiaro che se l è raggiungibile da x rispetto a w, allora anche il valore $l - x$ è una coordinata normalizzata orizzontale compatibile con w, essendo ottenibile dalla combinazione lineare $\sum_{i=1}^{n} \left[\left(b^l_i - b^x_i \right) l_i \right]$ ed appartenente quindi ad $N^*_v(l, w)$, dove i b^l_i ed i b^x_i sono gli elementi di una possibile coppia di cutting patterns B^x, B^l associata alla coppia di lunghezze (x, l). Si può quindi affermare che l'insieme $N^*_v(l, w)$ è incluso nell'insieme $N^+_v(l, w)$, definito nel precedente paragrafo. In modo simile, per una qualunque coppia (l, w), si definisce l'insieme $N^*_h(l, w) = \{y : (y, w) \in RN_h(l)\}$, incluso nell'insieme $N^+_h(l, w)$.

Si mostra nel seguito un esempio in cui, utilizzando le coordinate normalizzate raggiungibli, si ottiene un upper bound migliorato, cioè con valore inferiore. Si supponga di avere due tipi di pezzo, t_1 con $l_1 = 2$, $w_1 = 1$, $d_1 = 2$, $\pi_1 = 1$ e t_2 con $l_2 = 6$, $w_2 = 1$, $d_2 = 1$, $\pi_2 = 1$, ed uno stock con dimensioni $L_S = 6$, $W_S = 1$. In tal caso $N_v(6, 1) = \{2, 4, 6\}$, $N^+_v(6, 1) = \{2, 4\}$ ed $N^*_v(6, 1) = \varnothing$.

Utilizzando le coppie di coordinate normalizzate compatibili del precedente paragrafo, si ottiene $f(4, 1) = f_v(4, 1) = f(2, 1) + f(2, 1) = 2$, dove $f(2, 1) = f_0(2, 1) = 1$. Si ottiene inoltre $f(6, 1) = f_v(6, 1) = f(2, 1) + f(4, 1) = 3$, corrispondente, per l'intero stock, ad una soluzione non fattibile con tre pezzi del primo tipo. Questa somma è realizzata grazie al fatto che sia la lunghezza 2 che la lunghezza 4 sono compatibili con l'ampiezza 1. Al contrario, secondo la

nuova idea delle coordinate normalizzate raggiungibili, tale somma non dovrebbe essere realizzata, poiché nessuna soluzione normalizzata fattibile, per rettangoli con ampiezza 1, può sfruttare pienamente sia la lunghezza 2 a sinistra che la lunghezza 4 a destra di un taglio verticale alla coordinata 2.

Infatti, la lunghezza 6 non è raggiungibile dalla lunghezza 2, rispetto all'ampiezza 1, per cui la somma $f(2, 1) + f(4, 1)$ non viene realizzata se si sfrutta la nuova idea. Di conseguenza, si ottiene $f(6, 1) = f^*(5, 1) = f(4, 1) = 2$, corrispondente ad una soluzione con due soli pezzi, che è anche la soluzione fattibile ottima.

La formulazione rivisitata si ottiene sostituendo $N^*_v(l, w)$ ad $N^+_v(l, w)$ nella relazione (4.9) ed $N^*_h(l, w)$ ad $N^+_h(l, w)$ nella relazione (4.10). Si raffina inoltre la relazione (4.6), sostituita dalla (4.11).

$$f(l,w) = \begin{cases} 0 & \text{if} \quad l = 0 \quad \text{or} \quad w = 0 \\ \max\left\{f^*(l-1,w),\, f^*(l,w-1)\right\} & \text{if} \quad \begin{array}{c} l \notin N_v(L_S, w) \ \text{or} \\ w \notin N_h(l, W_S) \end{array} \\ \max\left\{f_0(l,w),\, f_v(l,w),\, f_h(l,w),\, f^*(l-1,w),\, f^*(l,w-1)\right\} & \text{else} \end{cases} \tag{4.11}$$

$$f_v(l,w) = \begin{cases} 0 & \text{if} \quad l < 2 \cdot l^w_{\min} \quad \text{or} \quad N^*_v(l,w) = \varnothing \\ \max\limits_{\substack{x \in N^*_v(l,w) \\ x \le \lfloor l/2 \rfloor}} \left\{f(x,w) + f(l-x,w)\right\} & \text{else} \end{cases} \tag{4.12}$$

$$f_h(l,w) = \begin{cases} 0 & \text{if} \quad w < 2 \cdot w^l_{\min} \quad \text{or} \quad N^*_h(l,w) = \varnothing \\ \max\limits_{\substack{y \in N^*_h(l,w) \\ y \le \lfloor w/2 \rfloor}} \left\{f(l,y) + f(l,w-y)\right\} & \text{else} \end{cases} \tag{4.13}$$

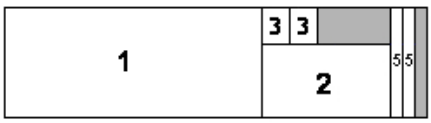
La relazione (4.11) rappresenta un miglioramento in termini di tempo computazionale, rispetto alla relazione (4.6). Se, per esempio, $w \in N_v(l, W_S)$ ma $l \notin N_v(L_S, w)$, allora $f_v(l, w)$ viene settata a zero attraverso la relazione (4.12) e si può anche affermare che nessun pezzo ha dimensioni (l, w). Quindi si ha anche $f_0(l, w) = 0$, secondo la nuova definizione di f_0 data alla fine del paragrafo 4.3.2. Inoltre, una qualunque soluzione associata ad una somma realizzata nel calcolo di $f_h(l, w)$ è simile a quella già considerata nel paragrafo 4.3.1 e mostrata in Figura 4.2b, vale a dire equivalente ad una soluzione che utilizza un taglio verticale che lascia sulla sua destra un sotto-rettangolo vuoto, come in Figura 4.2a. Ciò vuol dire che il termine $f^*(l - 1, w)$ è già sufficiente ad evitare di trascurare questo tipo di soluzioni. Il miglioramento consiste nel fatto che si può allora evitare il calcolo di $f_h(l, w)$.

Nel paragrafo 4.3.2 si è mostrato che, per un qualsiasi valore di lunghezza l, per un qualunque indice $j \in \{1, 2, \ldots, |IW|\}$ e per qualunque valore di ampiezza w contenuto nell'intervallo $[\underline{w}_j, \underline{w}_{j+1}[$, vale l'uguaglianza $N_v(l, w) = N_v(l, \underline{w}_j)$. Da ciò si può concludere che non è necessario costruire l'insieme $N^*_v(l, w)$ per tutte le coppie (l, w) di coordinate normalizzate, ma solo per le coppie $(l, \underline{w}_j)$, dove l è una coordinata normalizzata orizzontale e $j \in \{1, 2, \ldots, |IW|\}$. Analogamente, è sufficiente costruire solo gli insiemi $N^*_h(\underline{l}_j, w)$, per tutte le coordinate normalizzate verticali w e per tutti i valori $j \in \{1, 2, \ldots, |IL|\}$.

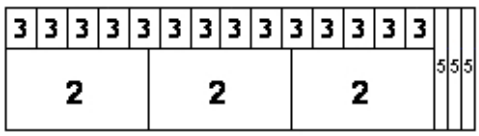
Si noti anche che, come per gli insiemi N_v, N_v^+, N_h ed N_h^+, valgono le due seguenti relazioni: $N^*_v(l, \underline{w}_j) \subseteq N^*_v(l, \underline{w}_{j+1})$ per $j = 1, 2, \ldots, |IW| - 1$ ed $N^*_h(\underline{l}_j, w) \subseteq N^*_h(\underline{l}_{j+1}, w)$ per $j = 1, 2, \ldots, |IL| - 1$.

Si noti infine che se, per una data ampiezza w, le due coppie di lunghezze (l_1, l_2) ed (l_2, l_3) appartengono ad $RN_v(w)$, non è certo che anche (l_1, l_3) vi appartenga, il che significa che non vale, in tal senso, alcuna proprietà transitiva, che varrebbe in caso di domanda di pezzi illimitata. Per esempio, si supponga, per una data ampiezza, di avere a disposizione quattro tipi di pezzi, con lunghezze rispettivamente 2, 3, 5 ed 8, tutti con domanda 1. In tal caso, 5 è raggiungibile da 2, poiché 2 + 3 = 5 ed 8 è raggiungibile da 5, poiché 5 + 3 = 8, ma 8 non è raggiungibile da 2, poiché un solo pezzo è disponibile per il tipo con lunghezza 3. Una simile proprietà vale per gli insiemi $RN_h(l)$.

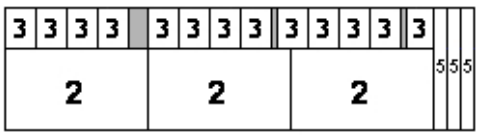
Nel seguito si fornisce un esempio riassuntivo attraverso cui si mostra come ciascun passo migliorativo di quelli descritti fino ad ora permetta di ottenere un valore ridotto dell'upper bound rispetto al passo precedente, in relazione all'intero stock. Questo esempio è stato ottenuto combinando i piccoli esempi mostrati nei paragrafi precedenti. Si consideri dunque il caso in cui lo stock abbia dimensioni $L_S = 87$ e $W_S = 3$, con $n = 6$ tipi di pezzi. I primi due tipi giocano il ruolo dei due tipi dell'esempio mostrato nel paragrafo 4.3.2. Il valore dei parametri associati sono $l_1 = 56$, $w_1 = 3$, $d_1 = 1$, $\pi_1 = 131$ ed $l_2 = 28$, $w_2 = 2$, $d_2 = 1$, $\pi_2 = 57$. Altri due tipi giocano il ruolo di quelli dell'esempio dato nel paragrafo 4.3.3, con $l_3 = 6$, $w_3 = 1$, $d_3 = 2$, $\pi_3 = 2$ ed $l_4 = 25$, $w_4 = 1$, $d_2 = 1$, $\pi_2 = 2$. Gli ultimi due tipi giocano il ruolo di quelli dell'esempio dato precedentemente in questo stesso paragrafo, con $l_5 = 1$, $w_5 = 3$, $d_5 = 2$, $\pi_5 = 2$ ed $l_6 = 3$, $w_6 = 3$, $d_6 = 1$, $\pi_6 = 2$. Tutti i parametri indicati sono stati opportunamente scelti in modo che ciascun passo migliorativo comporti un effetto simile a quello mostrato nel corrispondente esempio. A tal fine, è stato necessario modificare i valori dei parametri rispetto al relativo esempio, evitando che ciascuna coppia di tipi potesse interferire con gli effetti relativi alle altre due coppie. La soluzione fattibile ottima è mostrata in Figura 4.3a, ed ha un valore di profitto totale pari a 196.



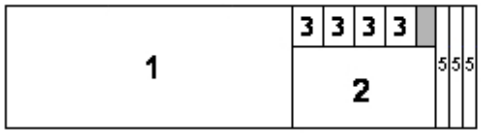
a) soluzione fattibile ottima



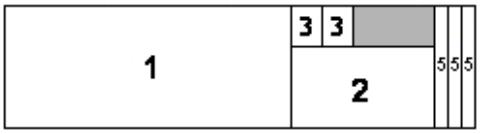
b) knapsack function classica



c) coordinate normalizzate



d) coordinate normalizzate compatibili



e) coppie di coordinate normalizzate

Figura 4.3 Esempio riassuntivo

Utilizzando la formulazione classica della knapsack function, si ottiene la soluzione non fattibile mostrata in Figura 4.3b, con valore di profitto pari a 205. Con il primo passo migliorativo, che sfrutta le coordinate normalizzate, la soluzione diventa quella non fattibile mostrata in Figura 4.3c, con valore di profitto pari a 203. Lo sfruttamento delle coordinate normalizzate compatibili conduce ad ottenere la soluzione non fattibile mostrata in Figura 4.3d, con valore di profitto pari a 202. Le coppie di coordinate normalizzate compatibili permettono invece di ottenere la soluzione non fattibile mostrata in Figura 4.3e, con valore di profitto pari a 198. Infine, utilizzando le coordinate normalizzate raggiungibili, è possibile

ottenere un valore di upper bound uguale all'ottimo ed associato allo schema di taglio della soluzione fattibile ottima già mostrata in Figura 4.3a. Si noti che, per questa istanza, l'upper bound mono-dimensionale $f_{1D}(87, 3)$ è uguale a 198. Tuttavia è possibile costruire una istanza leggermente differente, aumentando tutte le ampiezze di un fattore 10 ed aggiungendo un settimo tipo di pezzi t_7 con $l_7 = 87$, $w_7 = 1$, $d_7 = 1$ e $\pi_7 = 7$, ottenendo così $f_{1D}(87, 30) = 205$ mentre, per ciascun passo migliorativo, si ottiene lo stesso valore di knapsack function rispetto a prima, tranne per il caso classico, per il quale si passa dal valore 205 al valore $f(87, 30) = 241$.

4.3.5 Miglioramento dell'upper bound mono-dimensionale

Come conseguenza della definizione di coordinate normalizzate, nessuna soluzione può utilizzare una porzione dello stock più grande dell'area $L^*_S \cdot W^*_S$, secondo la definizione di L^*_S ed W^*_S data nel paragrafo 4.3.1. Quindi, si può usare il valore $f_{1D}(L^*_S, W^*_S)$, al posto di $f_{1D}(L_S, W_S)$, come termine mono-dimensionale dell'upper bound $UB(L_S, W_S)$ associato all'intero stock e prima definito con l'espressione $\min\{ f(L_S, W_S), f_{1D}(L_S, W_S) \}$.

Per esempio, se si ha un problema pesato con due tipi di pezzo quadrati, ed entrambi hanno dimensioni $l = w = 2$, rispettivamente con $d_1 = 1$, $\pi_1 = 2$ e $d_2 = 4$, $\pi_2 = 1$ ed uno stock quadrato con dimensioni $L_S = W_S = 5$, allora $L^*_S = W^*_S = 4$ ed $f_{1D}(L^*_S, W^*_S) = 5$, mentre $f_{1D}(L_S, W_S) = 6$, cioè la somma dei profitti di tutti i pezzi disponibili, poiché la somma delle loro aree è inferiore all'area dello stock. Inoltre si ottiene $f^*(L_S, W_S) = 8$, a causa della equivalenza geometrica dei due tipi di pezzo, per cui l'upper bound $UB(L_S, W_S)$ è ridotto da 6 a 5. Lo stesso miglioramento può essere utilizzato per calcolare un upper bound con valore ridotto per rettangoli $R(l, w)$, tali che almeno una fra l e w non sia una coordinata normalizzata.

Si ricorda che in letteratura l'upper bound mono-dimensionale è ottenuto alternativamente in due differenti modi. Il primo è basato su una procedura di programmazione dinamica, come in León, Miranda, Rodríguez e Segura (2007) ed il secondo utilizza una procedura greedy, come in Hifi e Zissimopoulos (1997), basato su un ulteriore rilassamento, secondo cui è permesso un numero frazionario di pezzi. La seconda alternativa genera un upper bound dominato da quello generato nel primo modo, ma necessita di un tempo di calcolo più piccolo.

León et al. hanno inoltre proposto un efficace miglioramento dei valori di knapsack function, sfruttando nella loro procedura di calcolo i valori degli upper bounds mono-dimensionali. Essi denotano con $V(A)$ l'upper bound mono-dimensionale relativo ad un generico rettangolo che ha area A, equivalente cioè al valore $f_{1D}(A)$ prima definito. Inoltre essi definiscono la knapsack function migliorata $F_V(l, w)$ come il minimo fra il valore $f(l, w)$ ed il valore $V(l \cdot w)$, ed utilizzano i valori di F_V al posto dei valori di f nel calcolo dei due termini f_v ed f_h . Di fondo, questo modo di calcolare i valori della knapsack function non causa alcun incremento del relativo tempo di calcolo. Si noti inoltre che questa idea può essere applicata anche alla knapsack function raffinata proposta in questo Capitolo. Nel seguito la tecnica di León et al. verrà denotata con il termine *bounds merging*.

4.4 Risultati computazionali

Sono stati realizzati dei test su alcune istanze della librearia Packlib[2], collezionata nel lavoro di Fekete e van der Veen (2007). Sono stati confrontati risultati ottenuti utilizzando la classica relazione ricorsiva (4.1) con quelli ottenuti attraverso le nuove relazioni (4.11)-(4.13), contenenti l'intero insieme di miglioramenti proposti. Per il primo metodo si è utilizzata la procedura di Gilmore e Gomory (1966) con tutti i dettagli implementativi descritti nel loro lavoro. I risultati sono stati ottenuti su un sistema Pentium IV 1.80 GHz, 1GB Ram, Windows XP SP2, utilizzando un codice C++.

I confronti si basano principalmente sui seguenti due parametri:

- F : il valore di knapsack function associato all'intero stock,
- CT : il tempo di calcolo totale necessario per ottenere tutti i valori della knapsack function.

I parametri associati alla formulazione classica vengono denotati con il pedice *old* e quelli associati alla nuova formulazione con il pedice *new*. Nella Tabella 4.1 si riportano anche altri sei parametri, tre relativi alla knapsack function e tre relativi al tempo di calcolo.

I primi tre sono *OPT*, ΔG e *WKF*. Il primo rappresenta il valore della soluzione fattibile ottima. Quando esso è soltanto il miglior valore noto, senza prova di ottimalità, il valore è contrassegnato da un asterisco.

Il parametro ΔG, uguale al rapporto $(F_{old} - OPT) / (F_{new} - OPT)$, rappresenta il miglioramento del gap fra l'upper bound F e l'ottimo. Il parametro *WKF* è una media pesata dei miglioramenti relativi a ciascun valore di knapsack function. Più precisamente, il valore di *WKF* è calcolato usando l'espressione $\sum_{\substack{l \in N_V(L_S, W_S) \\ w \in N_h(L_S, W_S)}} \left[\frac{f_{old}(l,w)}{f_{new}(l,w)} \cdot \frac{l \cdot w}{L_S^* \cdot W_S^*} \right]$, dove f_{old} ed f_{new} sono le knapsack functions ottenute rispettivamente nel vecchio e nel nuovo modo. Si noti che tale parametro rappresenta il miglioramento nell'utilizzo dei valori di knapsack function, come upper bounds, per ciascun possibile rettangolo contenuto nello stock.

Gli altri tre parametri sono *CTR*, T_{al} e CTR_{al}. Il parametro *CTR* è uguale al rapporto fra CT_{new} e CT_{old} , e quindi rappresenta, in termini di tempo di calcolo, il peggioramento quando maggiore di 1, ed il miglioramento, quando minore di 1. Il parametro T_{al} rappresenta il tempo di calcolo necessario per risolvere la corrispondente istanza del problema.

Infine, il parametro CTR_{al} è uguale al rapporto fra T_{al} e CT_{new} ed è espresso in termini percentuali. I tempi di calcolo *CT* sono indicati in millisecondi. Quando un valore di un tempo *CT* è inferiore ad un secondo, si riporta semplicemente il numero di millisecondi, altrimenti tutte le cifre necessarie. I tempi di calcolo T_{al} , al contrario, sono espressi in secondi.

Nella Tabella 4.1 le istanze sono arrangiate in due gruppi, secondo la loro dimensione e, per ciascun gruppo, sono separate in due sotto-gruppi secondo le due varianti, pesata (weighted) e non pesata (un-weighted), del problema.

Per le istanze A1, A2, A3, A4, A5 ed H, si riporta come T_{al} il tempo necessario per l'algoritmo esatto di Cung, Hifi e Le Cun (2000), come indicato nel loro lavoro. Per le istanze cgcut2, cgcut3, da CW1 a CW9 e da CU1 a CU9, si riporta il tempo necessario per l'algoritmo euristico GBSC di Fayard, Hifi e Zissimopoulos (1998), come indicato nel loro lavoro. Per le istanze di elevata dimensione (large size), il valore del parametro T_{al} è riportato come media per il corrispondente sotto-gruppo, come indicato nel lavoro di Hifi (2004), che ha utilizzato un apposito algoritmo euristico.

Istanza	*OPT*	*F*		ΔG	*WKF*	*CT* (msec)		*CTR*	T_{al} (sec)	CTR_{al}
		new	*old*			*new*	*old*			
Istanze pesate di dimensione medio-piccola										
A1	2020	2040	2220	10	1,140	63	16	3,938	3,31 [a]	1,9%
A2	2505	2830	3375	2,677	1,171	62	< 1	> 62	4,4 [a]	2,27%
A3	5451	5524	5562	1,521	1,035	140	16	8,75	6,57 [a]	2,1%
CW1	6402	6744	6744	1	1,003	281	31	9,065	5,31 [b]	5,3%
CW2	5354	5673	5673	1	1,004	406	78	5,205	6,83 [b]	5,9%
CW3	5689	5936	5936	1	1,001	1:047	156	6,712	17,32 [b]	6,0%
CW4	6175	6551	6551	1	1,001	2:000	594	3,367	30,84 [b]	6,5%
CW5	11659	11988	11988	1	1	2:937	1:218	2,411	19,84 [b]	14,8%
CW6	12923	13664	13664	1	1	12:469	1:844	6,762	78,97 [b]	15,8%
CW7	9898	9898	9898	-	1^+	2:031	343	5,921	61,39 [b]	3,3%
CW8	4605	5052	5125	1,163	1,003	3:000	266	11,27	107,42 [b]	2,8%
CW9	10748	10995	10995	1	1	3:016	468	6,444	125,30 [b]	2,4%
media								6,35	-	5,76%
Istanze non pesate di dimensione medio-piccola										
A4	6179	6264	6300	1,424	1,023	235	31	7,581	7,83 [a]	3%
A5	12985	13142	13174	1,204	1,204	438	47	9,319	14,52 [a]	3%
H	11586	12132	12348	1,396	1,396	47	47	1	1,85 [a]	2,5%
cgcut1	244	248	249	1,25	1,235	< 1	< 1	-	-	-
cgcut2	2892	3006	3076	1,614	1,006	78	< 1	> 78	1,05 [b]	7,4%
cgcut3	1860	2000	2240	2,714	1,151	78	15	5,2	2,37 [b]	3,3%
CU1	12330	12330	12330	-	1^+	172	46	3,739	7,09 [b]	2,4%
CU2	26100	26100	26100	-	1,001	438	94	4,660	19,83 [b]	2,2%
CU3	16608	16750	16750	1	1^+	594	62	9,581	30,47 [b]	1,9%
CU4	99495	99495	99656	∞	1,004	2:625	500	5,25	55,75 [b]	4,7%
CU5	173364	173568	174058	3,402	1,001	3:281	860	3,815	89,57 [b]	3,7%
CU6	158572	158572	158572	-	1,001	3:563	781	4,562	29,84 [b]	11,9%
CU7	247150	252313	252313	1	1,003	1:531	1:078	1,420	26,72 [b]	5,7%
CU8	433331	433331	433331	-	1,002	3:203	1:953	1,640	68,83 [b]	4,7%
CU9	657055	657055	657055	-	1,001	2:562	2:968	0,863	47,28 [b]	5,4%
media								5,16	-	4,84%
Istanze pesate di grandi dimensioni										
APT40	67154*	67654	67654	1	1,004	1.00:828	672	90,52		
APT41	206542*	215752	215752	1	1,003	1.15:750	1:640	46,19		
APT43	214651*	227938	227938	1	1,001	2.47:781	2:797	59,99		
APT44	73868*	74887	74887	1	1^+	29:750	672	44,27		
APT45	74691*	75888	75888	1	1^+	23:906	578	41,36		
APT46	149911*	151813	151813	1	1,003	1.07:078	1:312	51,12		
APT47	150234*	153058	153058	1	1,004	59:813	1:375	43,50		
APT48	167660*	170914	170914	1	1,001	1.25:218	1:609	52,96		
APT49	218388*	226610	226610	1	1^+	1.45:843	2:156	49,09		
media						1.15:107	-	53,22	85,37 [c]	88%
Istanze non pesate di grandi dimensioni										
APT30	140904	140904	140904	-	1^+	44:907	2:141	20,97		
APT31	822393	824701	824931	1,100	1^+	8.11:500	23:797	20,65		
APT32	38068	38068	38068	-	1,001	11:812	297	39,77		
APT33	236611*	236811	236818	1,035	1^+	1.31:906	3:813	24,10		
APT34	361167*	362520	362520	1	1^+	1.56:125	6:359	18,26		
APT35	621021*	622542	622644	1,067	1^+	3.45:234	11:797	19,09		
APT36	130744	130860	130860	1	1^+	29:532	1:515	19,49		
APT37	387276*	387533	387558	1,097	1,001	2.38:860	7:766	20,46		
APT38	261395*	261698	261698	1	1,001	1.36:344	4:625	20,83		
APT39	268750	269444	269444	1	1,002	59:093	3:734	15,83		
media						2.27:257	-	21,95	65,75 [c]	224%

* I valori *OPT* marcati da questo segno sono i migliori valori noti, senza prova di ottimalità

1^+ significa che il valore è maggiore di 1, ma differisce da 1 per una quantità inferiore a $5 \cdot 10^{-4}$

[a] ottenuto da Cung, Hifi e Le Cun (2000) con un approccio esatto bottom-up

[b] ottenuto da Fayard, Hifi e Zissimopoulos (1998) con l'algoritmo euristico GBSC

[c] ottenuto da Hifi (2004) con un algoritmo non esatto, ed ivi riportato come media per le istanze del sotto-gruppo

Tabella 4.1 Confronto fra le formulazioni classica e raffinata per la knapsack function

Si analizza per primo il sotto-gruppo di istanze pesate di dimensioni medio-piccole. Per l'istanza A1 il parametro ΔG ha valore 10, per A2 esso è 2,677 e per A3 è 1,521. Per le altre nove istanze del primo sotto-gruppo, cioè da CW1 a CW9, solo in un caso il valore del parametro *F* è migliore, vale a dire per CW8, per il quale ΔG = 1,163. Si noti inoltre che per l'istanza CW7 il valore della knapsack function è già uguale all'ottimo per cui nessun miglioramento era possibile. Tuttavia solo per le tre istanze CW5, CW6 e CW9, il parametro *WKF* vale 1, il che vuol dire che non vi è stato alcun miglioramento per nessun valore di knapsack function $f(l, w)$, mentre per le cinque istanze CW1, CW2, CW3, CW4 e CW7, il valore del parametro *WKF* è maggiore di 1.

Per il sotto-gruppo di quindici istanze non pesate, solo per due di esse, cioè CU3 e CU7, non c'è miglioramento in termini del parametro *F*. Sono state escluse in tal senso le cinque istanze CU1, CU2, CU6, CU8 e CU9, per le quali nessun miglioramento era possibile, come sopra per l'istanza CW7. Per l'istanza CU4, il valore migliorato del parametro *F* corrisponde all'ottimo. Per le altre sette istanze, cioè A4, A5, H, cgcut1, cgcut2, cgcut3 e CU5, il valore del parametro ΔG è nell'intervallo [1,204 ; 3,402]. Si noti inoltre che il valore del parametro *WKF* è sempre maggiore di 1.

Per le istanze pesate di grandi dimensioni, non c'è miglioramento in termini del valore del parametro *F*, per nessuna delle nove istanze elencate. Tuttavia, per tutte il valore del parametro *WKF* risulta maggiore di 1.

Infine, per le istanze non pesate di grandi dimensioni, si può notare che i valori del parametro *WKF* sono inferiori rispetto al precedente sotto-gruppo di istanze pesate, ma al contrario vi sono alcune istanze con un valore migliorato del parametro *F*. Per due istanze, cioè APT30 ed APT32, non era possibile alcun miglioramento per *F*. Per le quattro istanze APT34, APT36, APT38 ed APT39 il valore di *F* non è stato migliorato. Per le altre quattro istanze, cioè APT31, APT33, APT35 ed APT37, si è invece ottenuto un miglioramento ed il valore del parametro ΔG va da 1,018 ad 1,100.

Se ne conclude che i miglioramenti più evidenti dei valori della knapsack function sono stati ottenuti per le istanze di dimensione medio-piccola e nel caso delle istanze non pesate.

I tempi di calcolo necessari per la knapsack function migliorata sono quasi sempre maggiori rispetto alla formulazione classica. Infatti solo per l'istanza CU9 c'è un miglioramento, da quasi 3 a poco meno di 2,5 secondi, mentre per l'istanza H il tempo è lo stesso, cioè 47 millisecondi. Se si escludono, oltre a CU9 ed H, le istanze A2, cgcut1 e cgcut2, per le quali il valore del parametro CT_{old} è troppo piccolo, si può notare che per il sotto-gruppo di istanze pesate di dimensione medio-piccola, il rapporto *CTR* ha, come valore minimo, m_{CTR} = 2,411 e, come valore massimo, M_{CTR} = 11,27. Il valore medio è μ_{CTR} = 6,35 con un valore di deviazione standard pari a σ_{CTR} = 2,52. Per il secondo sotto-gruppo, con istanze non pesate, i valori degli analoghi parametri sono m_{CTR} = 1,420 , M_{CTR} = 9,581 , μ_{CTR} = 5,16 e σ_{CTR} = 2,59.

Per il gruppo di istanze di grandi dimensioni, i valori del parametro *CTR* sono maggiori rispetto al caso di dimensioni medio-piccole. In particolare, per il sotto-gruppo di istanze pesate, i corrispondenti valori sono m_{CTR} = 41,36 , M_{CTR} = 90,52 , μ_{CTR} = 53,22 e σ_{CTR} = 14,24 mentre, per il sotto-gruppo di istanze non pesate, m_{CTR} = 15,83 , M_{CTR} = 39,77 , μ_{CTR} = 21,95 e σ_{CTR} = 6,27.

Ai fini dell'analisi dei valori del parametro CTR_{al} , si esclude solo l'istanza cgcut1, per la quale $CT_{new} < 1$ millisecondo. Si vede che il valor medio è 5.76% per il primo sotto-gruppo di istanze pesate e 4.84% per il secondo di istanze non pesate. Per le istanze di grandi dimensioni e per entrambi i sotto-gruppi, le medie sono state ottenute come rapporto fra le medie rispettivamente di CT_{new} e di T_{al} , ottenendo così un valore di 88% per le istanze pesate e di 224% per quelle non pesate.

Si conclude che, anche rispetto ai tempi di calcolo, sembra che l'applicazione delle nuove idée sia migliore nel caso non pesato. Si noti anche che il tempo di calcolo CT_{new} è molto più piccolo rispetto a T_{al} , tranne per le istanze di grandi dimensioni, ma in tal caso il valore del parametro T_{al} è stato ottenuto con un approccio euristico, appositamente utilizzato per evitare elevati tempi di calcolo su questo tipo di istanze e senza alcuna pretesa di ottenere l'ottimo.

L'efficacia dei miglioramenti va necessariamente misurata tenendo conto non solo del valore di knapsack function $f(L_S, W_S)$, ma piuttosto del valore dell'upper bound $UB(L_S, W_S)$, usualmente ottenuto come minimo fra $f(L_S, W_S)$ ed $f_{1D}(L_S, W_S)$, o con il bounds merging spiegato nel paragrafo 4.3.5. Quindi, in Tabella 4.2 si riportano i valori di sei upper bound per l'intero stock, organizzati in tre coppie. I primi due, denotati rispettivamente con UB e UB^g, sono basati sugi miglioramenti originali proposti in questo Capitolo. L'upper bound UB è ottenuto utilizzando una procedura di programmazione dinamica per ottenere l'upper bound mono-dimensionale, mentre per UB^g è utilizzata una procedura greedy. L'apice g denota la stessa differenza nelle altre due coppie di upper bounds. Il terzo ed il quarto upper bound sono denotati rispettivamente con UB_V ed UB^g_V. Essi sono ottenuti con tutti i miglioramenti originali proposti ed in più con lo sfruttamento del bounds merging. Si noti che nel calcolo di UB_V sono quindi concentrate tutte le migliori tecniche, per cui tale upper bound domina tutti gli altri. Il quinto upper bound è denotato con F_V e corrisponde all'upper bound calcolato da León et al.. Infine F^g_V è calcolato come F_V, utilizzando una procedura greedy anziché la procedura di programmazione dinamica per l'upper bound mono-dimensionale. Si riportano i tempi di calcolo T_g e T_d , necessari a calcolare la lista di valori per l'upper bound mono-dimensionale, utilizzando rispettivamente la procedura greedy e quella di programmazione dinamica. Tali tempi sono espressi in millisecondi.

Istanza	F	UB	UB^g	UB_V	UB^g_V	F_V	F^g_V	T_g (msec)	T_d (msec)
A1	2040	2040	2040	2040	2040	2140	2177	< 1	235
A2	2830	2705	2725	2705	2725	2705	2725	< 1	328
A3	5524	5524	5524	5524	5524	5538	5562	< 1	579
A4	6264	6264	6264	6264	6264	6264	6300	< 1	656
A5	13142	13142	13142	13142	13142	13173	13174	< 1	1:797
H	12132	12132	12132	12114	12132	12336	12348	< 1	32
cgcut1	248	248	248	248	248	248	249	< 1	94
cgcut2	3006	2919	2920	2919	2920	2919	2920	< 1	140
cgcut3	2000	2000	2000	2000	2000	2020	2040	< 1	250

Tabella 4.2 Confronto fra upper bounds

Si sottolinea che in Tabella 4.2 sono stati elencati i valori di upper bound relativi alle sole istanze per le quali almeno uno di essi differisse dal valore del parametro F riportato in Tabella 4.1 ed anche in Tabella 4.2. Questo avviene solo per nove istanze, cioè da A1 ad A5, H, cgcut1, cgcut2 e cgcut3. Si sottolinea inoltre che per le istanze di elevate dimensioni, la procedura di programmazione dinamica per il calcolo dell'upper bound mono-dimensionale è risultato troppo oneroso in termini di tempo di calcolo, per cui esso non è stato realizzato.

Si confronta dapprima l'upper bound UB^g, ottenuto con i miglioramenti originali proposti e sfruttando una procedura greedy, con l'upper bound F_V proposto da León et al., che utilizza una procedura di programmazione dinamica. Per le istanze A2 e cgcut2 l'upper bound UB^g è maggiore di F_V, mentre esso è inferiore per le cinque istanze A1, A3, A5, H e cgcut3.

Si noti inoltre che solo per le istanze A2 e cgcut2 la differenza fra F ed UB non è nulla, e ciò significa che solo in pochi casi è importante utilizzare l'upper bound mono-

dimensionale, e che solo per l'istanza H c'è differenza fra UB ed UB_V, e ciò significa che solo in pochissimi casi è importante applicare il bounds merging.

Sulla base di queste considerazioni, si presuppone che una buona scelta sia utilizzare l'upper bound UB^g_V, dominato dall'upper bound UB_V solo per le tre istanze A2, H e cgcut2. Infatti, come spiegato sopra, nessun aumento di tempo di calcolo è necessario per sfruttare il bounds merging proposto da León et al. mentre un buon miglioramento del tempo di calcolo è ottenuto sfruttando una procedura greedy in luogo di una procedura di programmazione dinamica, come si può notare dai valori di T_g e T_d. Si noti inoltre che il peggioramento nella scelta della procedura, in termini di valore dell'upper bound, sembra avere un impatto maggiore quando si passa da F_V a F^g_V. Infatti, per tale coppia di upper bounds, si può notare un valore differente per tute le nove istanze, mentre per la coppia UB ed UB^g solo due istanze presentano valori differenti, cioè A2 e cgcu2. Ciò vuol dire che i miglioramenti proposti risultano più robusti.

Infine in Tabella 4.3 sono riportati i tempi di calcolo ed i valori di knapsack function associati all'intero stock, per i quattro passi migliorativi spiegati attraverso le diverse sezioni del paragrafo 4.3, per un sottoinsieme di istanze, ed esattamente tre per ciascuno dei quattro sotto-gruppi evidenziati nella Tabella 4.1. Il primo tempo di calcolo parziale, denotato con CT_n , fa riferimento all'utilizzo base delle coordinate normalizzate, come descritto nel paragrafo 4.3.1. Il secondo, basato sulle coordinate normalizzate compatibili introdotte nel paragrafo 4.3.2, è denotato con CT_c. Il terzo è relativo alle coppie di coordinate normalizzate, spiegate nel paragrafo 4.3.3, ed è denotato con CT_{cc} . Il quarto è denotato con CT_r ed è associato alle coordinate normalizzate raggiungibili, definite nel paragrafo 4.3.4 ed uguale al tempo di calcolo CT_{new} riportato in Tabella 4.1. Si riportano inoltre due altri tempi di calcolo T_c e T_r. Il primo è la porzione di tempo necessaria a calcolare gli insiemi $N_v(l, w)$ ed $N_h(l, w)$ nel calcolo associato al tempo CT_c , ed il secondo è la porzione di tempo necessaria a calcolare sia gli insiemi $N^*_v(l, w)$ che gli insiemi $N^*_h(l, w)$ nella computazione associata a CT_r. Questi tempi di calcolo sono espressi in millisecondi.

Istanza	CT_n	CT_c	CT_{cc}	CT_r	T_c (msec)	T_r (msec)
CW1	234	234	287	281	16	88
CW2	344	381	343	406	16	126
CW3	1:093	999	1:046	1:047	47	237
CU1	140	172	157	172	16	54
CU2	410	379	42	438	32	142
CU3	331	350	366	594	47	295
APT40	23:250	21:500	27:875	1.00:828	422	37:484
APT41	1.12:185	1.06:200	1.17:297	1.15:750	328	17:469
APT43	2.18:750	2.05:984	2.34:532	2.47:781	734	50:718
APT30	34:797	32:235	40:656	44:907	250	15:422
APT31	7.15:031	6.37:266	8.02:783	8.11:500	797	1.15:938
APT32	5:197	4:978	6:665	11:812	172	6:678

Tabella 4.3 Dettaglio dei tempi di calcolo

Si può notare che in quasi tutti i casi, tranne che per CW1, CW3 ed APT41, il tempo di calcolo CT_r è il maggiore. Inoltre, il calcolo di tutti gli insiemi N^* necessita di un maggior tempo T_r , rispetto a T_c , ma esso determina una riduzione del tempo di calcolo necessario nella seconda fase che calcola la tabella di valori di knapsack function. Infatti la porzione di tempo T_r è una parte non piccola del tempo CT_{new}, ma per tutte le istanze mostrate, il rapporto fra CT_r e CT_n è nell'intervallo [1 ; 3], il che vuol dire che i quattro tempi di calcolo sono comparabili.

4.5 Conclusioni

In questo Capitolo si è affrontato il problema di Cutting Stock bidimensionale a ghigliottina, proponendo un miglioramento dei valori di upper bound da usare per tutti i possibili rettangoli contenuti nello stock. I miglioramenti sono basati sullo sfruttamento delle coordinate normalizzate, e conducono ad una rivisitazione della knapsack function. Essi sono stati testati su istanze di letteratura, mostrando buoni risultati.

Le prospettive concernono due aspetti. Il primo consiste nel trovare ulteriori miglioramenti che possano essere più efficaci anche per le istanze di dimensioni elevate, in particolare per il caso pesato. Il secondo aspetto consiste nel ricercare strategie per guidare la scelta fra i sei possibili upper bounds confrontati attraverso la Tabella 4.2 nel precedente paragrafo.

Capitolo 5

Tecniche di calcolo

dei nuovi upper bounds

In molti problemi della vita reale, il principale vincolo fisico consiste in una limitata disponibilità di risorsa, o spaziale da poter riempire con determinati oggetti piccoli, o materiale da poter trasformare in desiderati piccoli oggetti. Nel primo caso, si è di fronte ad un problema di Packing, mentre nel secondo caso si ha un problema di Cutting.

Per l'insieme di problemi di Cutting e Packing, due lavori introduttivi sono stati quelli di Kantorovich (1939) e di Brooks, Smith, Stone e Tutte (1940), mentre dettagliate classificazioni possono essere trovate in Dyckhoff (1990) e, recentemente, in Wäscher, Haußner e Schumann (2007). Utili bibliografie sull'argomento si trovano in Dyckhoff e Finke (1992), Sweeney e Paternoster (1992), Dyckhoff, Scheithauer e Terno (1997) ed in Lodi, Martello e Monaci (2002). Per entrambe le classi di problemi, due obiettivi usuali sono la minimizzazione della risorsa usata per inserire o estrarre tutti gli oggetti desiderati, o la massimizzazione del guadagno associato al sottoinsieme di oggetti che possono essere inseriti nella (o estratti dalla) risorsa limitata.

Nel seguito si tratta il problema di Cutting bidimensionale, con un solo stock, o lastra, ricadendo così nel secondo tipo di obiettivi, e con oggetti tutti rettangolari. Si considera il caso in cui vi sia la necessità di utilizzare solo tagli verticali ed orizzontali che siano a *ghigliottina*, cioè che partino e terminino sui bordi del rettangolo tagliato. Questo problema è noto in letteratura come *Two-Dimensional Guillotine Cutting problem* (TDGC). Esso è molto interessante per le applicazioni nella trasformazione di materiale grezzo, per esempio vetro e legno, nonché metallo, carta, etc. Nel lavoro di Fayard, Hifi e Zissimopoulos (1998) c'è una interessante classificazione delle versioni di questo problema.

L'attenzione è qui concentrata sui metodi di risoluzione esatta. Fra questi, si possono distinguere differenti approcci, per esempio in termini di programmazione lineare (Lodi e Monaci, 2003; Belov, 2003), schemi di programmazione dinamica (Gilmore e Gomory, 1966; Hifi e Zissimopoulos, 1996), o approcci ad hoc, ben noti in letteratura, cioè l'approccio *top-down* e l'approccio *bottom-up*. Lavori relativi all'approccio top-down sono Herz (1972), Christofides e Whitlock (1977), Beasley (1985), Hifi e Zissimopoulos (1997), Hifi (1997a), Hifi (2004). Lavori relativi all'approccio bottom-up sono Wang (1983), Oliveira e Ferreira (1990), Viswanathan e Bagchi (1993), Hifi (1997b), Cung, Hifi e Le Cun (2000), G, Seong e Kang (2003), León, Miranda, Rodríguez e Segura (2007).

La maggior parte di tali metodi, ed in particolare quelli ad hoc, fa massiccio uso di upper bounds con lo scopo di accelerare le procedure di enumerazione implicita. In letteratura, ed in particolare per i metodi ad hoc, vengono utilizzati principalmente due tipi di upper bounds. Il primo è basato su un rilassamento di mono-dimensionalità, che utilizza cioè solo le aree sia dei pezzi che della lastra. Il secondo è adottato quando vi sia un numero limitato di pezzi desiderati da estrarre. Esso è basato su un rilassamento che consiste nel trascurare le limitazioni sui numeri di pezzi e che sfrutta la knapsack function, definita

appunto come la funzione che associa ad ogni coppia (l, w) di dimensioni, il valore ottimo relativo ad un rettangolo R che abbia tali dimensioni, in assenza di vincoli di domanda sui pezzi. Tale generico rettangolo sarà in seguito denotato con $R(l, w)$.

Nel primo paragrafo viene formalizzato il problema TDGC e vengono spiegate le sue varianti, riprese dal lavoro di Fayard, Hifi e Zissimopoulos (1998). Il secondo paragrafo riassume la formulazione della knapsack function e la corrispondente procedura computazionale, entrambe date nel lavoro di Gilmore e Gomory (1966). Si riporta l'errore notato da Herz (1972) in tale procedura e si mostra come correggerla ed anche come velocizzare il calcolo della tabella di valori della knapsack function. Il terzo paragrafo ha lo scopo di riassumere le coordinate normalizzate ed il modo proposto nel precedente Capitolo per migliorare gli upper bounds basati sulla knapsack function, in termini di riduzione dei valori. Viene anche proposta una implementazione ottimizzata per il calcolo delle coordinate normalizzate raggiungibili. Il quarto paragrafo descrive una procedura originale per unificare i miglioramenti con l'idea che supporta la procedura classica di Gilmore e Gomory. Il quinto paragrafo mostra i risultati computazionali mentre nel sesto sono date le conclusioni del Capitolo.

5.1 Definizione del problema

Nel problema del Cutting Stock bidimensionale a ghigliottina (*Two-Dimensional Guillotine Cutting Stock* o TDGC *problem*), i dati sono rappresentati da una coppia (*S*, *T*). Il rettangolo *S*, chiamato anche stock, rappresenta la lastra (o pannello) di materiale grezzo da tagliare ed ha dimensioni intere (L_S, W_S). L'insieme *T* contiene *n* quadruple $t_1 = (l_1, w_1, d_1, \pi_1)$, …, $t_n = (l_n, w_n, d_n, \pi_n)$, che rappresentano *n* tipi di pezzo rettangolari. Ciascun pezzo t_i ha lunghezza l_i , ampiezza w_i , area $l_i \cdot w_i$, massima domanda d_i e peso (o profitto) π_i , tutti interi positivi. Per ciascun tipo t_i , si suppone che ciascun pezzo possa essere estratto dalla lastra, vale a dire che entrambe le relazioni $l_i \le L_S$ e $w_i \le W_S$ sono valide.

Al fine di formalizzare in modo semplice il problema, sono necessarie alcune definizioni preliminari. Si intende per *cutting pattern* un qualsiasi vettore $B = (b_1, \ldots, b_n)$ di dimensione *n* ed i cui elementi siano numeri interi. Ad esso si associa implicitamente un insieme di pezzi che contenga, per ciascun tipo t_i , esattamente b_i pezzi, dove b_i è l'*i*-mo elemento del vettore *B*.

Un cutting pattern si dice *bounded*, o limitato, quando la relazione $b_i \le d_i$ valga per tutti i valori interi *i* da 1 ad *n*. In pratica ciò significa che il corrispondente insieme di pezzi è contenuto nell'insieme di pezzi disponibili, secondo la domanda associata a ciascun tipo di pezzi.

Un cutting pattern è *consistent*, o coerente, quando il corrispondente insieme di pezzi può essere estratto dallo stock con un insieme di tagli a ghigliottina verticali o orizzontali. Il corrispondente insieme di tagli, in grado di estrarre il relativo insieme di pezzi, e di produrre eventualmente degli scarti, è definito come *cutting scheme*, o schema di taglio.

Un cutting pattern si dice *feasible*, o fattibile, quando esso è limitato e coerente.

L'obiettivo nel problema del Cutting Stock consiste nello scegliere il miglior cutting pattern, cioè al quale sia associato il massimo valore di funzione obiettivo. Per ciascun cutting pattern *B*, esso è definito come $\Pi(B) = \sum_{i=1}^{n} (b_i \cdot \pi_i)$, cioè come somma dei profitti dei pezzi dell'insieme relativo a *B*.

Sotto questo aspetto, esistono due varianti del problema, a seconda di come sia definito il profitto dei pezzi. Quando, per ciascun tipo, il profitto è uguale all'area, il problema è classificato come *un-weighted*, o non pesato, mentre in caso contrario come *weighted*, o pesato. Si noti che nel caso non pesato, conviene comunque definire il profitto π_i, appunto uguale all'area $l_i \cdot w_i$, in modo da poter sempre utilizzare l'espressione generale su indicata per $\Pi(B)$. Il problema TDGC può dunque essere descritto nel seguente modo:

"massimizzare $\Pi(B)$ al variare dei cutting patterns *B* fattibili"

E' facile mostrare che tale problema è NP-hard, dato che, se $W_S = w_1 = w_2 = \ldots = w_n$, esso diventa equivalente al problema dello zaino mono-dimensionale, noto essere NP-hard.

Per ciascun tipo t_i , non più di $D_i = \lfloor L_S / l_i \rfloor \cdot \lfloor W_S / w_i \rfloor$ pezzi possono essere estratti dalla lastra *S*. Quindi, è inutile assegnare alla corrispondente domanda d_i un valore superiore a D_i . Quando, per almeno un tipo t_i , la relazione $d_i < D_i$ è valida, il problema è classificato come *constrained*, o vincolato, data la domanda limitata. Se invece $d_i = D_i$ per tutti i tipi t_i , con $i = 1, \ldots, n$, allora si parla di problema *un-constrained*, o non vincolato.

Di seguito si trattano gli upper bounds usualmente utilizzati negli approcci esatti ad hoc per la variante *constrained* con orientamento fisso dei pezzi, sia weighted che un-weighted.

5.2 Knapsack function

Gilmore e Gomory (1966) hanno introdotto e studiato le cosiddette *knapsack functions*. Qui si denota con $f(l, w)$ il valore della *knapsack function* per un dato rettangolo $R(l, w)$, che abbia dimensioni non superiori a quelle della lastra S.

Il valore $f(l, w)$ è definito come il valore ottimo di funzione obiettivo ottenibile per il problema parziale associato al rettangolo R. Essi hanno applicato questa definizione alla versione del problema *un-constrained* ed hanno studiato in dettaglio la relativa computazione, sia teoricamente che proceduralmente, ma la loro definizione può essere adottata, come essi stessi hanno sottolineato, per una qualunque versione.

Una qualunque soluzione per R deve necessariamente avere una delle seguenti tre caratteristiche alternative. Essa può estrarre un solo pezzo, oppure estrarne almeno due ed il corrispondente *cutting scheme* può contenere, come taglio principale che divida R in due parti, o un taglio verticale o un taglio orizzontale. Per ciascuna di queste tre possibilità, può essere definito un corrispondente termine parziale, per cui $f(l, w)$ può essere ottenuto come massimo fra questi tre termini, denotati rispettivamente con $f_0(l, w)$, $f_v(l, w)$ ed $f_h(l, w)$.

Il termine $f_0(l, w)$ è il valore ottimo associato al primo caso. Quindi, se almeno un pezzo può essere estratto da R, cioè le relazioni $l_i \le l$ e $w_i \le w$ valgono per il corrispondente tipo t_i, allora $f_0(l, w)$ è pari a $\max_{i=1,\dots,n} \{\pi_i : l_i \le l, w_i \le w\}$. In caso contrario, $f_0(l, w) = 0$.

Il termine $f_v(l, w)$ è associato al caso di taglio verticale. Se si definisce come coordinata di un taglio la sua distanza dal vertice in basso a sinistra di R, allora si può denotare con x_1 la coordinata di un generico taglio verticale. Ogni possibile coordinata x_1, cioè intera, positiva, ed inferiore alla lunghezza l di R, identifica un insieme di soluzioni, corrispondente a tutti i possibili cutting schemes che contengono un taglio verticale alla coordinate x_1 che tagli R. Il migliore valore, denotato con $\Pi(x_1)$, fra quelli di tali soluzioni può essere ottenuto come somma fra il valore ottimo associato al sotto-rettangolo, ottenuto a sinistra del taglio, denotato con $R_1(x_1, w)$, e quello associato al sotto-rettangolo, ottenuto a destra del taglio, denotato con $R_2(l - x_1, w)$. Se ne conclude che il valore $f_v(l, w)$ è ottenibile come massimo fra i valori $\Pi(x_1)$, al variare di x_1. Considerazioni simili possono essere fatte per il termine $f_h(l, w)$ associato al caso di taglio principale orizzontale. Dunque il calcolo di tutti i valori $f(l, w)$, al variare delle dimensioni l e w, può essere realizzato con la seguente relazione ricorsiva, riportata nel lavoro di Hifi (1997a), dove i termini f_v ed f_h non appaiono esplicitamente.

$$f(l,w) = \begin{cases} 0 & \text{if} \quad l = 0 \quad \text{or} \quad w = 0 \\ \max\left\{ f_0(l,w),\ \max_{\substack{x_1,x_2: \\ x_1+x_2 \le l \\ 0<x_1 \le x_2}} \{f(x_1,w) + f(x_2,w)\},\ \max_{\substack{y_1,y_2: \\ y_1+y_2 \le w \\ 0<y_1 \le y_2}} \{f(l,y_1) + f(l,y_2)\} \right\} & \text{else} \begin{bmatrix} 1 \le l \le L_S \\ 1 \le w \le W_S \end{bmatrix} \end{cases} \tag{5.1}$$

Questa formulazione, equivalente a quella data da Gilmore e Gomory (1966), utilizza anche la coordinata x_2 per il termine parziale associato a tagli verticali e la coordinata y_2 per il termine parziale associato a tagli orizzontali. Quando, per esempio, $x_2 < l - x_1$, il valore di x_2 rappresenta la coordinata di un secondo taglio verticale, che genera un ulteriore sotto-rettangolo $R_3(l - x_1 - x_2, w)$ alla sua destra, e nessun ulteriore valore è aggiunto nella somma

che esprime $f_v(l, w)$, poiché tale terzo sotto-rettangolo è considerato vuoto. Il risultato finale è però lo stesso.

Gilmore e Gomory hanno anche proposto una procedura più efficiente rispetto ad una che ricalchi la relazione (5.1). Tuttavia Herz (1972) ha mostrato che essa è scorretta. Nel seguito, vengono riassunti gli aspetti principali dei miglioramenti procedurali introdotti, ovvero quelli che sono interessanti per le considerazioni descritte in questo Capitolo. Inoltre viene data esplicitamente la correzione relativa alla nota di Herz e viene aggiunto un miglioramento implementativo che è conseguenza delle definizioni di Gilmore e Gomory, ma che essi non hanno sfruttata.

Le differenze fra la procedura di Gilmore e Gomory e la relazione (5.1) consistono innanzitutto nel fatto che essi considerano solo coppie x_1, x_2 tali che $x_1 + x_2 = l$ e coppie y_1, y_2 tale che $y_1 + y_2 = w$, ma questo non è scorretto, come si evince dalle considerazioni fatte in precedenza. Essi, inoltre, utilizzano degli accorgimenti per velocizzare la computazione. Per poterli sfruttare, necessitano però di utilizzare due termini in più, oltre ai tre prima evidenziati, fra cui selezionare il massimo da assegnare a $f(l, w)$. Tali due termini sono $f(l - 1, w)$ ed $f(l, w - 1)$, aggiunti per evitare di trascurare le soluzioni con un terzo sotto-rettangolo vuoto. Del resto è ovvio che $f(l, w)$ deve essere non inferiore né all'uno né all'altro.

In termini informali, si può affermare che i loro accorgimenti, che hanno lo scopo di velocizzare la computazione, consistono nell'enumerare uno solo fra modi diversi, ma simmetrici, di ottenere un valore $f(l, w)$ attraverso un insieme di tagli verticali, paralleli e di uguale lunghezza, selezionando cioè una sola fra le possibili soluzioni associate a tali tagli, che hanno tutte un valore appunto pari ad $f(l, w)$. Analogo discorso vale per soluzioni simmetriche ottenibili con tagli orizzontali, paralleli e di uguale lunghezza.

Per poter utilizzare i suddetti accorgimenti, Gilmore e Gomory definiscono due ulteriori funzioni, qui denotate con $\lambda(l, w)$ e $\omega(l, w)$. La prima, cioè $\lambda(l, w)$, è relativa ai tagli verticali e la seconda, cioè $\omega(l, w)$, ai tagli orizzontali. Nel seguito viene data la definizione formale di λ ed ω per poi mostrare come il loro utilizzo permetta di realizzare una strategia di anti-simmetria, con riferimento ad un esempio con tagli verticali.

Se $f(l, w) = f(l - 1, w)$, allora $\lambda(l, w)$ è settato pari a 0, altrimenti $\lambda(l, w)$ è uguale al minimo valore x_1 tale che $1 \leq x_1 \leq l$ ed $f(l, w) = f(x_1, w) + f(l - x_1, w)$, dove $f(0, w) = 0$. Dunque $\lambda(l, w) = l$ implica che $f(l, w)$ è ottenibile o come valore di una soluzione contenente un solo pezzo, esattamente di dimensioni (l, w), oppure attraverso una somma associata al termine parziale per i tagli orizzontali, ma non attraverso una somma associata al termine parziale per i tagli verticali. Una analoga definizione ed analoghe considerazioni valgono per la funzione $\omega(l, w)$.

Per comprendere come esse siano sfruttate, si supponga per esempio che u_1, u_2 ed u_3 siano tre lunghezze tali che $u_1 < u_2 < u_3$ e sia $R(l, w)$ un rettangolo tale che $l = u_1 + u_2 + u_3$. Si supponga inoltre che le migliori soluzioni per i tre sotto-rettangoli $R_1(u_1, w)$, $R_2(u_2, w)$ ed $R_3(u_3, w)$ non siano ottenibili utilizzando un taglio principale verticale, vale a dire che $u_i = \lambda(u_i, w)$ per $i = 1, 2$ e 3, secondo la definizione data della funzione λ.

Se il valore $f(l, w)$ è ottenibile come somma dei tre valori $f(u_1, w)$, $f(u_2, w)$ ed $f(u_3, w)$, allora si deduce facilmente che $f(u_i + u_j, w) = f(u_i, w) + f(u_j, w)$ per le coppie $i, j \in \{1, 2, 3\}$ tali che $i \neq j$. Ne consegue che $f(l, w)$ può essere ottenuto con tre somme diverse ma equivalenti, $f(u_1, w) + f(u_2 + u_3, w)$, $f(u_2, w) + f(u_1 + u_3, w)$ oppure $f(u_3, w) + f(u_1 + u_2, w)$. Guardando la relazione (5.1), questo significa che vi sono sei possibili valori di x_1 tali che $f(l, w) = f(x_1, w) + f(l - x_1, w)$, ed essi sono u_1, u_2, u_3, $u_1 + u_2$, $u_1 + u_3$ ed $u_2 + u_3$, i quali, se si suppone senza perdita di generalità che essi siano tutti diversi, sono riportati in ordine crescente.

La strategia di Gilmore e Gomory impone di considerare solo valori x_1 tali che valgono le due seguenti condizioni. La prima, denotata con $C_{v,1}$, richiede che $x_1 = \lambda(x_1, w)$ e la

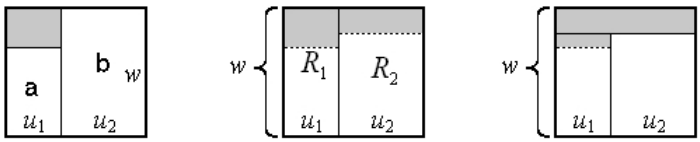
seconda, con $C_{v,2}$, richiede che $x_1 \leq \lambda(l - x_1, w)$, permettendo così, nell'esempio che si sta trattando, di considerare solo uno fra i sei possibili valori di x_1, e cioè il minimo.

Infatti, la condizione $C_{v,1}$ esclude gli ultimi tre valori, cioè $u_1 + u_2$, $u_1 + u_3$ e $u_2 + u_3$, poiché $\lambda(u_1 + u_2, w) = \lambda(u_1 + u_3, w) = u_1$ e $\lambda(u_2 + u_3, w) = u_2$. La condizione $C_{v,2}$ esclude i valori u_2 ed u_3, permettendo dunque di utilizzare il solo valore u_1. Infatti, vale la disuguaglianza $u_1 \leq \lambda(u_2 + u_3, w)$, mentre $u_2 > \lambda(u_1 + u_3, w)$ ed $u_3 > \lambda(u_1 + u_2, w)$.

Si può ora notare che la posizione $\lambda(l, w) = 0$ nel caso in cui $f(l, w) = f(l - 1, w)$ è utile ad evitare di usare il valore $f(l, w)$ in somme legate a tagli verticali, per esempio in una somma del tipo $f(l, w) + f(l_2, w)$. Questo è corretto poiché il valore $f(l - 1, w)$ può essere utilizzato per ottenere lo stesso risultato, attraverso la somma $f(l - 1, w) + f(l_2, w)$.

La condizione $C_{v,2}$ è sfruttata per costruire una procedura iterativa, rispetto al termine associato ai tagli verticali, che ha due cicli, uno innestato nell'altro. Quello esterno è ripetuto per ogni possibile valore dell'ampiezza w. Nel ciclo più esterno la variabile x_2 va da 1 ad L_S. Nel ciclo più interno la variabile x_1 va da 1 a $\lambda(x_2, w)$ dove, se si denota la somma $x_1 + x_2$ con l, allora x_2 ha il ruolo di $l - x_1$ nella condizione $C_{v,2}$, così come espressa sopra. Per ciascun valore permesso di x_1, i valori $f(x_1 + x_2, w)$ e $\lambda(x_1 + x_2, w)$ possono essere aggiornati, aumentando il primo e riducendo il secondo.

Due miglioramenti sono così ottenuti. Il primo consiste nel fatto che solo valori x_1 tali che l'uguaglianza $x_1 = \lambda(x_1, w)$ sia valida, vengono utilizzati. Inoltre, solo i valori non maggiori di $\lambda(x_2, w)$, invece che di $L_S - x_2$, vengono permessi. Tuttavia, Gilmore e Gomory non hanno pienamente sfruttato la condizione $C_{v,1}$ e di seguito viene proposto un modo per farlo. Prima però si conclude il riassunto della loro procedura. Oltre alle condizioni $C_{v,1}$ e $C_{v,2}$, essi impongono anche la condizione $C_{v,3}$, la quale richiede che $\omega(x_1, w) > 0$. Comunque tale condizione è quella che causa l'errore notato da Herz, per cui viene nel seguito sostituita con una condizione rilassata, che richieda cioè che una sola fra le relazioni $\omega(x_1, w) > 0$ e $\omega(x_2, w) > 0$ sia valida. Questo corregge l'errore, poiché permette di ottenere soluzioni come quella di Figura 5.1a, utilizzata da Herz per dimostrare l'errore.



a) b) c)

Figura 5.1 Condizione $C_{v,3}$

Si nota infatti che $\omega(u_1, w) = 0$, poiché il pezzo 'a' sulla sinistra del taglio verticale ha un'ampiezza minore di w, e la somma $f(u_1, w) + f(u_2, w)$ non viene permessa, dato che $x_1 = u_1$ è scartato per l'erronea condizione $C_{v,3}$, mentre $x_1 = u_2$ è scartato per la condizione $C_{v,2}$ poiché vale $u_2 > \lambda(u_1, w) = u_1$. La correzione prima proposta per la condizione $C_{v,3}$, invece, permette di usare il valore $x_1 = u_1$, poiché $\omega(u_2, w) = w > 0$. Per quanto si sappia, nessuno aveva ancora esplicitamente dato la correzione, da quando Herz ha provato l'errore, probabilmente poiché Herz ha anche proposto una procedura alternativa basata sullo sfruttamento delle coordinate normalizzate, di cui si parla nel paragrafo successivo.

In generale, la condizione $C_{v,3}$ può essere usata poiché, se $\omega(x_1, w) = \omega(x_2, w) = 0$, entrambe le uguaglianze $f(x_1, w) = f(x_1, w - 1)$ e $f(x_2, w) = f(x_2, w - 1)$ valgono, secondo la definizione di ω, per cui la migliore soluzione (Figura 5.1b), sia per $R_1(x_1, w)$ che per $R_2(x_2, w)$, è ottenibile con un taglio orizzontale che lasci una parte residua vuota sopra di esso. La soluzione mostrata in Figura 5.1b per il rettangolo $R(x_1 + x_2, w)$, con valore pari alla somma $f(x_1, w) + f(x_2, w)$ ed ottenibile con un taglio verticale "sommando" poi le migliori

soluzioni associate ad R_1 ed R_2, può essere ottenuta anche con un taglio principale che sia orizzontale (Figura 5.1c). Quindi $f(x_1, w) + f(x_2, w) \leq f(x_1 + x_2, w - 1)$ e dunque è inutile realizzare la somma.
Nel caso di taglio principale orizzontale, analoghe condizioni $C_{h,1}$, $C_{h,2}$ e $C_{h,3}$ vengono usate.

Si sottolinea ora che, se la condizione $C_{v,3}$ non viene utilizzata, l'effetto è che per una qualsiasi lunghezza l, un maggior numero di coppie x_1, x_2 tali sia che $x_1 + x_2 = l$ sia che $f(l, w) = f(x_1, w) + f(x_2, w)$, può essere utilizzato per realizzare la relativa somma con l'effetto che $\lambda(l, w)$ può diventare più piccolo di l. Questo effetto può a sua volta far ridurre lo sforzo computazionale complessivo, sulla base delle due condizioni $C_{v,1}$ e $C_{v,2}$. Inoltre, nessun valore $\omega(l, w)$ deve più essere letto per la computazione associata a tagli principali verticali, per cui il maggiore sforzo computazionale associato al calcolo della somma $f(x_1, w) + f(x_2, w)$ viene così bilanciato. Considerazioni simili possono essere fatte sulla condizione $C_{h,3}$. Dunque, un primo miglioramento per la procedura di Gilmore e Gomory può essere ottenuto evitando di utilizzare le condizioni $C_{v,3}$ e $C_{h,3}$.

Si passa ora a spiegare perché si è detto che Gilmore e Gomory non hanno pienamente sfruttato la condizione $C_{v,1}$, ovvero $x_1 = \lambda(x_1, w)$. Infatti, per ciascun valore di w, come asserito prima, essi realizzano un ciclo in cui il valore x_1 parte da 1 e cresce con passo 1. Al contrario, si propone di costruire un apposito insieme $\Lambda(w)$, che contenga solo i valori x_1 per i quali l'uguaglianza $x_1 = \lambda(x_1, w)$ sia valida, sfruttando poi $\Lambda(w)$ allo scopo di realizzare il ciclo interno solo per i valori che vi appartengono, ovviamente senza superare $\lambda(x_2, w)$, rispettando così anche la condizione $C_{v,2}$, come peraltro già nella procedura di Gilmore e Gomory. Analogamente, si propone di definire e sfruttare gli insiemi $\Omega(l)$ per i cicli interni associati ai termini relativi a tagli principali orizzontali.

Il Box 5.1 riporta dunque la procedura **P**, che rappresenta una versione rifinita rispetto alla procedura classica e che incorpora i miglioramenti proposti.

Sono stati utilizzati i simboli f', λ', ω', Λ' ed Ω' per denotare variabili ed insiemi che assumono un valore transitorio, aggiornato durante la procedura. Al termine della procedura, per ciascuna coppia di dimensioni (l, w), ognuno di essi equivale al corrispondente fra f, λ, ω, Λ ed Ω. Inoltre vengono esplicitamente utilizzati i simboli f_v per il termine parziale della relazione (5.1), associato ai tagli verticali, ed f_h per il termine parziale associato ai tagli orizzontali.

Quindi sono utilizzati anche i simboli f_v' ed f_h' per denotare i corrispondenti valori transitori. Infine si sottolinea che ulteriori passi della procedura di Gilmore e Gomory, non soddisfacenti le definizioni teoriche da essi date, sono stati corretti come spiegato meglio nel seguito.

All'inizio della procedura **P**, tutti i valori $f_v'(l, w)$ ed $f_h'(l, w)$ vengono inzializzati a zero. Inoltre, per ciascun tipo t_i , con dimensioni (l_i, w_i), vengono realizzate le inizializzazioni $\lambda'(l_i, w_i) = l_i$ e $\omega'(l_i, w_i) = w_i$, considerando così le soluzioni che estraggono solo un corrispondente pezzo dal rettangolo $R(l_i, w_i)$, senza l'ausilio di tagli. Infine, tutti gli insiemi $\Lambda'(w)$ e $\Omega'(l)$ vengono inizializzati vuoti. I due cicli esterni, rispettivamente sulle ampiezze e sulle lunghezze, permettono di eseguire il *core* della procedura per ogni possibile coppia di dimensioni (x_2, y_2).

Il *core* è costituito da due parti. La prima serve a calcolare in modo definitivo i valori $f(x_2, y_2)$, $\lambda(x_2, y_2)$ ed $\omega(x_2, y_2)$. Prima di comprenderla appieno, è necessario studiare la seconda parte.

Procedure P

Set $f_v'(l, w) = f_h'(l, w) = 0$ per $1 \le l \le L_S$, $1 \le w \le W_S$
Set $\lambda'(l_i, w_i) = l_i$ ed $\omega'(l_i, w_i) = w_i$ per $i = 1, \ldots, n$
Set $\Lambda(w) = \varnothing$ per $1 \le w \le W_S$ ed $\Omega(l) = \varnothing$ per $1 \le l \le L_S$
For $y_2 = 1$ **To** W_S **Do** //ciclo esterno sulle ampiezze
For $x_2 = 1$ **To** L_S **Do** //ciclo esterno sulle lunghezze
//Verifica finale per i valori $f'(x_2, y_2)$, $\lambda'(x_2, y_2)$ ed $\omega'(x_2, y_2)$
Set $f_v(x_2, y_2) = f_v'(x_2, y_2)$ and $f_h(x_2, y_2) = f_h'(x_2, y_2)$
Set $f'(x_2, y_2) = \max\{ f_0(x_2, y_2) , f_v(x_2, y_2) , f_h(x_2, y_2) \}$
If $f'(x_2, y_2) > \max\{f(x_2 - 1, y_2), f(x_2, y_2 - 1)\}$ **Then**
Begin // Blocco **A**
If $f'(x_2, y_2) > f_v(x_2, y_2)$ **Then Set** $\lambda'(x_2, y_2) = x_2$
If $f'(x_2, y_2) > f_h(x_2, y_2)$ **Then Set** $\omega'(x_2, y_2) = y_2$
End
Else
Begin //Blocco **B**
Set $f'(x_2, y_2) = \max\{f(x_2 - 1, y_2), f(x_2, y_2 - 1)\}$

If $f(x_2 - 1, y_2) \ge f(x_2, y_2 - 1)$ **Then Set** $\lambda'(x_2, y_2) = 0$
Else If $f'(x_2, y_2) > f_v(x_2 - 1, y_2)$ **Then Set** $\lambda'(x_2, y_2) = y_2$

If $f(x_2, y_2 - 1) \ge f(x_2 - 1, y_2)$ **Then Set** $\omega'(x_2, y_2) = 0$
Else If $f'(x_2, y_2) > f_h(x_2 - 1, y_2)$ **Then** $\omega'(x_2, y_2) = y_2$
End

Set $f(x_2, y_2) = f'(x_2, y_2)$, $\lambda(x_2, y_2) = \lambda'(x_2, y_2)$ ed $\omega(x_2, y_2) = \omega'(x_2, y_2)$

//Aggiornamento degli insiemi
If $\lambda'(x_2, y_2) = x_2$ **Then** add x_2 to the set $\Lambda'(y_2)$
If $\omega'(x_2, y_2) = y_2$ **Then** add y_2 to the set $\Omega'(x_2)$

//valgono $\Lambda'(y_2) = \Lambda(y_2) \cap \{1, \ldots, x_2\}$ e $\Omega'(x_2) = \Omega(x_2) \cap \{1, \ldots, y_2\}$
//ciclo interno sulle lunghezze
For Each $x_1 \in \Lambda'(y_2)$ **such that** $x_1 \le \lambda(x_2, y_2)$ **Do**
If $f(x_1, y_2) + f(x_2, y_2) > f_v'(x_1 + x_2, y_2)$ **Then**
Set $f_v'(x_1 + x_2, y_2) = f(x_1, y_1) + f(x_2, y_2)$, $\lambda'(x_1 + x_2, y_2) = x_1$
Else If $f(x_1, y_2) + f(x_2, y_2) = f_v'(x_2, y_1 + y_2)$ **Then**
//vale $x_1 < \lambda'(x_1 + x_2, y_2)$
Set $\lambda'(x_1 + x_2, y_2) = x_1$
End For

//ciclo interno sulle ampiezze
For Each $y_1 \in \Omega'(x_2)$ **such that** $y_1 \le \omega'(x_2, y_2)$ **Do**
If $f(x_2, y_1) + f(x_2, y_2) > f_h'(x_2, y_1 + y_2)$ **Then**
Set $f_h'(x_2, y_1 + y_2) = f(x_2, y_1) + f(x_2, y_2)$, $\omega'(x_2, y_1 + y_2) = y_1$
Else If $f(x_2, y_1) + f(x_2, y_2) = f_h'(x_2, y_1 + y_2)$ **Then**
//vale $y_1 < \omega'(x_2, y_1 + y_2)$
Set $\omega'(x_2, y_1 + y_2) = y_1$
End For
End For
End For

Box 5.1 Procedura **P**

La seconda parte verifica se i valori $f_v'(l, y_2)$ e $\lambda'(l, y_2)$, con $l > x_2$, ed i valori $f_h'(x_2, w)$ ed $\omega'(x_2, w)$, con $w > y_2$, debbano essere aggiornati. Questo viene fatto combinando il valore $f(x_2, y_2)$ rispettivamente con i valori $f(x_1, y_2)$, con $x_1 \le x_2$, e con i valori $f(x_2, y_1)$, con $y_1 \le y_2$. Solo i valori x_1 permessi dalle condizioni $C_{v,1}$ e $C_{v,2}$ sono utilizzati e si sottolinea che i corrispondenti valori $f(x_1, y_2)$ ed $f(x_2, y_1)$ sono già definitivamente disponibili, in virtù del fatto che $x_1 \le x_2$ ed $y_1 \le y_2$ ed i cicli esterni sono realizzati con valori crescenti di x_2 ed y_2.

Se si trova qualche valore x_1 per il quale la somma $f(x_1, y_2) + f(x_2, y_2)$ equivalga il valore corrente di $f_v'(x_1 + x_2, y_2)$, ciò significa che una precedente somma $f(x'_1, y_2) + f(x'_2, y_2)$ ha dato lo stesso risultato, con $x'_2 < x_2$ e dunque tale che $x'_1 = x_1 + x_2 - x'_2 > x_1$. Questo è vero per il fatto che il ciclo esterno sulle lunghezze è realizzato con valori crescenti della variabile x_2. Quindi x_1 è inferiore al valore corrente di $\lambda'(x_1 + x_2, y_2)$, che dunque necessita di essere aggiornato. Una proprietà simile vale per il ciclo interno sulle ampiezze.

La nuova correzione rispetto alla procedura di Gilmore e Gomory è nei due blocchi **A** e **B** della prima parte del *core* della procedura. Essi determinano i valori finali di f, λ ed ω per le correnti dimensioni (x_2, y_2). Infatti Gilmore e Gomory usano una sequenza di passi equivalente a quella riportata nel Box 5.2, dove gli aggiornamenti nei cicli interni sono direttamente realizzati sui valori transitori $f'(l, w)$, dato che essi non usato f_v ed f_h .

Per entrambe le procedure, il valore $f'(x_2, y_2)$, all'inizio del *core*, corrisponde al massimo valore ottenuto con le somme permesse $f(u', y_2) + f(u'', y_2)$, tali che $u' + u'' = x_2$, e le somme permesse $f(x_2, z') + f(x_2, z'')$, tali che $z' + z'' = y_2$. Dunque, deve essere eseguita una verifica finale, che confronti il valore transitorio $f'(x_2, y_2)$ con $f(x_2 - 1, y_2)$ ed $f(x_2, y_2 - 1)$. Entrambi o uno solo di questi valori potrebbero infatti essere maggiori di $f'(x_2, y_2)$, poiché non tutte le somme sono permesse, ma questo aspetto è stato già precedentemente analizzato.

GG Procedure

Set $f'(l, w) = f_0(l, w)$ per $0 \le l \le L_S$, $0 \le w \le W_S$
Set $\lambda'(l_i, w_i) = l_i$ ed $\omega'(l_i, w_i) = w_i$ per $i = 1, \ldots, n$

For $y_2 = 1$ **To** W_S **Do** // ciclo esterno sulle ampiezze
  For $x_2 = 1$ **To** L_S **Do** // ciclo esterno sulle lunghezze

    If $f'(x_2, y_2) > f(x_2 - 1, y_2)$ **Then**
      Realizza il ciclo interno sulle lunghezze, utilizzando anche $C_{v,3}$
    Else
      Set $f'(x_2, y_2) = f(x_2 - 1, y_2)$
      Set $\lambda'(x_2, y_2) = 0$
    End If

    If $f'(x_2, y_2) > f(x_2, y_2 - 1)$ **Then**
      Realizza il ciclo interno sulle ampiezze, utilizzando anche $C_{h,3}$
    Else
      Set $f'(x_2, y_2) = f(x_2, y_2 - 1)$
      Set $\omega'(x_2, y_2) = 0$
    End If

    Set $f(x_2, y_2) = f'(x_2, y_2)$; $\lambda(x_2, y_2) = \lambda'(x_2, y_2)$; $\omega(x_2, y_2) = \omega'(x_2, y_2)$
  End For
End For

Box 5.2 Procedura di Gilmore e Gomory

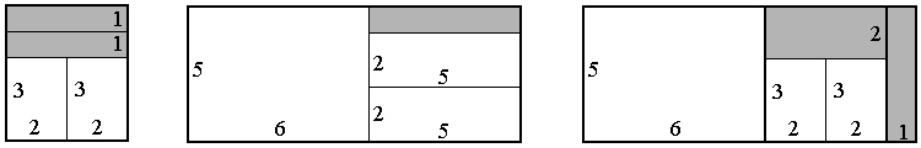
Se il massimo fra i tre valori $f_0(x_2, y_2)$, $f_v'(x_2, y_2)$ ed $f_h'(x_2, y_2)$, vale a dire $f'(x_2, y_2)$, è maggiore sia di $f(x_2 - 1, y_2)$ che di $f(x_2, y_2 - 1)$, allora la procedure **P** verifica i valori di $\lambda'(x_2, y_2)$ ed $\omega'(x_2, y_2)$, poiché il primo è stato settato tenendo conto solo degli aggiornamenti di $f_v'(x_2, y_2)$ ed il secondo tenendo conto solo di $f_h'(x_2, y_2)$. Infatti, se $f(x_2, y_2)$ è maggiore di $\max\{ f(x_2 - 1, y_2), f_v(x_2, y_2) \}$, allora $\lambda(x_2, y_2)$ è uguale ad x_2, ed analogamente per $\omega(x_2, y_2)$.

Se invece il massimo fra i tre valori $f_0(x_2, y_2)$, $f_v'(x_2, y_2)$ ed $f_h'(x_2, y_2)$ è minore o uguale sia di $f(x_2 - 1, y_2)$ che di $f(x_2, y_2 - 1)$, allora il valore $f(x_2, y_2)$ deve essere uguale al massimo fra questi due valori. Se il massimo è $f(x_2 - 1, y_2)$, allora $\lambda(x_2, y_2) = 0$, secondo la sua definizione, altrimenti vale la relazione $f(x_2, y_2) > f(x_2 - 1, y_2)$ e quindi, se anche la relazione $f(x_2, y_2) > f_v(x_2, y_2)$ è valida, allora $\lambda(x_2, y_2)$ è uguale ad x_2, come prima. Per $\omega(x_2, y_2)$ valgono argomentazioni simili.

Quindi, se $f'(x_2, y_2) < f(x_2 - 1, y_2) < f(x_2, y_2 - 1)$, il valore finale di $f'(x_2, y_2)$ deve essere uguale ad $f(x_2, y_2 - 1)$, maggiore sia di $f(x_2 - 1, y_2)$ che di $f_v'(x_2, y_2)$, e dunque il valore finale di $\lambda'(x_2, y_2)$ deve essere x_2. La procedura di Gilmore e Gomory, al contrario, setta erroneamente $\lambda'(x_2, y_2) = 0$, mentre il blocco **B** funziona correttamente.

Per dimostrare che l'errore si può effettivamente realizzare, si mostra un esempio. Si supponga di avere tre tipi di pezzo, $t_1 = (l_1 = 2, w_1 = 3, \pi_1 = 2)$, $t_2 = (l_2 = 5, w_2 = 2, \pi_2 = 3)$ ed infine $t_3 = (l_3 = 6, w_3 = 5, \pi_3 = 10)$, ed una lastra con dimensioni $L_S = 11$ e $W_S = 5$. Sulla base delle definizioni di f, λ ed ω, supponiamo che i seguenti valori siano correttamente calcolati. Si ha $f(3, 5) = f(2, 5) = f(2, 2) = f_0(2, 2) = 2$ e quindi $\lambda(3, 5) = \lambda(2, 5) = 0$. Il valore $f(4, 3) = 4$ corrisponde solo alla soluzione (Figura 5.2a) che estrae due pezzi di tipo t_1 attraverso un taglio verticale di coordinata 2, e dunque $\lambda(4, 3) = 2$ ed $\omega(4, 3) = 3$. Inoltre, $f(4, 5) = f(4, 4) = 4$, dove ciascuno di questi due valori corrisponde o alla soluzione con un rettangolo di scarto sopra un taglio orizzontale alla coordinata 3 ed estraente due pezzi di tipo t_2 sotto di esso, o alla soluzione che estrae gli stessi due pezzi con un taglio principale verticale alla coordinata 2 e poi un taglio orizzontale di coordinata 3 sia a destra che a sinistra di esso. Quindi $\lambda(4, 5) = \lambda(4, 4) = 2$ ed $\omega(4, 5) = \omega(4, 4) = 0$. In realtà, la condizione corretta $C_{v,3}$ utilizzata nella procedura classica, non permetterebbe la costruzione della seconda soluzione, per cui si otterrebbe $\lambda(4, 5) = \lambda(4, 4) = 4$. Poi $f(5, 3) = f(5, 2) = f_0(5, 2) = 3$ da cui $\omega(5, 3) = 0$. Il valore $f(5, 4) = 6$ corrisponde solo alla soluzione (Figura 5.2b) che estrae due pezzi di tipo t_2 con un taglio orizzontale alla coordinata 2, per cui $\lambda(5, 4) = 5$ ed $\omega(5, 4) = 2$. Infine il valore $f(6, 5) = 10$ corrisponde solo alla soluzione che estrae un pezzo di tipo t_3, senza tagli, per cui si ottiene $\lambda(6, 5) = 6$ ed $\omega(6, 5) = 5$.

Il valore $f(11, 5)$, associato all'intera lastra, deve essere 16, corrispondente alla soluzione (Figura 5.2b) che estrae un pezzo di tipo t_3 sulla sinistra di un taglio verticale di coordinata 6, e due tipi di pezzo t_2 sul rettangolo residuo destro, attraverso due ulteriori tagli orizzontali. Questa soluzione corrisponde alla somma $f(6, 5) + f(5, 5)$, dove il valore $f(5, 5)$ dovrebbe essere settato uguale a 6. Tuttavia, nessuna somma permessa, relativa a tagli verticali o orizzontali, determina l'aggiornamento dei valori $f'(5, 5)$ ed $\lambda'(5, 5)$ per cui $f'(5, 5)$ mantiene il valore $f_0(5, 5) = 3$ fino all'inizio del *core* con $x_2 = 5$ ed $y_2 = 5$. Quindi ci si trova nelle condizioni in cui $3 = f'(5, 5) < f(4, 5) = 4 < f(5, 4) = 6$, per cui la procedura classica setta correttamente $f(5, 5) = 6$, ma erroneamente $\lambda(5, 5) = \omega(5, 5) = 0$, come spiegato in generale sopra. L'effetto è che la somma $f(6, 5) + f(5, 5)$ non è permessa per ottenere $f(11, 5)$ e quindi essi ottengono erroneamente $f(11, 5)$ uguale ad $f(10, 5) = f(6, 5) + f(4, 5) = 10 + 4 = 14$, che corrisponde alla soluzione mostrata in Figura 5.2c.



a) b) c)

Figura 5.2 Errori nella procedura classica

Un diverso errore accade se, in generale, $f'(x_2, y_2) < f(x_2, y_2 - 1) < f(x_2 - 1, y_2)$ e nessuna somma $f(x_2, z') + f(x_2, z'')$ determina il settaggio del valore $\omega'(x_2, y_2)$. Infatti, nella procedura classica, tale valore non viene più settato, poiché $f'(x_2, y_2)$ è settato pari ad $f(x_2 - 1, y_2)$. Al contrario, il blocco **B** setta correttamente $\omega'(x_2, y_2) = y_2$.

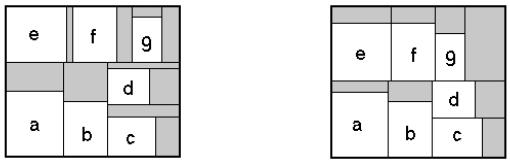
Per provare che anche questo errore si possa effettivamente realizzare, basta sfruttare lo stesso esempio di prima, scambiando le dimensioni sia della lastra, da (11, 6) a (6, 11), che dei pezzi, per i cui tipi le dimensioni diventano rispettivamente (3, 2), (4, 5) e (5, 6). Per motivazioni del tutto analoghe a prima, quando $x_2 = y_2 = 5$, prima di ottenere il valore finale di $f'(5, 5)$, valgono le relazioni $3 = f'(5, 5) < f(5, 4) = 4 < f(4, 5) = 6$, per cui il valore $\omega'(5, 5)$ non viene mai settato.

5.3 Coordinate normalizzate e relativi miglioramenti

In questo paragrafo si riassumono le *coordinate normalizzate* proposte da Herz (1972) e da Christofides e Withlock (1977), prima di riportare come esse sono state usate nel precedente Capitolo per migliorare gli upper bounds basati sul raffinamento della knapsack function. Si propone inoltre una procedura ottimizzata per calcolare sia le coordinate normalizzate che gli insiemi di coordinate normalizzate raggiungibili.

5.3.1 Coordinate normalizzate

Christofides e Withlock (1977) hanno definito le *soluzioni normalizzate*. Si tratta di soluzioni in cui tutti i pezzi sono posizionati il più in basso a sinistra possibile, e cioè nessuno di essi può essere traslato ulteriormente, in basso o a sinistra, senza sovrapporsi ad altri pezzi e rispettando il taglio a ghigliottina (Figura 5.3). E' chiaro che una qualunque soluzione possa essere trasformata in una equivalente soluzione normalizzata, attraverso una opportuna traslazione dei suoi pezzi e dei suoi tagli, come in Figura 5.3, e dunque è sufficiente considerare solo le soluzioni normalizzate, come se non ne esistessero altre.



a) non normalizzata b) normalizzata

Figura 5.3 Soluzioni normalizzate

Le possibili coordinate di tagli in soluzioni normalizzate vengono chiamate coordinate normalizzate. Esse possono essere calcolate attraverso due procedure di programmazione dinamica (Hifi, 2004), che calcolano le tabelle di valori rispettivamente della funzione $g_{v,i}(x)$ associata ai tagli verticali e della funzione $g_{h,i}(y)$ associata ai tagli orizzontali. Per la prima funzione, i tipi di pezzo vengono ordinati con ampiezza non decrescente e la relazione ricorsiva utilizzata è:

$$g_{v,i}(x) = \begin{cases} 0 & \text{if} \quad i = 0 \quad \text{and} \quad x = 0 \\ \infty & \text{if} \quad i = 0 \quad \text{and} \quad x > 0 \\ g_{v,i-1}(x) & \text{if} \quad i > 0 \quad \text{and} \quad x < l_i \\ \min\left\{ g_{v,i-1}(x), \max\left\{ w_i, \min_{1 \le k \le \min\left\{ d_i, \left\lfloor \frac{x}{l_i} \right\rfloor \right\}} \left\{ g_{v,i-1}(x - k \cdot l_i) \right\} \right\} \right\} & \text{else} \end{cases} \tag{5.3}$$

Una analoga relazione è utilizzata per la funzione $g_{h,i}(y)$, per la quale i tipi di pezzo vengono però ordinati per lunghezza non decrescente.

Si denota con $N_v(l, w)$ l'insieme di coordinate normalizzate orizzontali per possibili tagli verticali su un rettangolo $R(l, w)$ e può essere ottenuto come $\{x \le l : g_{v,n}(x) \le w\}$. In modo simile, per i tagli orizzontali, l'insieme $N_h(l, w)$ di coordinate normalizzate verticali è uguale a $\{y \le w : g_{h,n}(y) \le l\}$. Si sottolinea che ognuno di questi insiemi contiene lo zero, dato che $g_{v,n}(0) = g_{h,n}(0) = 0$.

5.3.2 Coordinate normalizzate raggiungibili e knapsack function migliorata

Nel precedente Capitolo sono state introdotte e definite le coordinate normalizzate raggiungibili, con lo scopo di ottenere upper bounds migliorati, sfruttando un raffinamento della knapsack function. E' stata usata l'idea già data in Herz (1972) ed in Beasley (1985), sfruttandone però le conseguenze nel caso di domanda limitata.

Si considerino, per esempio, un'ampiezza w e due lunghezze x ed l tali che $0 < x < l$. La lunghezza l è detta *raggiungible* dalla lunghezza x, e la lunghezza x è detta *osservabile* dalla lunghezza l, rispetto all'ampiezza w, quando esistono due cutting patterns, rispettivamente $B^x = (b^x_1, \ldots, b^x_n)$ e $B^l = (b^l_1, \ldots, b^l_n)$, tali che le seguenti quattro condizioni siano valide

- $x = \sum_{i=1}^{n} \left(b_i^x \cdot l_i \right)$

- $l = \sum_{i=1}^{n} \left(b_i^l \cdot l_i \right)$

- $b^x_i \leq b^l_i \leq d_i$ per ogni $i = 1, \ldots, n$

- per ogni $i = 1, \ldots, n$, se $w_i > w$ allora $b^l_i = b^x_i = 0$.

Se valgono tutte e quattro, vuol dire che la lunghezza l può essere ottenuta come combinazione lineare di lunghezze dei tipi di pezzo, utilizzando solo i tipi estraibili dal rettangolo $R(l, w)$ e tenendo conto della domanda limitata, ed inoltre che la lunghezza x può essere ottenuta come somma delle lunghezze di un sottoinsieme dei pezzi coinvolti nella combinazione associata ad l. In pratica, a partire dalla lunghezza x, è possibile raggiungere la lunghezza l utilizzando solo lunghezze di pezzi disponibili non usati per ottenere x. Per questa motivazione si è utilizzato il termine "raggiungibile". Inoltre si possono definire gli insiemi $RN_v(w) = \{(x, l) : l$ è raggiungibile da x rispetto all'ampiezza $w\}$.

Per esempio, se per una data ampiezza w, vi sono quattro tipi di pezzo disponibili, con lunghezza rispettivamente 2, 3, 5 ed 8, tutti con domanda 1, allora 5 è raggiungibile da 2, dato che 2 + 3 = 5 ed 8 è raggiungibile da 5, poiché 5 + 3 = 8, ma 8 non è raggiungibile da 2, poiché un solo pezzo è disponibile per il tipo con lunghezza 3, e ciò vuol dire anche che, in generale, non vale alcuna proprietà transitiva. Essa varrebbe nel caso di domanda illimitata.

Per le coordinate normalizzate verticali, definizioni simili possono essere date. Sulla base di tali definizioni, nel Capitolo precedente, è stata proposta una formulazione rivisitata per la knapsack function, in modo tale che i corrispondenti valori rappresentino l'ottimo associato a problemi parziali con domanda sì illimitata, ma escludendo dei cutting schemes non compatibili con nessun soluzione normalizzata che sia fattibile rispetto alla domanda limitata. La formulazione proposta viene riportata di seguito.

$$f(l,w)=\begin{cases} 0 & \text{if} \quad l=0 \quad \text{or} \quad w=0 \\ \max\left\{f^*(l-1,w),\, f^*(l,w-1)\right\} & \text{if} \quad \begin{array}{c} l \notin N_v(L_S,w) \ \text{or} \\ w \notin N_h(l,W_S) \end{array} \\ \max\left\{f_0(l,w),\, f_v(l,w),\, f_h(l,w),\, f^*(l-1,w),\, f^*(l,w-1)\right\} & \text{else} \end{cases} \tag{5.3}$$

$$f_v(l,w)=\begin{cases} 0 & \text{if} \quad l<2\cdot l^w_{\min} \quad \text{or} \quad N_v^*(l,w)=\varnothing \\ \max\limits_{\substack{x\in N_v^*(l,w) \\ x\le\lfloor l/2\rfloor}} \left\{f(x,w)+f(l-x,w)\right\} & \text{else} \end{cases} \tag{5.4}$$

$$f_h(l,w)=\begin{cases} 0 & \text{if} \quad w<2\cdot w^l_{\min} \quad \text{or} \quad N_h^*(l,w)=\varnothing \\ \max\limits_{\substack{y\in N_h^*(l,w) \\ y\le\lfloor w/2\rfloor}} \left\{f(l,y)+f(l,w-y)\right\} & \text{else} \end{cases} \tag{5.5}$$

dove

$$l \in N_v(L_S, W_S) \ \text{ ed } \ w \in N_h(L_S, W_S)$$

Tali relazioni determinano il calcolo dei valori della knapsack function solo per le coppie (l, w) di coordinate normalizzate, cioè tali che $l \in N_v(L_S, W_S)$ e $w \in N_h(L_S, W_S)$. Inoltre, per ciascuna coppia (l, w) che non soddisfi tale proprietà, un upper bound per il relativo $R(l, w)$ può essere ottenuto come $f(l^*, w^*)$, dove l^* è la massima coordinata orizzontale non maggiore di l ed analogamente w^* è la massima coordinata normalizzata verticale non superiore a w.

I due valori l^w_{min} e $w^l_{\min}$ sono rispettivamente il minimo elemento positivo di $N_v(L_S, w)$ ed il minimo elemento positivo di $N_h(l, W_S)$. L'insieme $N_v^*(l, w)$ è l'insieme di lunghezze x osservabili da l rispetto a w, cioè uguale a $\{x : (x, l) \in RN_v(w)\}$. Analogamente, si ha l'uguaglianza $N_h^*(l, w) = \{y : (y, w) \in RN_h(l)\}$.

Nel precedente Capitolo è mostrato che è importante calcolare esplicitamente non tutti gli insiemi $N_v(l, w)$, ma solo gli insiemi $N_v(L_S, w)$ e solo per i valori di ampiezza w corrispondenti alle ampiezze dei pezzi. Inoltre, è importante calcolare gli insiemi $N_v^*(l, w)$ solo per le coppie (l, w) tali che l sia una coordinata normalizzata orizzontale e w l'ampiezza di qualche pezzo. Allo stesso modo, devono essere calcolati solo gli insiemi $N_h(l, W_S)$, per i valori di l che corrispondono alle lunghezze dei pezzi, e solo gli insiemi $N_h^*(l, w)$, tali che w è una coordinata normalizzata verticale ed l è la lunghezza di qualche pezzo.

E' inoltre mostrato che, se $w_1 < w_2$, allora $N_v(L_S, w_1) \subseteq N_v(L_S, w_2)$ e, per ciascuna coordinata normalizzata orizzontale l, $N_v^*(l, w_1) \subseteq N_v^*(l, w_2)$. Analoghe proprietà valgono per gli insiemi di coordinate normalizzate verticali.

5.3.3 Calcolo delle coordinate normalizzate raggiungibili

In seguito si propone una procedura ottimizzata per il calcolo degli insiemi $N_v^*(l, w)$ e degli insiemi $N_h^*(l, w)$. La procedura, inoltre, genera anche gli insiemi $N_v(l, w)$ ed $N_h(l, w)$.

Prima di spiegare la procedura è però necessario introdurre nuove definizioni. Quando un valore x è ottenuto come $\sum_{i=1}^{n}\left(b_i^x \cdot l_i\right)$, si dice che il cutting pattern $B^x = (b^x_1, \ldots, b^x_n)$ genera il valore x. Se per due cutting patterns $B^x = (b^x_1, \ldots, b^x_n)$ e $B^l = (b^l_1, \ldots, b^l_n)$, la relazione $b^x_i \le b^l_i$ è valida per $i = 1, \ldots, n$, si dice che B^x è contenuto in B^l e si denota ciò con $B^x \subseteq B^l$. In effetti

l'inclusione è la relazione fra gli insiemi di pezzi associati a questi due cutting patterns. Se inoltre il valore x generato da B^x è inferiore ad l, generato da B^l, si dice che B^x è strettamente contenuto in B^l, denotando ciò con $B^x \subset B^l$.

Ancora, data una coppia di cutting patterns B_1 e B_2, si definisce somma $B_1 \oplus B_2$ il cutting pattern che abbia come i-mo elemento la somma fra l'i-mo elemento di B_1 e l'i-mo elemento di B_2. Inoltre, se $B_2 \subseteq B_1$, si definisce inoltre la differenza $B_1 - B_2$ come il cutting pattern B_3 tale che $B_1 \oplus B_2 = B_3$.

Infine, si dice che un cutting pattern B è compatibile con l'ampiezza w quando, per ciascun tipo di pezzo t_i che abbia ampiezza non maggiore di w, il corrispondente elemento b_i di B sia nullo.

Nel Box 5.3 si riporta la procedura **PV** che costruisce gli insiemi $N^*_v(l, \underline{w}_j)$, dove $\underline{w}_j$ è una fra le ampiezze dei pezzi. Si suppone che i tipi di pezzo siano indicizzati per ampiezza non decrescente. Come detto prima, la procedura permette di costruire anche gli insiemi $N_v(L_S, \underline{w}_j)$ e l'insieme $N_v(L_S, W_S)$. Un' analoga procedura **PH** può essere usata per costruire gli insiemi di coordinate normalizzate verticali $N^*_h(\underline{l}_j, w)$, dove $\underline{l}_j$ denota una fra le possibili lunghezze di pezzi, e gli insiemi $N_h(\underline{l}_j, W_S)$, nonché l'insieme $N_h(L_S, W_S)$.

Si utilizzano le notazioni $N_v'(L_S, W_S)$ ed $N^*_v'(l, w)$ per insiemi transitori. Ciascuno di essi, al termine della procedura, corrisponde alla definizione rispettivamente di $N_v(L_S, W_S)$ e di $N^*_v(l, w)$.

La procedura **PV** comincia inizializzando $N_v'(L_S, W_S)$ pari ad un insieme che contenga solo lo zero, $N^*_v'(0, 0)$ all'insieme vuoto ed il parametro $\underline{w}$ a zero. Tale parametro rappresenta l'ampiezza correntemente considerata fra quelle possibili dei pezzi.

Il ciclo principale è sui tipi di pezzo e, per ciascun tipo, i corrispondenti aggiornamenti vengono realizzati., secondo le ulteriori combinazioni di lunghezze di pezzi possibili utilizzando anche il tipo corrente t_i . Si definisce BS_i l'insieme di bounded cutting patterns B che abbiano gli elementi b_j nulli, con $j = i+1, \ldots, n$, e generanti lunghezze non superiori alla lunghezza L_S della lastra.

Per ciascun tipo t_i, vengono realizzati due passi principali. Il primo (step I nel Box 5.3) verifica se una nuova ampiezza di pezzi debba essere considerata, vale a dire se $w_i > \underline{w}$. Se è così, allora l'insieme $N_v(L_S, \underline{w})$ è uguale all'insieme transitorio $N_v'(L_S, W_S)$, poiché tutte le combinazioni lineari di lunghezze di pezzi, corrispondenti ai cutting patterns di BS_{i-1} , sono state considerate, tenendo conto solo dei pezzi con ampiezza non maggiore di $\underline{w}$, come si dimostrerà a breve nel seguito.

Inoltre, se $w_i > \underline{w}$, per ciascun valore l del corrente insieme $N_v'(L_S, W_S)$, l'insieme $N^*_v'(l, \underline{w})$ è diventato definitivo. I suoi elementi sono assegnati anche all'insieme $N^*_v'(l, w_i)$, poiché queste combinazioni lineari sono chiaramente compatibili anche con la nuova ampiezza w_i. Per tutti gli altri valori l che saranno contenuti in $N_v(L_S, W_S)$ al termine della procedura, l'insieme $N^*_v(l, \underline{w})$ è implicitamente considerato vuoto. Il parametro $\underline{w}$ è poi aggiornato se $i < n+1$, altrimenti il ciclo principale termina e lo step I è stato eseguito solo al fine di realizzare le assegnazioni finali associate alla massima ampiezza possibile fra quelle dei pezzi. Si utilizza quindi un valore fittizio di ampiezza, denotato con w_{n+1} e pari a $W_S + 1$.

Per capire come lavora lo step II, è necessario spiegare alcune proprietà. Si supponga induttivamente che i precedenti step, realizzati per i tipi $t_1, t_2, \ldots, t_{i-1}$, abbiano permesso di trovare correttamente tutte le coppie (x, l) tali che $l \le L_S$ è raggiungibile da $x > 0$ rispetto a $\underline{w}$, secondo la sola disponibilità dei pezzi di tipo $t_1, t_2, \ldots, t_{i-1}$ e che, per ciascuna di queste coppie, il valore x sia stato aggiunto all'insieme $N^*_v'(l, \underline{w})$.

```
Procedure PV
    Set N_v'(L_S, W_S) = {0} ; Set N*_v'(0, 0) = ∅ ; Set w = 0
    For i = 1 To n+1 Do //ciclo principale sulle ampiezze dei pezzi
        //Step I.
        If w_i > w Then
            Set N_v(L_S, w) = N_v'(L_S, W_S)
            For Each l ∈ N_v(L_S, w) Do Set N*_v'(l, w_i) = N*_v(l, w) = N*_v'(l, w) End For
            If i < n+1 Set w = w_i Else Exit (dal for su i) End If
        End If

        //Step II.
        Set SN = N_v'(L_S, W_S) //Saved set
        For Each l ∈ SN Do Set SDN(l) = N*_v'(l, w) End For //Saved Delta sets

        For k = 1 To min{ d_i, ⌊ L_S / l_i ⌋ } Do
            For Each l ∈ SN Do Set DN(l + l_i) = ∅ End For //inizializzazione Delta sets
            Set NSN = ∅ // inizializzazione Next Saved set

            For Each l ∈ SN such that l + l_i ≤ L_S Do //ciclo sulle coordinate salvate
                //Step II.a. l + l_i è una nuova potenziale coordinata normalizzata
                Add l + l_i to the set N_v'(L_S, W_S)
                Add l + l_i to the set NSN

                //Step II.b. l + l_i è raggiungibile da l.
                If l ∉ N*_v'(l + l_i, w) Then
                    Add l to the set DN(l + l_i)
                    Add l to the set N*_v'(l + l_i, w)
                End If

                For Each x ∈ SDN(l) Do //ciclo sulle coordinate salvate x osservabili
                                       //da l rispetto a w.
                    //Step II.c. l + l_i è raggiungibile da x.
                    If x ∉ N*_v'(l + l_i, w) Then
                        Add x to the set DN(l + l_i)
                        Add x to the set N*_v'(l + l_i, w)
                    End If

                    //Step II.d. l + l_i è raggiungibile da x + l_i.
                    If x + l_i ∉ N*_v'(l + l_i, w) Then
                        Add x + l_i to the set DN(l + l_i)
                        Add x + l_i to the set N*_v'(l + l_i, w)
                    End If
                End For
            End For

            //Saved set e Saved Delta sets
            Set SN = NSN
            For Each l ∈ SN Do Set SDN(l) = DN(l) End For
        End For
    End For
    N_v(L_S, W_S) = N_v'(L_S, W_S) //settaggio finale
End Procedure
```

Box 5.3 Procedura **PV**

Ciò vuol dire che tutti i possibili bounded cutting patterns B di BS_{i-1}, cioè tali che $b_j = 0$ per $j = i, \ldots, n$, sono stati considerati e, per ciascuno di tali B, generante il generico valore l, tutti i corrispondenti cutting patterns B^x strettamente contenuti in B sono stati considerati, dato che, per ciascun B^x, generante il generico valore x, la coppia di lunghezze (x, l) è stata trovata. Questa assunzione è ovviamente soddisfatta per $i = 1$.

Aggiungendo anche la disponibilità dei d_i pezzi di tipo t_i, gli ulteriori cutting patterns da considerare, appartenenti cioè a BS_i ma non a BS_{i-1}, hanno l'i-mo elemento positivo e, per $j = i+1, \ldots, n$, il j-mo elemento pari a zero. Ciascuno di essi può quindi essere ottenuto attraverso una somma del tipo $B \oplus \Delta_{i,k}$, dove B è un cutting pattern di BS_{i-1}, precedentemente ottenuto utilizzando solo i tipi di pezzo $t_1, t_2, \ldots, t_{i-1}$, cioè senza alcun pezzo di tipo t_i, e $\Delta_{i,k}$ è il cutting pattern che ha tutti elementi nulli, tranne l'i-mo, che ha valore k, con $1 \le k \le d_i$, a rappresentare così un insieme di pezzi contenente solo k pezzi di tipo t_i, con cardinalità k, positiva e non superiore a d_i.

Questa condizione permette di ottenere, dalle somme $B \oplus \Delta_{i,k}$, solo bounded cutting patterns, cioè senza eccedere la domanda di alcun tipo di pezzi. Da notare anche che è inutile considerare per k valori maggiori di $\lfloor L_S / l_i \rfloor$, poiché L_S verrebbe ecceduta dal valore di lunghezza generato da un qualsiasi cutting pattern che avesse un i-mo elemento maggiore di $\lfloor L_S / l_i \rfloor$, per cui si utilizza il valore $d'_i = \min\{d_i, \lfloor L_S / l_i \rfloor\}$ al posto di d_i.

Si denoti, per $k \ge 1$, con $B_{i,k}$ un generico cutting pattern di BS_i, ottenibile dunque attraverso una somma $B \oplus \Delta_{i,k}$, con $B \in BS_{i-1}$ secondo quanto spiegato prima. Esso può anche essere ottenuto attraverso la somma $B_{i,(k-1)} \oplus \Delta_{i,1}$, dove $B_{i,(k-1)}$ è il cutting pattern strettamente contenuto in $B_{i,k}$, avente gli stessi elementi di $B_{i,k}$ tranne l'i-mo, che ha valore $k - 1$ anziché k. Di conseguenza, per $k = 1$, $B_{i,0}$ denota un generico cutting pattern di BS_{i-1}. Si possono dunque ottenere le lunghezze generate da ogni possibile $B_{i,k}$ di BS_i aggiungendo la lunghezza l_i del tipo t_i, alla lunghezza generata dal corrispondente cutting pattern $B_{i,(k-1)}$. Per $k \ge 1$ si denota con $BS_{i,k}$ il sottoinsieme di BS_i che contiene i cutting patterns genericamente denotati sopra con $B_{i,k}$, e con $BS_{i,0}$ l'insieme BS_{i-1}. Quindi $BS_i = \bigcup_{k=0}^{d'_i} BS_{i,k}$.

Per tutti i valori positivi di k, lo step II della procedura **PV** congela nell'insieme *SN*, chiamato *Saved set*, l'insieme corrente di coordinate normalizzate orizzontali generate dai cutting patterns dell'insieme $BS_{i,(k-1)}$, ovvero ottenibili usando esattamente $k - 1$ pezzi di tipo t_i e nessun altro pezzo di altri tipi t_j con $j = i+1, \ldots, n$.

Infatti, sulla base della discussione precedente, è sufficiente utilizzare solo i valori l generati dai cutting patterns di $BS_{i,(k-1)}$, poiché il corrispondente insieme di valori, ottenibili attraverso somme del tipo $l + l_i$ per ciascuno di tali valori l, è l'insieme dei valori generati dai cutting patterns di $BS_{i,k}$.

Quindi, prima di realizzare gli aggiornamenti per il valore $k = 1$, la procedura **PV** setta *SN* pari ad $N_v'(L_S, W_S)$, che contiene correntemente i valori generati dai cutting patterns di $BS_{i,0}$, cioè BS_{i-1}. Attraverso lo step II.a, i valori $l + l_i$ generati dai cutting patterns di $BS_{i,k}$ sono aggiunti all'insieme *NSN*, chiamato *New Saved set* ed imposto inizialmente vuoto. Prima di aumentare il valore di k, *SN* è posto uguale ad *NSN* e quindi, per induzione, quando k cresce, *SN* contiene i valori generati dai cutting patterns di $BS_{i,(k-1)}$.

Finora il discorso ha riguardato solo la generazione di tutte le coordinate normalizzate orizzontali. Nel seguito si tratta l'aggiornamento degli insiemi $N^*_v'(l, \underline{w})$. Informalmente, per ciascun valore di k e per ciascun valore di l generato da un cutting pattern di $BS_{i,(k-1)}$, si può affermare che due proprietà valgono per ciascun $x \in FN^*_v'(l, \underline{w})$, dove $FN^*_v'(l, \underline{w})$ rappresenta l'insieme congelato corrispondente all'insieme $N^*_v'(l, \underline{w})$ subito dopo gli aggiornamenti realizzati per il precedente valore di k.

Da notare che $x \in FN^*_v'(l, \underline{w})$ significa che l è raggiungibile da x rispetto a $\underline{w}$, considerando non più di $k-1$ pezzi di tipo t_i. Perciò $l + l_i$ è raggiungibile sia da x che da $x + l_i$ rispetto a $\underline{w}$ quando si considerino k pezzi di tipo t_i, e dunque x ed $x + l_i$ vanno aggiunti all'insieme $N^*_v'(l + l_i, \underline{w})$.

Nel seguito questa proprietà viene provata formalmente spiegando come la procedura **PV** realizza gli aggiornamenti. Si consideri preliminarmente che, per ciascun cutting pattern $B_{i,k}$ di $BS_{i,k}$, generante il generico valore l, bisogna trovare tutti i cutting patterns B strettamente contenuti in $B_{i,k}$ e, per ciascuno di tali B, bisogna aggiungere il valore x, da esso generato, all'insieme $N^*_v'(l, \underline{w})$.

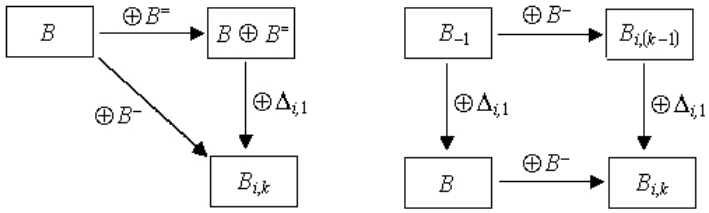
Si consideri dunque uno fra i cutting patterns genericamente denotati con B e si denoti con B^- il cutting pattern ottenuto come differenza fra $B_{i,k}$ e B, cioè tale che $B_{i,k} = B \oplus B^-$. Siano inoltre $(\bar{b}_1, ..., \bar{b}_n)$ gli elementi di B^- e si denoti con $\bar{b}_{\Sigma(i-1)}$ la somma $\sum_{j=1}^{i-1} \bar{b}_j$. Sia per $\bar{b}_{\Sigma(i-1)}$ che per $\bar{b}_i$, si pososno distinguere tre casi alternativi: uguale a 0, uguale ad 1 ed, infine, maggiore di 1. Di conseguenza, per B^-, vi sono nove possibili casi, dato che $\bar{b}_j = 0$ per $j = i+1, \ldots, n$. Si sottolinea tuttavia che il caso $\bar{b}_{\Sigma(i-1)} = \bar{b}_i = 0$ non è possibile. Gli altri otto casi possono essere riarrangiati accorpandoli in tre gruppi. Il primo è associato alle condizioni $\bar{b}_i = 1$ e $\bar{b}_{\Sigma(i-1)} = 0$, il secondo alle condizioni $\bar{b}_i \geq 1$ e $\bar{b}_i + \bar{b}_{\Sigma(i-1)} \geq 2$ ed, infine, il terzo alle condizioni $\bar{b}_i = 0$ e $\bar{b}_{\Sigma(i-1)} > 0$, dove i raggruppamenti sono scelti al fine di semplificare la spiegazione e la comprensione.

Nel primo caso, $B^- = \Delta_{i,1}$, e dunque $B \in BS_{i,(k-1)}$. Quindi il valore l generato da B è contenuto nell'insieme SN e la procedura **PV** tratta questo caso attraverso lo step II.b. Inoltre, in questo caso, l'insieme di pezzi associato a B^- ha un solo pezzo di tipo t_i e nessun pezzo di nessun altro tipo.

Nel secondo caso (Figura 5.4a), l'insieme di pezzi associato a B^- ha almeno un pezzo di tipo t_i ed, inoltre, almeno un altro pezzo di un qualsiasi tipo fra $t_1, \ldots, t_i$. Se si denota con $B^=$ il cutting pattern tale che $B^- = B^= \oplus \Delta_{i,1}$, allora $B_{i,k} = B \oplus (B^= \oplus \Delta_{i,1}) = (B \oplus B^=) \oplus \Delta_{i,1}$ e cioè $B \oplus B^= \in BS_{i,(k-1)}$, dove, sulla base delle assunzioni relative a questo secondo caso, almeno un elemento di $B^=$ è positivo ed il suo i-mo elemento, denotato con $\bar{\bar{b}}_i$, è inferiore a k. ciò significa che il valore di lunghezza l generato da $B \oplus B^=$ è contenuto nell'insieme SN ed il valore di lunghezza x generato da $B^=$, strettamente contenuto in $B \oplus B^=$, è positivo ed appartiene all'insieme $N^*_v'(l, \underline{w})$ al termine degli aggiornamenti realizzati per il precedente valore di k.

Nel terzo caso (Figura 5.4b), l'insieme di pezzi associati a B^- non ha pezzi di tipo t_i, ma solo pezzi di altri tipi e quindi il cutting pattern B appartiene a $BS_{i,k}$ poiché il suo i-mo elemento è uguale a k. Denotando con $B_{i,(k-1)}$ il cutting pattern tale che $B_{i,k} = B_{i,(k-1)} \oplus \Delta_{i,1}$ e, similmente, con B_{-1} il cutting pattern tale che $B = B_{-1} \oplus \Delta_{i,1}$, si ottiene $B_{i,(k-1)} = B_{-1} \oplus B^-$, cioè la lunghezza l generata da $B_{i,(k-1)}$ è raggiungibile dalla lunghezza x generata da B_{-1} e la lunghezza generata da B è ottenibile come somma fra x e la lunghezza l_i associata al tipo t_i e generata da $\Delta_{i,1}$.

Come per il secondo caso, l appartiene all'insieme SN ed x appartiene all'insieme $N^*_v'(l, \underline{w})$ per cui, per trattare sia il secondo che il terzo caso, è sufficiente scandire tutti i valori x, che appartengono all'insieme $N^*_v'(l, \underline{w})$ all'inizio del k-mo ciclo, ed aggiungere entrambi i valori x ed $x + l_i$ all'insieme $N^*_v'(l + l_i, \underline{w})$, come appunto la procedura **PV** fa attraverso gli step II.c e II.d.



a) secondo caso b) terzo caso

Figura 5.4 Dettaglio della procedura **PV**

Per comprendere il ruolo degli insiemi *SDN*(*l*) e *DN*(*l*), si sottolinea innanzitutto che il valore l_i non dipende da *k*, e quindi è inutile aggiungere i valori *x* ed $x + l_i$ all'insieme $N^*_v{}'(l + l_i, \underline{w})$ per più di una volta. Per tener conto di questa considerazione, la procedura **PV** utilizza gli insiemi *SDN*(*l*), chiamati *Saved Delta sets*, affinché contengano tutti i nuovi valori aggiunti al corrispondente insieme $N^*_v{}'(l, \underline{w})$ durante gli aggiornamenti realizzati per il precedente valore di *k*. Tale insieme è inizialmente posto pari ad $N^*_v{}'(l, \underline{w})$. Per ciascun valore di *k*, gli insiemi *DN*(*l*), chiamati *Delta sets*, sono inizializzati vuoti e sono utilizzati per contenere tutti i nuovi valori del corrispondente insieme $N^*_v{}'(l, \underline{w})$. Prima di aumentare il valore di *k*, ciascun *SDN*(*l*) è posto uguale al corrispondente *DN*(*l*). Da notare che sono considerati solo gli insiemi *SDN*(*l*) associati ai valori di *l* appartenenti al nuovo insieme *SN*. Infatti, per il nuovo valore di *k*, solo tali valori vengono poi utilizzati per realizzare gli aggiornamenti. Di conseguenza, vengono trattati solo gli insiemi *DN*($l + l_i$), per ciascun $l \in SN$, poiché solo i valori $l + l_i$ potranno essere inseriti nell'insieme *NSN*.

Un nota finale molto importante è che la procedura **PV** è scalabile, mentre la procedura dinamica basata sulla relazione (5.2) non gode di tale proprietà. Dunque, se le dimensioni di tutti i pezzi e della lastra vengono incrementate di un fattore intero *m*, il tempo computazionale necessario alla procedura classica aumenta, mentre per la procedura **PV** esso resta inalterato.

5.4 Procedura unificata

In questo paragrafo viene spiegato come unificare i miglioramenti proposti per gli upper bounds con la procedura ottimizzata **P**. Nel primo sotto-paragrafo si fa questo per la versione un-constrained del problema, cioè con domande illimitate di pezzi, e nel secondo si descrive come generalizzare per la versione constrained, cioè con domande limitate. Si ricorda che l'interesse è rivolto al caso constrained, ma l'analisi del caso un-constrained permette di ricavare utili considerazioni in merito.

5.4.1 Caso a domanda illimitata

Per il caso di domande illimitate, si definisce $NU_v(L_S, w)$ come l'insieme di coordinate normalizzate orizzontali compatibili con w, cioè analoghe all'insieme $N_v(L_S, w)$ del caso di domanda limitata. Si noti che, in presenza di domande illimitate, data una qualunque coppia di valori x_1 ed x_2 appartenenti ad $NU_v(L_S, w)$, anche $x_1 + x_2$ vi appartiene. Inoltre, se si denotano con $NU^*_v(l, w)$ gli insiemi di coordinate normalizzate osservabili associate alla domanda illimitata, allora valgono le due seguenti proprietà. Se x_1 ed x_2 appartengono ad $NU_v(L_S, w)$, allora sia x_1 che x_2 appartengono anche all'insieme $NU^*_v(x_1 + x_2, w)$, vale a dire che $x_1 + x_2$ è raggiungibile da entrambe, rispetto a w. Inoltre, se $l_1 \in NU^*_v(l_2, w)$ ed $l_2 \in NU^*_v(l_3, w)$, allora $l_1 \in NU^*_v(l_3, w)$. In generale, queste proprietà non valgono nel caso di domande limitate.

Analogamente, per le coordinate di tagli orizzontali, si possono definire gli insiemi $NU_h(w, L_S)$ e gli insiemi $NU^*_h(l, w)$, per i quali valgono analoghe proprietà.

Si ricorda che la procedura **P** è stata sviluppata per il caso di domande illimitate e si sottolinea che la condizione $C_{v,1}$, cioè $x_1 = \lambda(x_1, w)$, può essere soddisfatta solo da valori x_1 che appartengano ad $NU_v(L_S, w)$. Infatti, per ogni altro valore x_1 , $f(x_1, w) = f(x_1 - 1, w)$ e dunque $\lambda(x_1, w) = 0$.

Ciò vuol dire che la procedura **P** incorpora implicitamente la computazione delle coordinate normalizzate compatibili, cioè appartenenti ad $NU_v(L_S, w)$, per un qualunque valore di w, senza la necessità di realizzare una computazione che si basi sulla relazione (5.2). Simili considerazioni valgono per la condizione $C_{h,1}$.

Come spiegato alla fine del paragrafo 5.3, le procedure **PV** e **PH** sono scalabili e dunque la prima idea da utilizzare è quella di ottenere le coordinate normalizzate con le procedure **PV** e **PH** e realizzare, nella procedura **P**, i cicli esterni non su tutti i possibili valori di x_2 ed y_2 , ma solo sulle coordinate normalizzate, cioè rispettivamente con $x_2 \in NU_v(L_S, W_S)$ e con $y_2 \in NU_h(L_S, W_S)$.

Inoltre, si potrebbe pensare di realizzare i cicli interni sulle lunghezze solo se $x_2 \in NU_v(L_S, y_2)$ e solo sui valori x_1 che pure appartengano all'insieme $NU_v(L_S, y_2)$, poiché questi valori sono quelli tali che $x_2 \in NU^*_v(x_1 + x_2, y_2)$. L'insieme $\Lambda(y_2)$ può ancora essere usato per le strategie di anti-simmetria, per cui x_1 deve appartenere all'insieme intersezione fra $NU_v(L_S, y_2)$ e $\Lambda(y_2)$, senza superare $\lambda(x_2, y_2)$.

In modo simile, si può pensare di realizzare il ciclo interno sulle ampiezze solo nel caso in cui $y_2 \in NU_h(x_2, W_S)$ e solo sui valori di y_1 appartenenti all'insieme intersezione fra $NU_h(x_2, W_S)$ ed $\Omega(x_2)$, senza superare $\omega(x_2, y_2)$. Nel seguito si denota con $\mathbf{P}^+$ la procedura così ottenuta per il caso di domande illimitate.

Si ricorda che l'obiettivo finale è discutere la computazione degli upper bounds nel caso constrained, cioè con domande dei pezzi limitate, ma la procedura $\mathbf{P}^+$ è utile anche nel caso constrained. Infatti se, per ciascun tipo t_i , la domanda d_i è maggiore o uguale al massimo fra $\lfloor L_S / l_i \rfloor$ e $\lfloor W_S / w_i \rfloor$, allora gli insiemi di coordinate normalizzate sono gli stessi rispetto al

caso un-constrained. Questo caso particolare viene qui denotato come *semi-constrained*. La procedura $\mathbf{P}^+$ può dunque essere usata nel caso semi-constrained.

5.4.2 Caso a domanda limitata

In questo paragrafo si dimostra che la procedura $\mathbf{P}^+$, in generale, non è adatta al caso constrained per ottenere i valori definiti dalla formulazione (5.3)-(5.5). Innanzitutto verranno indicate le modifiche di base necessarie per il caso constrained e poi si mostrerà nel seguito che esse non sono sufficienti, attraverso tre esempi che inducono ulteriori utili definizioni. Una volte introdotte queste ultime, si potranno generalizzare e formalizzare le considerazioni derivanti dai tre esempi per dare così i fondamenti teorici necessari a descrivere brevemente le modifiche da apportare alla procedura.

Ovviamente, la prima modifica da apportare consiste nell'utilizzare, per esempio per le coordinate normalizzate orizzontali, gli insiemi $N_v(L_S, w)$ ed $N^*_v(l, w)$ al posto degli insiemi $NU_v(L_S, w)$ ed $NU^*_v(l, w)$. Dobbiamo però considerare che la procedura $\mathbf{P}^+$ sfrutta proprietà valide per gli insiemi $NU_v(L_S, w)$ ed $NU^*_v(l, w)$, ma non per gli insiemi $N_v(L_S, w)$ ed $N^*_v(l, w)$. Per aggiustare in modo opportuno la procedura, è necessario cambiare il modo in cui viene realizzato il ciclo interno sulle lunghezze, allo scopo di utilizzare tutti i valori x_1 tali che $x_1 + x_2$ sia raggiungibile da x_2 rispetto al valore corrente di w.

La procedura $\mathbf{P}^+$ ottiene ciò, nel caso di domanda illimitata, utilizzando, per ogni valore di x_2, i valori di x_1 appartenenti all'insieme $NU_v(L_S, y_2)$, ma se $x_1 \in N_v(L_S, y_2)$ non si può similmente concludere che $x_2 \in N^*_v(x_1 + x_2, y_2)$. Quindi la procedura modificata deve considerare esplicitamente gli insiemi di coordinate raggiungibili.

Tuttavia può essere difficile usare gli insiemi $N^*_v(l, w)$, che associano, ad ogni valore l, i valori $x < l$ da cui l è raggiungibile, vale a dire i valori osservabili da l. Vanno dunque esplicitamente costruiti ed utilizzati gli insiemi di valori raggiungibili.

A tal fine, si definiscono gli insiemi $N^{**}_v(x, w)$ come $\{l : (x, l) \in RN_v(w)\}$ e, per le coordinate verticali, gli insiemi $N^{**}_h(l, y)$ come $\{w : (y, w) \in RN_h(l)\}$. Le procedure **PV** e **PH** possono essere facilmente modificate per ottenere questi insiemi al posto degli insiemi N^* oppure contemporaneamente ad essi. Basta infatti che, per ogni valore x da aggiungere all'insieme $N^*_v(l, w)$, si aggiunga il valore l all'insieme $N^{**}_v(x, w)$.

Si introducono inoltre l'insieme $\underline{N}^*_v(l, w) = \{x \in N^*_v(l, w) : x \le \lfloor l / 2 \rfloor\}$ e l'insieme $\underline{N}^{**}_v(l, w) = \{m \in N^{**}_v(l, w) : l \le \lfloor m / 2 \rfloor\}$. La loro definizione è utile poiché la strategia di anti-simmetria utilizzata di fondo consiste nell'imporre $x_1 \le x_2$. Da notare che ciascun insieme $N^*_v(l, w)$ è semplicemente pari all'insieme unione $\underline{N}^*_v(l, w) \cup \{l - x : x \in \underline{N}^*_v(l, w)\}$ ed una simile proprietà vale per ciascun insieme $N^{**}_v(l, w)$ rispetto al relativo insieme $\underline{N}^{**}_v(l, w)$. Analoghe definizioni di $\underline{N}^*_h(l, w)$ ed $\underline{N}^{**}_h(l, w)$ possono essere introdotte per le coordinate normalizzate verticali.

Quindi, la procedura $\mathbf{P}^+$ può facilmente essere modificata utilizzando, nel ciclo interno sulle lunghezze, i soli valori di lunghezza dell'insieme ottenuto come intersezione fra i due insiemi $\{m - x_2 : m \in \underline{N}^{**}_v(x_2, y_2)\}$ e $\Lambda'(y_2)$, ed usare, nel ciclo interno sulle ampiezze, i valori di ampiezza appartenenti all'insieme ottenuto come intersezione fra i due insiemi $\{m - y_2 : m \in \underline{N}^{**}_h(x_2, y_2)\}$ ed $\Omega'(x_2)$. La procedura modificata è denotata con $\mathbf{P}^{++}$.

5.4.2.1 *Esempi di non correttezza*

Si mostrano nel seguito tre esempi peculiari per provare che la procedura $\mathbf{P}^{++}$ necessita di essere ulteriormente modificata, per poi descrivere più avanti come modificarla.

Come spiegato nel paragrafo 5.2, la posizione $\lambda(l, w) = 0$ quando $f(l, w) = f(l - 1, w)$ ha senso poiché in tal caso ogni somma ottenibile utilizzando $f(l, w)$ è ottenibile utilizzando

anche $f(l-1, w)$. Tuttavia, nel caso di domande limitate, non è certo che il valore $f(l-1, w)$ possa essere utilizzato al posto di $f(l, w)$. Per provare ciò, si consideri il seguente Esempio 1, con due tipi di pezzo, rispettivamente t_1 con $l_1 = 4$, $w_1 = 2$, $d_1 = 1$, $\pi_1 = 2$ e t_2 con $l_2 = 5$, $w_2 = 1$, $d_2 = 1$, $\pi_2 = 1$, ed infine una lastra di dimensioni $L_S = 9$ e $W_S = 2$. In questo caso, si ottiene $N_v(9, 1) = \{5\}$, $N_v(9, 2) = \{4, 5, 9\}$, $N^*_v(4, 2) = N^*_v(5, 2) = \varnothing$ ed $N^*_v(9, 2) = \{4, 5\}$, cioè la coordinata 9 è raggiungibile sia da 4 che da 5, rispetto all'ampiezza 2. Dunque, dalle relazioni (5.3)-(5.5), si ottiene $f(5, 1) = 1$ ed $f(5, 2) = f(4, 2) = 2$.

Se si setta $\lambda(5, 2) = 0$ a causa dell'uguaglianza $f(5, 2) = f(4, 2)$, allora non si permette di realizzare la somma $f(5, 2) + f(4, 2)$. Anche la somma $f(4, 2) + f(4, 2)$ non viene realizzata, poiché 8 non è una coordinata normalizzata orizzontale. La conseguenza è che $f(9, 2)$ è erroneamente posto pari ad $f(5, 2)$. Si noti che lo stesso errore può avvenire anche se 8 fosse una coordinate normalizzata orizzontale, ma non raggiungibile da 4, per esempio aggiungendo un terzo tipo di pezzo, vale a dire t_3 con $l_3 = 8$, $w_3 = 2$, $d_3 = 1$ e $\pi_3 = 1$.

Per superare questo problema, si potrebbe utilizzare la posizione $\lambda(l, w) = l$ appunto anche nel in cui valga $f(l, w) = f(l-1, w)$, ed analogamente porre $\omega(l, w) = w$ anche quando $f(l, w) = f(l, w-1)$, riducendo però il vantaggio di utilizzare gli insiemi Λ ed Ω, dato che questi ultimi possono diventare più grandi. Tuttavia la strategia che si proporrà più avanti sarà differente.

L'Esempio 2 ha invece i seguenti dati di input. Una lastra con dimensioni $L_S = 14$ e $W_S = 1$ e tre tipi di pezzo, t_1 con $l_1 = 2$, $w_1 = 1$, $d_1 = 1$, $\pi_1 = 3$, t_2 con $l_2 = 4$, $w_2 = 1$, $d_2 = 2$, $\pi_2 = 8$ e t_3 con $l_3 = 5$, $w_3 = 1$, $d_3 = 2$, $\pi_3 = 9$. In tal caso $N_v(14, 1) = \{2, 4, 5, 8, 9, 10, 11, 12, 13, 14\}$, $N^*_v(8, 1) = \{4\}$, $N^*_v(9, 1) = \{4, 5\}$, $N^*_v(10, 1) = \{2, 4, 5, 6\}$ poiché 10 è ottenibile sia con la somma 2+4+4 che con la somma 5+5, $N^*_v(11, 1) = \{2, 4, 5, 6, 7, 9\}$, $N^*_v(12, 1) = \{2, 5, 7, 10\}$, $N^*_v(13, 1) = \{4, 5, 8\}$ ed infine $N^*_v(14, 1) = \{4, 5, 9, 10\}$ poiché 14 è ottenibile solo con la somma 5+5+4.

Vengono quindi ottenuti i seguenti valori. I valori $f(2, 1) = f_0(2, 1) = 3$ e, similmente, $f(4, 1) = f_0(4, 1) = 8$ ed $f(5, 1) = f_0(5, 1) = 9$, per cui $\lambda(2, 1) = 2$, $\lambda(4, 1) = 4$ e $\lambda(5, 1) = 5$. Il valore $f(6, 1) = 11$, con la somma $f(2, 1) + f(4, 1)$, per cui $\lambda(6, 1) = 2$. Il valore $f(7, 1) = 12$, attraverso la somma $f(2, 1) + f(5, 1)$, per cui $\lambda(7, 1) = 2$. Il valore $f(8, 1) = 16$, con la somma $f(4, 1) + f(4, 1)$, per cui $\lambda(8, 1) = 4$. Il valore $f(9, 1) = 17$, attraverso la somma $f(4, 1) + f(5, 1)$, per cui $\lambda(9, 1) = 4$.

Inoltre il valore $f(10, 1) = 19$ è ottenuto con la somma $f(2, 1) + f(8, 1)$, per cui $\lambda(10, 1) = 2$. Quindi, la somma $f(4, 1) + f(10, 1) = 27$ non è permessa, dato che $4 > \lambda(10, 1)$ e cioè la condizione $C_{v,2}$ non è soddisfatta con i valori $x_2 = 10$ ed $x_1 = 4$. Anche la somma $f(5, 1) + f(9, 1) = 26$ non viene realizzata poiché $5 > \lambda(9, 1)$, per cui $f(14, 1) = f(13, 1) = 25$, dove $f(13, 1)$ è ottenuto con la somma $f(4, 1) + f(9, 1)$.

Comunque, la soluzione ottima fattibile per l'intera lastra estrae due pezzi di tipo t_3 ed un pezzo di tipo t_2, ed ha dunque valore 26, e cioè il valore $f(14, 1)$ ottenuto non è un upper bound valido. La motivazione è che la condizione $C_{v,2}$ non funziona bene nel caso di domande limitate. Infatti, in questo caso, si ha $\lambda(10, 1) = 2$, ma non tutti i valori raggiungibili da 10 sono raggiungibili anche da 2. Come spiegato nel paragrafo 5.2, la condizione $C_{v,2}$ è applicata per evitare di realizzare più somme equivalenti e, per esempio, essa funziona bene per il valore $f(13, 1)$ poiché evita che si realizzi la somma $f(5, 1) + f(8, 1) = 2$, equivalente alla somma $f(4, 1) + f(9, 1)$. Si noti che il valore 13 è ottenibile solo con la somma 4+4+5, mentre 10 è ottenibile con due somme, cioè 2+4+4 e 5+5, ma solo la seconda ha addendi coinvolti anche nella somma 4+5+5, che è la sola a dare 14 come risultato. Quindi, in questo esempio, il valore di $\lambda(10, 1) = 2$ non è per niente collegato al valore 14 raggiungibile da 10.

Inoltre, anche se si modificasse l'Esempio 2 aggiungendo un quarto tipo di pezzi, cioè t_4 con $l_4 = 12$, $w_4 = 1$, $d_4 = 1$ e $\pi_4 = 1$, si otterrebbe un nuovo esempio dove 14 sarebbe

raggiungibile anche da 2, ma tutto il discorso resterebbe valido, dato che la somma $f(2, 1) + f(12, 1)$ dà 25 come risultato.

Per ottenere un risultato corretto dovremmo aggiungere un ulteriore quinto tipo di pezzi, cioè t_5 con $l_5 = 6$, $w_5 = 1$, $d_5 = 1$ e $\pi_5 = 1$, poiché si otterrebbe non solo che 14 sarebbe raggiungibile da 2 per l'aggiunta del tipo t_4, ma anche che 12, cioè 14 – 2, sarebbe raggiungibile da 4, cioè 14 – 10. Infatti, 12 diventerebbe ottenibile con la somma 2+4+6, e la nuova somma $f(4, 1) + f(8, 1) = 24$ verrebbe permessa facendo così ottenere $f(12, 1) = 24$. Di conseguenza, si avrebbe $f(2, 1) + f(12, 1) = 27$, generando così un valore di upper bound $f(14, 1) = 27$ che risulta corretto.

In generale, si può definire $DRN_v(w)$ come l'insieme di triple (x, l, m), dove x, l, ed m sono coordinate normalizzate orizzontali compatibili con l'ampiezza w, e tali sia che $0 < x \leq l \leq m$, sia che entrambe le coppie (x, m) ed $(m – l, m – x)$ appartengono all'insieme $RN_v(w)$, cioè m è raggiungibile da x ed $m – x$ è raggiungibile da $m – l$, entrambi rispetto a w. Si suppone che $RN_v(w)$ contenga, per ciascun valore $l \in N_v(L_S, w)$, la coppia (l, l), estendendo così la definizione del Capitolo precedente, ma senza perdere la correttezza delle considerazioni ivi riportate. Dunque, per ciascuno di tali valori l, e per ciascun valore m maggiore di l ed appartenente ad $N_v(L_S, w)$, la condizione $(l, l, m) \in DRN_v(w)$ equivale alla condizione $(l, m) \in RN_v(w)$. Il nome DRN è stato scelto per tale insieme per ricordare l'acronimo "doppio normalizzato raggiungibile".

In modo simile, si possono definire gli insiemi $DRN_h(l)$, contenenti triple di coordinate normalizzate verticali compatibili con la lunghezza l.

L'Esempio 3 ha lo scopo di mostrare come l'utilizzo della condizione $C_{v,1}$ possa determinare errori, ed ha i seguenti dati di input. Una lastra con dimensioni $L_S = 39$ e $W_S = 3$ e sette tipi di pezzi, cioè t_1 con $l_1 = 3$, $w_1 = 2$, $\pi_1 = 2$, t_2 con $l_2 = 5$, $w_2 = 2$, $\pi_2 = 3$, t_3 con $l_3 = 7$, $w_3 = 1$, $\pi_3 = 2$, t_4 con $l_4 = 9$, $w_4 = 3$, $\pi_4 = 9$, t_5 con $l_5 = 17$, $w_5 = 3$, $\pi_5 = 15$, t_6 con $l_6 = 19$, $w_6 = 2$, $\pi_6 = 17$ ed infine t_7 con $l_7 = 21$, $w_7 = 1$, $\pi_7 = 7$, tutti con domanda 1. In questo caso $N_v(39, 1) = \{7, 21\}$ ed $N_v(7, 1) = N_v(21, 1) = \varnothing$, $N_v(39, 2) = \{3, 5, 7, 8, 10, 12, 15, 19, 21, 22, 24, 26, 27, 28, 29, 31, 33, 34, 36\}$, $N^*_v(3, 2) = N^*_v(5, 2) = N^*_v(7, 2) = \varnothing$, $\underline{N}^*_v(8, 2) = \underline{N}^*_v(10, 2) = \{3\}$, $\underline{N}^*_v(12, 2) = \{5\}$, $\underline{N}^*_v(15, 2) = \{3, 5, 7\}$, $N^*_v(19, 2) = N^*_v(21, 2) = \varnothing$, $N^*_v(38,2) = \{19\}$, mentre per tutti gli altri valori l di $N_v(39, 2)$ che siano maggiori di 21, l'insieme $\underline{N}^*_v(l, 2)$ contiene tutti i valori di $N^*_v(l – 19, 2) \cup \{19, l – 19\}$ se l è raggiungibile da 19 e tutti i valori di $N^*_v(l – 21, 2) \cup \{21, l – 21\}$ se l è raggiungibile da 21. Questo avviene perchè 19 è il massimo valore inferiore alla metà di L_S e quindi nessun valore ottenuto come somma fra 19, oppure 21, ed un qualsiasi altro addendo, può appartenere all'insieme $\underline{N}^*_v(l, 2)$, che contiene solo valori non superiori alla metà di l. Per esempio, 31 è raggiungibile sia da 19 che da 21, per cui $\underline{N}^*_v(31, 2) = N^*_v(12, 2) \cup N^*_v(10, 2) \cup \{10, 12, 19, 21\}$, cioè $\{5, 7, 10, 12, 19, 21\}$.

Si ha $f(7, 1) = f_0(7, 1) = 2$, $f(21, 1) = f_0(21, 1) = 7$ ed $f(28, 1) = f(7, 1) + f(21, 1) = 9$. Inoltre, per $w = 2$, si ha $f(3, 2) = f_0(3, 2) = 2$, $f(7, 2) = f(5, 2) = f_0(5, 2) = 3$, $f(10, 2) = f(8, 2) = f(3, 2) + f(5, 2) = 5$, $f(12, 2) = f(5, 2) + f(7, 2) = 6$, $f(15, 2) = f(3, 2) + f(12, 2) = 8$, $f(21, 2) = f(19, 2) = f_0(19, 2) = 17$ e, per ogni altro valore l appartenente ad $N_v(39, 2)$, e dunque maggiore di 21, se l è raggiungibile da 19 rispetto a $w = 2$, allora $f(l, 2)$ è pari alla somma $f(l – 19, 2) + f(19, 2)$, cioè $f(l – 19, 2) + 17$ e, in caso contrario, $f(l, 2) = f(l', 2)$ dove l' è il massimo valore di $N_v(39, 2)$ inferiore ad l e raggiungibile da 19 rispetto a 2. Queste uguaglianze valgono perchè $f_0(19, 2) = 17$ è ben più grande di $f_0(21, 2) = 7$.

I valori associati all'ampiezza $w = 3$ sono elencati in Tabella 5.1, che riporta le somme che danno ciascun valore l di $N_v(39, 3)$ ed anche, per ciascuno di tali l, la somma che determina la coppia di valori $f(l, 3)$ e $\lambda(l, 3)$ secondo la formulazione (5.3)-(5.5).

Per gli scopi di questo paragrafo, è necessario considerare solo pochi di tali valori. Si sottolinea che quando $f(l, 3) = f(l – 1, 3)$, si è utilizzata la posizione $\lambda(l, 3) = l$, come

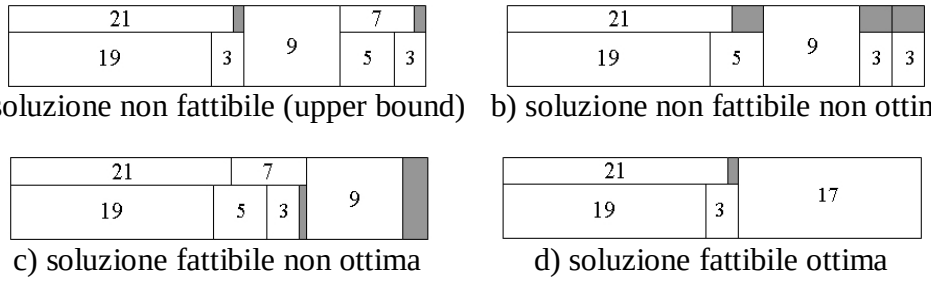
consigliato dopo l'Esempio 1. I valori $\lambda(l, 3)$ ottenuti in tal modo sono contrassegnati in Tabella 5.1 con un asterisco. Si è fatto ciò per rendere chiaro il fatto che l'errore che emerge in questo nuovo esempio è dovuto solo all'utilizzo della condizione $C_{v,1}$. Oltretutto, nessuno di questi valori influenza la discussione. Si noti dunque come tutti i valori fino ad $f(38, 3)$ sono calcolati correttamente, vale a dire che sono uguali a quelli teoricamente determinati dalla formulazione (5.3)-(5.5).

$N_v(39, 3) = \{\mathbf{3}, \mathbf{5}, \mathbf{7}, 8, \mathbf{9}, 10, 12, 14, 15, 16, \mathbf{17}, \mathbf{19}, 20, \mathbf{21}, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39\}$				
l	$\underline{N}_v^*(l, 3)$	Somme per l	$f(l, 3) / \lambda(l, 3)$	Somma per $f(l,3)$
3	∅	2	2 / 3	$f_0(3, 3)$
5	∅	5	3 / 5	$f_0(4, 3)$
7	∅	7	5 / 7	$f(7, 1) + f(7, 2)$
8	{3}	3+5	7 / 8	$f(8, 1) + f(8, 2)$
9	∅	9	9 / 9	$f_0(9, 3)$
10	{3}	3+7	9 / 10*	$f(9, 3)$
12	{3, 5}	5+7 , 3+9	11 / 3	$f(3, 3) + f(9, 3)$
14	{5}	5+9	12 / 5	$f(5, 3) + f(9, 3)$
15	{3, 5, 7}	3+5+7	13 / 3	$f(3, 3) + f(12, 3)$
16	{7}	7+9	14 / 7	$f(7, 3) + f(9, 3)$
17	{3, 5, 8}	3+5+9 , 17	16 / 8	$f(8, 3) + f(9, 3)$
19	{3, 7, 9}	3+7+9 , 19	19 / 19	$f(19, 1) + f(19, 2)$
20	{3}	3+17	19 / 20*	$f(19, 3)$
21	{5, 7, 9}	5+7+9 , 21	24 / 21	$f(21, 1) + f(21, 2)$
22	{3, 5}	5+17 , 3+19	26 / 22	$f(22, 1) + f(22, 2)$
24	{3, 5, 7, 8, 9, 10, 12}	3+5+7+9 , 7+17 , 5+19 , 3+21	27 / 24	$f(24, 1) + f(24, 2)$
25	{3, 5, 8}	3+5+17	28 / 3	$f(3, 3) + f(22, 2)$
26	{5, 7, 9}	9+17 , 7+19 , 5+21	28 / 26*	$f(25, 3)$
27	{3, 5, 7, 8, 10}	3+5+19 , 3+7+17	29 / 3	$f(3, 3) + f(24, 3)$
28	{7, 9}	9+19 , 7+21	31 / 28	$f(28, 1) + f(28, 2)$
29	{3, 5, 7, 8, 9, 10, 12}	3+9+17 , 5+7+17 , 3+7+19 , 3+5+21	31 / 29*	$f(28, 3)$
30	{9}	9+21	33 / 9	$f(9, 3) + f(21, 3)$
31	{3, 5, 7, 9, 10, 12, 14}	5+9+17 , 3+9+19 , 5+7+19 , 3+7+21	35 / 9	$f(9, 3) + f(22, 3)$
32	{3, 5, 7, 8, 10, 12, 15}	3+5+7+17	35 / 32*	$f(31, 3)$
33	{3, 5, 7, 9, 12, 14, 16}	7+9+17 , 5+9+19, 3+9+21 , 5+7+21	36 / 9	$f(9, 3) + f(24, 3)$
34	{3, 5, 7, 8, 9, 10, 12, 14, 15, 17}	3+5+9+17, 3+5+7+19	37 / 3	$f(3, 3) + f(31, 3)$
35	{5, 7, 9, 14, 16}	7+9+19 , 5+9+21	37 / 35*	$f(34, 3)$
36	{3, 5, 7, 8, 9, 10, 12, 14, 15, 17}	17+19 , 3+5+9+19 , 3+5+7+21	38 / 3	$f(3, 3) + f(33, 3)$
37	{7, 9, 16}	7+9+21	40 / 9	$f(9, 3) + f(28, 3)$
38	{3, 5, 7, 8, 9, 12, 14, 16, 17}	5+7+9+17 , 3+5+9+21 , 17+21	40 / 38*	$f(37, 3)$
39	{3, 17, 19}	3+17+19	40	$f(3, 3) + f(36, 3)$
			42	$f(17, 3) + f(22, 3)$
			38	$f(19, 3) + f(20, 3)$
			40	$f(38, 3)$

Tabella 5.1. Esempio 3 di non correttezza della condizione classica $C_{v,1}$.

Il valore $f(39, 3)$ dovrebbe essere posto al massimo fra i valori $f(3, 3) + f(36, 3)$, $f(17, 3) + f(22, 3)$, $f(19, 3) + f(20, 3)$ ed $f(38, 3)$. La prima somma dà come risultato il valore di profitto 40, la seconda 42 e la terza 38. Il quarto ed ultimo valore è $f(38, 3) = 40$. Il valore selezionato dovrebbe dunque essere 42, corrispondente alla soluzione, non fattibile, mostrata in Figura 5.5a, dove è indicata la lunghezza di ciascun pezzo. Tuttavia, $\lambda(17, 3) = 8 < 17$, per cui il valore 17 non può assumere il ruolo di x_1 nella seconda somma, a causa della condizione $C_{v,1}$ che risulta non soddisfatta. Quindi il valore scelto per $f(39, 3)$ è 40, corrispondente alle due soluzioni, una fattibile e l'altra no, mostrate in Figura 5.5b ed in Figura 5.5c. L'errore che

ne consegue è dovuto al fatto che la soluzione fattibile ottima, mostrata in Figura 5.5d, ha valore di profitto 41.



a) soluzione non fattibile (upper bound) b) soluzione non fattibile non ottima

c) soluzione fattibile non ottima d) soluzione fattibile ottima

Figura 5.5. Soluzioni per l'Esempio 3

In un modo del tutto simile all'Esempio 1, l'errore scompare se la seguente condizione diventa vera, per esempio per l'aggiunta di nuovi tipi di pezzo. Se la tripla (8, 17, 39) appartiene a $DRN_v(3)$, allora 31 è raggiungibile da 9 rispetto a 3, e dunque la nuova somma $f(9, 3) + f(22, 3) = 35$ viene realizzata, determinando così $f(31, 3) = 35$ e $\lambda(31, 3) = 9$. Inoltre, anche la nuova somma $f(8, 3) + f(31, 3) = 35 + 7 = 42$ viene realizzata, facendo così ottenere il corretto upper bound $f(39, 3) = 42$.

5.4.2.2 *Definizioni*

Prima di spiegare come utilizzare gli insiemi $DRN_v(w)$ e $DRN_h(l)$ per aggiustare la procedura $\mathbf{P}^{++}$, si mostrano condizioni più forti per una tripla (x, m, l), le quali implicano $(x, l, m) \in DRN_v(w)$. A tal fine, si introduce una nuova definizione. Per tre dati valori x, l ed m tali che $x \leq l \leq m$, si dice che m è raggiungibile da x attraverso l, rispetto a w, quando esistono tre bounded cutting patterns compatibili con w, B^x, B^l e B^m, generanti rispettivamente x, l ed m e tali che ciascuno di essi è contenuto nel successivo, cioè $B^x \subseteq B^l \subseteq B^m$. L'insieme di tali triple viene denotato con $TRN_v(w)$, dove il nome *TRN* è stato scelto per ricordare l'acronimo "tripla raggiungibilità fra normalizzate".

Si sottolinea che, se $x = l$, allora si può utilizzare $B^x = B^l$ per cui $(x, l, m) \in TRN_v(w)$ equivale a $(l, m) \in RN_v(w)$, mentre se $l = m$, allora si può utilizzare $B^l = B^m$ per cui $(x, l, m) \in TRN_v(w)$ equivale a $(x, l) \in RN_v(w)$. Quindi, per ciascun l appartenente ad $N_v(L_S, w)$, vale la condizione $(l, l, l) \in TRN_v(w)$.

La condizione $(x, l, m) \in TRN_v(w)$ implica che $(x, l, m) \in DRN_v(w)$. Infatti, se $x = l$ o se $l = m$, questo appare ovvio, mentre, quando $x < l < m$, se i corrispondenti tre cutting patterns sono B^x, B^l e B^m, valgono le seguenti considerazioni. Innanzitutto, $B^x \subset B^m$ e dunque $(x, m) \in RN_v(w)$. Inoltre m è raggiungibile anche da l, rispetto a w, poiché $B^l \subset B^m$. Quindi, sia $m - l$ che $m - x$ sono coordinate normalizzate orizzontali compatibili con w. Inoltre $m - x$ è generato dal cutting pattern ottenuto come differenza fra B^m e B^x e denotato con B^{m-x}, ed $m - l$ è generato dal cutting pattern ottenuto come differenza fra B^m e B^l e denotato con B^{m-l}. Quindi $m - x$ è raggiungibile da $m - l$ rispetto a w, poiché $B^x \subset B^l$ implica $B^{m-l} \subset B^{m-x}$.

Si sottolinea anche che $(x, l, m) \in DRN_v(w)$ non implica che $(x, l, m) \in TRN_v(w)$, cosa mostrata dall'Esempio 2, per i valori $x = 2$, $l = 10$ ed $m = 14$, dopo l'aggiunta dei tipi di pezzo t_4 e t_5. Inoltre, se (x, l, m) appartiene a $TRN_v(w)$, allora anche $(x, x + (m - l), m)$ ed $(l - x, l, m)$ appartengono a $TRN_v(w)$. Che ciò valga per la prima tripla, è ovvio se $l = m$, mentre, se $l < m$, allora $B^x \subseteq B^l \subset B^m$ e di conseguenza $B^x \subset B^x \oplus B^{m-l} \subseteq B^m$. Per la seconda tripla, è ovvio nel caso $x = l$, mentre, se $x < l$, allora $B^x \subset B^l \subseteq B^m$ per cui $B^l - B^x \subset B^l \subseteq B^m$.

Una ulteriore condizione può essere introdotta, che implichi $(x, l, m) \in TRN_v(w)$. Sia $\beta_v(l, w)$ l'insieme di bounded cutting patterns, compatibili con w e generanti l. Se per una data coppia (l, w), solo un cutting pattern B^l è contenuto in $\beta_v(l, w)$, allora per ciascun valore di x osservabile da l rispetto a w, e per ciascun valore m raggiungibile da l rispetto a w, si ha che m è raggiungibile da x attraverso l, rispetto a w, cioè che esistono altri due bounded cutting patterns B^x ed B^m, appartenenti rispettivamente a $\beta_v(x, w)$ ed a $\beta_v(m, w)$, tali che $B^x \subseteq B^l \subseteq B^m$.

Infatti, basta considerare un qualunque cutting pattern B^m di $\beta_v(m, w)$ tale che $B^l \subseteq B^m$. Esso esiste certamente dato che m è raggiungibile da l rispetto a w. Inoltre, si può considerare un qualsiasi cutting pattern B^x di $\beta_v(x, w)$, tale che $B^x \subseteq B^l$. Anch'esso esiste certamente poiché l è raggiungibile da x rispetto a w. Quindi $B^x \subseteq B^l \subseteq B^m$ e dunque la tripla (x, l, m) appartiene all'insieme $TRN_v(w)$.

Si denota con $BN_v(w)$ l'insieme di valori l per cui valga la suddetta proprietà. Per mostrare che la condizione $(x, l, m) \in TRN_v(w)$ non implica che $l \in BN_v(w)$, si può aggiungere un sesto tipo di pezzi t_6 all'Esempio 2, con dimensioni $l_6 = 8$ e $w_6 = 1$, domanda $d_6 > 1$ ed un qualsiasi profitto π_6. In tal modo, la lunghezza $m = 14$ diventa raggiungibile dalla lunghezza $x = 2$ attraverso la lunghezza $l = 10$. Infatti 14 diventa ottenibile anche attraverso la somma 2+8+4. Tuttavia, la lunghezza 10 è ancora ottenibile con più di una somma, per esempio 2+8 e 5+5.

Si sottolinea che l'insieme $BN_v(w)$ può essere costruito come l'insieme di tutti i valori che, nella procedura **PV**, sono aggiunti solo una volta all'insieme $N_v'(L_S, W_S)$, attraverso lo step II.a (Box 5.3), prima che $\underline{w}$ superi w.

Prima di discutere come modificare la procedura $\mathbf{P}^{++}$, vengono date altre definizioni che permettono di generalizzare e formalizzare le considerazioni derivanti dai tre esempi sopra esposti. Ci si concentrerà sul calcolo del termine parziale f_v relativo ai tagli verticali. Per il termine f_h relativo ai tagli orizzontali, analoghe definizioni possono essere date ed analoghe modifiche possono essere apportate alla procedura.

Attraverso l'Esempio 2, si è mostrato che la condizione $C_{v,2}$ non funziona correttamente in presenza di domande di pezzi limitate. Nel caso di domande illimitate, il suo scopo, insieme alla condizione $C_{v,1}$, è evitare che somme equivalenti ma simmetriche siano realizzate.

Si consideri, in particolare, per una data ampiezza w, una lunghezza $l \in N_v(L_S, w)$. Per un dato cutting pattern B di $\beta_v(l, w)$, si definisce come partizione di B un qualunque insieme $\{B_1, B_2, \ldots, B_k\}$ di cutting patterns, che generano valori positivi e tali che B sia ottenibile con la somma $\overset{k}{\underset{i=1}{\oplus}} B_i$. Ciò significa che tutti i cutting patterns appartenenti ad una partizione sono contenuti in B e che, se si denota con $x(B_j)$ il valore generato dal cutting pattern B_j , allora $l = \sum_{i=1}^{k} x(B_i)$. Si definisce $P(B)$ come l'insieme di tutte le possibili partizioni associate ad un cutting pattern B e si dice che due partizioni $p^1 = \{B^1_1, B^1_2, \ldots, B^1_k\}$ e $p^2 = \{B^2_1, B^2_2, \ldots, B^1_q\}$ di $P(B)$ sono equivalenti se i due insiemi $\{x(B^1_1), x(B^1_2), \ldots, x(B^1_k)\}$ ed $\{x(B^2_1), x(B^2_2), \ldots, x(B^1_q)\}$ sono uguali. Ovviamente, a tal fine, i due valori k e q devono essere uguali. Questa definizione induce una relazione di equivalenza sull'insieme $P(B)$.

Si stabilisce inoltre un ordinamento fra le partizioni di $P(B)$. Data una partizione, si considerano i suoi cutting patterns in una sequenza $B_1, B_2, \ldots, B_k$ tale che, per ogni coppia i, j, con $i < j$, il valore $x(B_i)$ generato da B_i sia maggiore o uguale al valore $x(B_j)$ generato da B_j . Inoltre, per una coppia di partizioni non equivalenti $p^1 = \{B^1_1, B^1_2, \ldots, B^1_k\}$ e $p^2 = \{B^2_1, B^2_2, \ldots, B^1_q\}$, rispettivamente composte da k e q cutting patterns, si definisce la relazione d'ordine in modo ricorsivo dicendo che $p^1 < p^2$ se $x(B^1_1) < x(B^1_2)$ o se valgono sia $x(B^1_1) = x(B^2_1)$ sia $rp^1 < rp^2$, dove $rp^1 = \{B^1_2, \ldots, B^1_k\}$ ed $rp^2 = \{B^2_2, \ldots, B^1_q\}$ sono ottenute rimuovendo

rispettivamente B^1_1 da p^1 e B^2_2 da p^2 e dunque essi sono costituiti rispettivamente da $k-1$ e da $q-1$ cutting patterns. Da notare che né rp^1 né rp^2 possono essere vuote poiché se $k = 1$ e $q > 1$ allora $x(B^1_1) = l > x(B^2_1)$ ed, analogamente nel caso $k > 1$ e $q = 1$. Il caso $k = 1$ e $q = 1$ non è possibile, altrimenti p^1 e p^2 sarebbero equivalenti, in contraddizione con l'ipotesi. Inoltre, se $x(B^1_1) = x(B^2_1)$, allora anche rp^1 ed rp^2 non sono equivalenti. Da notare che sia la relazione d'ordine che quella di equivalenza possono essere estese a coppie di partizioni associate a differenti cutting patterns. Si denota infine con $\underline{P}_v(l, w)$ l'insieme ottenuto come unione degli insiemi $P(B)$ corrispondenti a ciascun bounded cutting pattern B di $\beta_v(x, w)$.

Ancora, la partizione $p^1 = \{B^1_1, B^1_2, \ldots, B^1_k\}$ si dice dominata dalla partizione $p^2 = \{B^2_1, B^2_2, \ldots, B^1_q\}$ quando $q < k$ e l'insieme $\{x(B^2_1), x(B^2_2), \ldots, x(B^2_q)\}$ corrispondente p^2 può essere ottenuto riarrangiando gli elementi dell'insieme $\{x(B^1_1), x(B^1_2), \ldots, x(B^1_k)\}$ corrispondente a p^1 in q sottoinsiemi separati e non vuoti, $\{x(B^1_{\alpha(1,1)}), x(B^1_{\alpha(1,2)}), \ldots, x(B^1_{\alpha(1,s(1))})\}$, …, $\{x(B^1_{\alpha(q,1)}), x(B^1_{\alpha(q,2)}), \ldots, x(B^1_{\alpha(q,s(q))})\}$ tali che, sommando, per ciascun $j = 1, \ldots, q$, gli s(j) elementi del j-mo sottoinsieme, il risultato della somma $\sum_{i=1}^{s(j)} x\left(B_{\alpha(j,i)}\right)$ è uguale a $x(B^2_j)$. In pratica, si può costruire una partizione pp^1 di p^1, nel senso classico del termine, tale che il valore generato da ciascun cutting pattern di p^2 è uguale alla somma dei valori generati dai cutting patterns di un differente sotto-insieme, elemento di pp^1. Da notare che, come conseguenza di tale definizione, una qualunque partizione p è minore di una qualunque partizione dominante p, secondo la relazione d'ordine sopra stabilita.

Per esempio, nell'Esempio 2, con i soli primi tre tipi di pezzi, la lunghezza 10 è ottenibile con le somme 2+4+4 e 5+5, ed è generabile quindi dai due bounded cutting patterns $B_a = \{1, 2, 0\}$ e $B_b = \{0, 0, 2\}$, che generano 10 rispettivamente attraverso la combinazione lineare $1 \cdot l_1 + 2 \cdot l_2$ ed attraverso $2 \cdot l_3$, e quindi $\beta_v(10, 1) = \{B_a, B_b\}$. Le quattro partizioni di B_a sono $p_a^1 = \{B_a^1{}_1, B_a^1{}_2, B_a^1{}_3\}$, con $B_a^1{}_1 = \{1, 0, 0\}$ e $B_a^1{}_2 = B_a^1{}_3 = \{0, 1, 0\}$, $p_a^2 = \{B_a^2{}_1, B_a^2{}_2\}$, con $B_a^2{}_1 = \{1, 0, 0\}$ e $B_a^2{}_2 = \{0, 2, 0\}$, $p_a^3 = \{B_a^3{}_1, B_a^3{}_2\}$, con $B_a^3{}_1 = \{1, 1, 0\}$ e $B_a^3{}_2 = \{0, 1, 0\}$, ed infine $p_a^4 = \{B_a^4{}_1\}$, con $B_a^4{}_1 = \{1, 2, 0\}$. Gli insiemi di valori generati associati a queste quattro partizioni sono $\{2, 4, 4\}$ per p_a^1, $\{2, 8\}$ per p_a^2, $\{4, 6\}$ per p_a^3 e $\{10\}$ per p_a^4. Quindi non vi sono due differenti partizioni di $P(B_a)$ equivalenti. Inoltre, $p_a^1 < p_a^2 < p_a^3 < p_a^4$. Infine, p_a^1 è dominato da p_a^2, dato che $\{2, 4+4\} = \{2, 8\}$, da p_a^3 dato che $\{4, 2+4\} = \{2, 6\}$ e da p_a^4 poiché $\{2+4+4\} = \{10\}$, mentre p_a^2 è dominato da p_a^4, dato che $\{2+8\} = \{10\}$, ed anche p_a^3 è dominato da p_a^4, poiché$\{4+6\} = \{10\}$.

Le due possibili partizioni di B_b sono $p_b^1 = \{B_b^1{}_1, B_b^1{}_2\}$, con $B_b^1{}_1 = B_b^1{}_2 = \{0, 0, 1\}$ e $p_b^2 = \{B_b^2{}_1\}$, con $B_b^2{}_1 = \{0, 0, 2\}$ e gli insiemi associati di valori generati sono $\{5, 5\}$ per p_b^1 e $\{10\}$ per p_b^2. Inoltre, p_b^1 e p_b^2 non sono equivalenti, $p_b^1 < p_b^2$ e p_b^1 è dominato da p_b^2, dato che $\{5+5\} = \{10\}$.

Esempi di relazione di ordine e di equivalenza coinvolgenti partizioni associati a differenti cutting patterns, sono dati da $p_a^1 < p_b^2$ e dall'equivalenza di p_a^4 e p_b^2.

5.4.2.3 *Analisi delle condizioni classiche*

Sulla base delle definizioni introdotte, si discute ora il modo di funzionare delle condizioni $C_{v,1}$ e $C_{v,2}$ prima di sottolineare, nel successivo sotto-paragrafo, quando esse possono fallire nel loro intento nel caso constrained, formalizzando e generalizzando le considerazioni emerse dai su esposti Esempi 1, 2 e 3.

Si supponga che, per una data coppia (l, w) di coordinate normalizzate tali che l sia compatibile con w e che $N_v^*(l, w) \neq \varnothing$, il valore $f_v(l, w)$, dato dalla relazione (5.4), sia ottenibile attraverso la somma

$$\sum_{i=1}^{k} f\left(x(B_i^l), w\right) \tag{5.6}$$

dove $p^l = \{B^l_1, B^l_2, \ldots, B^l_k\}$ è una partizione del cutting pattern B^l, appartenente a $\beta_v(x, w)$, e dunque contenuta in $\underline{P}_v(l, w)$. Si supponga inoltre che p^l sia tale che la seguente proprietà sia valida

$$\lambda(x(B^l_i), w) = x(B^l_i) \text{ per ogni } i = 1, \ldots, n \tag{5.7}$$

dove, senza perdita di generalità, gli elementi di p^l sono arrangiati in ordine non crescente, cioè in modo tale che se $i < j$, allora $x(B^l_i) \geq x(B^l_j)$. La stessa ipotesi per $f_v(l, w)$, con $k = 3$, $x(B^l_3)$ denotato da u_1, $x(B^l_2)$ denotato da u_2 ed $x(B^l_3)$ denotato da u_1, è stata utilizzata nel paragrafo 5.2 per dare brevemente un esempio del modo in cui le condizioni $C_{v,1}$ e $C_{v,2}$ funzionano. Si supporrà nel seguito, senza perdita di generalità, e fino a che esplicitamente controindicato, che solo una esistente partizione goda di tali proprietà.

In generale, si può affermare che lo scopo di $C_{v,1}$ è permettere di realizzare la somma (5.6) e non permettere che lo stesso avvenga per le analoghe somme associate alle partizioni che dominano p^l. Per provare questo, è necessario stabilire un risultato preliminare, cioè che, per ciascun sottoinsieme $\{B_{\alpha(1)}, B_{\alpha(2)}, \ldots, B_{\alpha(s)}\}$ di cutting patterns di p^l, con $s > 1$, il valore $f\left(\sum_{i=1}^{s} x(B^l_{\alpha(i)}), w\right)$, vale a dire $f\left(x\left(\bigoplus_{i=1}^{s} B^l_{\alpha(i)}\right), w\right)$, è uguale a $\sum_{i=1}^{s} f\left(x(B^l_{\alpha(i)}), w\right)$. Infatti il segno di "maggiore o uguale" è ovviamente valido ed il caso "maggiore" è impossibile, altrimenti un valore maggiore per $f_v(l, w)$ potrebbe essere ottenuto attraverso la somma $f\left(\sum_{i=1}^{s} x(B^l_{\alpha(i)}), w\right) + \sum_{j \notin A} f\left(x(B^l_j), w\right)$ associata alla partizione $\left\{\bigoplus_{i=1}^{s} B^l_{\alpha(i)}\right\} \cup \bigcup_{j \notin A} \{B^l_j\}$, dove A è l'insieme di indici $\{\alpha(1), \alpha(2), \ldots, \alpha(s)\}$. Di conseguenza $f\left(\sum_{i=1}^{s} x(B^l_{\alpha(i)}), w\right) = \sum_{i=1}^{s} f\left(x(B^l_{\alpha(i)}), w\right)$, che è uguale a $\sum_{i=1}^{s-1} f\left(x(B^l_{\alpha(i)}), w\right) + f\left(x(B^l_{\alpha(s)}), w\right)$, vale a dire $f\left(\sum_{i=1}^{s-1} x(B^l_{\alpha(i)}), w\right) + f\left(x(B^l_{\alpha(s)}), w\right)$. Da ciò si ricava che, se si suppone che $\alpha(1) < \alpha(2) < \ldots < \alpha(s)$, allora vale la seguente

$$\lambda\left(\sum_{i=1}^{s} x(B^l_{\alpha(i)}), w\right) \leq x(B^l_{\alpha(s)}) \tag{5.8}$$

Quindi $\lambda\left(\sum_{i=1}^{s} x(B^l_{\alpha(i)}), w\right) < \sum_{i=1}^{s} x(B^l_{\alpha(i)})$ e dunque la condizione $C_{v,1}$ non permette che la somma $\sum_{i=1}^{s} x(B^l_{\alpha(i)})$ sia utilizzata nel ruolo di x_1 in alcuna somma del tipo $f(x_2, w) + f(x_1, w)$. Solo i valori $x(B^l_i)$ possono essere usati a tal fine e quindi solo la somma (5.6) può essere realizzata, evitando cioè quelle associate a partizioni che dominano p^l.

L'effetto della condizione $C_{v,2}$ è di ottenere il valore $\sum_{i=1}^{k} f\left(x(B^l_i), w\right)$ solo attraverso la somma fra i due addendi $\sum_{i=1}^{k-1} f\left(x(B^l_i), w\right)$ e $f\left(x(B^l_k), w\right)$, fra le somme $\sum_{j \neq i} f\left(x(B^l_j), w\right) + f\left(x(B^l_i), w\right)$ identificate da ciascun possibile valore di i. In particolare, si suppone, senza perdita di generalità, che non esistono due diversi valori i e j tali che $x(B^l_i) = x(B^l_j)$, e quindi la

condizione $C_{v,2}$, per ciascun $i < k$, evita che la somma $\sum_{j\neq i} f\left(x(B_j^l), w\right) + f\left(x(B_i^l), w\right)$ sia realizzata. Infatti, sfruttando la proprietà (5.8) con l'insieme di indici A contenente tutti i valori da 1 ad n, tranne i, si può affermare che $\lambda\left(\sum_{j\neq i} x(B_j^l), w\right) \leq x(B_k^l)$, poiché k è il massimo indice di A. Dunque, se $i < k$, allora vale la relazione $\lambda\left(\sum_{j\neq i} x(B_j^l), w\right) < x(B_i^l)$ e quindi $x(B^l_i)$ non può assumere il ruolo di x_1 quando $x_2 = \sum_{j\neq i} x(B_j^l)$. Solo $x(B^l_k)$ può farlo, con $x_2 = \sum_{j=1}^{k-1} x(B_j^l)$.

Se si denota la somma $\sum_{i=1}^{k-1} x(B_i^l)$ con $x^l_{1,k-1}$ e ciascun valore $x(B_j^l)$ con x^l_j, allora la conclusione è che le due condizioni $C_{v,1}$ e $C_{v,2}$ hanno lo scopo di permettere che l'unica somma realizzata sia $\sum_{i=1}^{k-1} f\left(x^l_{1,k-1}, w\right) + f\left(x^l_k, w\right)$, fra quelle sopra indicate. Da notare pure che in generale più somme, con lo stesso massimo risultato per $f_v(l, w)$, potrebbero essere permesse, corrispondenti a differenti cutting patterns, ma non a differenti partizioni, tali che l'una domini l'altra, poiché solo per una di esse è possibile che valga l'uguaglianza $\lambda(x(B^l_i), w) = x(B^l_i)$ per ogni i.

Nel seguito si prova la validità di due teoremi, nel caso di domanda illimitata. Il primo è in relazione alla condizione $C_{v,1}$ ed il secondo anche alla condizione $C_{v,2}$. Si denota inoltre con $\underline{P}^*_v(l, w)$ l'insieme delle partizioni di $\underline{P}_v(l, w)$ tali che la corrispondente somma (5.6) dia $f_v(l, w)$ come risultato. Esso è non vuoto poiché sopra si è ipotizzato che $N_v^*(l , w)$ non sia vuoto. Nella discussione fatta prima, si è assunto, senza provarlo, che esista almeno una partizione p^l di $\underline{P}^*_v(l, w)$ per la quale la proprietà (5.7) fosse valida. Inoltre non si è provato che l'unica somma realizzabile $\sum_{i=1}^{k-1} f\left(x^l_{1,k-1}, w\right) + f\left(x^l_k, w\right)$ in accordo alle condizioni $C_{v,1}$ e $C_{v,2}$ sia effettivamente realizzata. Se entrambe queste cose vengono provate, ne consegue che le condizioni $C_{v,1}$ e $C_{v,2}$ lavorano correttamente nel caso di domanda illimitata, mentre esse possono essere non corrette nel caso di domanda limitata, come mostrato attraverso i tre esempi di prima.

Si comincia con il provare il seguente Teorema 5.1.

Teorema 5.1.
Nel caso di domanda illimitata, esiste almeno una partizione $p^l = \{B^l_1, B^l_2, \ldots, B^l_k\}$, appartenente a $\underline{P}^*_v(l, w)$ e tale che $\lambda(x(B^l_i), w) = x(B^l_i)$ per ogni $i = 1, \ldots, k$.

Prova. Infatti, se per una data p^l di $\underline{P}^*_v(l, w)$ questa proprietà non vale per qualche suo cutting pattern B^l_j, si può costruire una nuova partizione p^l_2, a partire da p^l e rimpiazzando ciascuno di tali B^l_j con i cutting patterns di una partizione di $\underline{P}^*_v(x(B^l_j), w)$ che soddisfi il teorema, e che esiste per ipotesi induttiva. Ovviamente p^l_2 soddisfa la proprietà indicata nella tesi. Del resto il cutting pattern associato a p^l_2 non necessita di essere bounded e quindi esso è certamente compatibile con w. Ciò prova la validità del Teorema 5.1.

Dunque la somma desiderata $\sum_{i=1}^{k-1} f\left(x^l_{1,k-1}, w\right) + f\left(x^l_k, w\right)$ è permessa dalla condizione $C_{v,1}$ nel caso di domanda illimitata. Si noti pure che nel caso di domanda limitata, non si può affermare che il cutting pattern associato a p^l_2 sia bounded.

Comunque, lo scopo ultimo è mostrare che esiste almeno una partizione p^l di $\underline{P}^*_v(l, w)$ tale che le relazioni $\lambda(x^l_{1,k-1}, w) \geq x^l_k = \lambda(x^l_k, w)$ siano valide, cioè tale che la corrispondente somma sia permessa sia da $C_{v,1}$ che da $C_{v,2}$.

Per ottenere tale risultato, si prova preliminarmente il seguente Teorema 5.2.

Teorema 5.2.
Nel caso di domanda illimitata, la relazione $\lambda(l, w) = x^l_k$ vale per almeno una partizione p^l appartenente a $\underline{P}^*_v(l, w)$.

Prova. Si cominci con il notare che, attraverso la (5.8), si è stabilita la validità della tesi con il segno "minore o uguale" anziché soltanto "uguale". Si suppone induttivamente che il segno di "uguale" valga per ogni x inferiore ad l, denotando con $p^*(x, w)$ la corrispondente partizione. Se, per una data p^l di $\underline{P}^*_v(l, w)$, vale $\lambda(l, w) < x^l_k$, allora $f_v(l, w)$ può essere ottenuta con la somma $f(\lambda(l, w), w) + f(l - \lambda(l, w), w)$ e quindi si può considerare la partizione p^l_2 che ha, come corrispondente insieme di lunghezze, l'insieme ottenuto a partire da quello associato alla partizione $p^*(l - \lambda(l, w), w)$ aggiungendovi il valore $\lambda(l, w)$, che è inferiore ad x^l_k. E' certo che esista qualche cutting pattern B di $\beta_v(l, w)$ tale che $p^l_2 \in P(B)$, grazie alla domanda illimitata di pezzi, mentre nel caso di domanda limitata ciò non è vero. Inoltre, è sicuro che $p^l_2 \in \underline{P}^*_v(l, w)$, ovvero che la somma associata a p^l_2 dia come risultato $f_v(l, w)$.

Si noti che la partizione p^l_2 è inferiore alla partizione p^l, in accordo con la relazione d'ordine stabilita prima. Per p^l la condizione $\lambda(l, w) = x^l_k$ è stata supposta essere falsa. Per p^l_2 la corrispondente condizione è $\lambda(l, w) = \lambda(l, w)$, poiché $\lambda(l, w)$ è il minimo elemento dell'insieme di lunghezze associata a p^l_2, e quindi è certamente valida. Si noti infatti che la minima lunghezza associata a $p^*(l - \lambda(l, w), w)$ non può essere inferiore a $\lambda(l, w)$, altrimenti la somma associata a p^l_2 non potrebbe dare $f_v(l, w)$ come risultato, secondo la definizione di $\lambda(l, w)$ come massimo valore di x per il quale vale l'uguaglianza $f_v(l, w) = f(l, w) + f(l - x, w)$. Ciò prova il Teorema 5.2.

Per raggiungere lo scopo ultimo su dichiarato, si può partire da una partizione p^l di $\underline{P}^*_v(l, w)$ tale che la (5.7) sia valida. Essa esiste in accordo al Teorema 5.1. Se, per tale p^l, anche la seconda relazione di interesse, $\lambda(x^l_{1,k-1}, w) \geq x^l_k$, è valida, allora essa è la partizione desiderata cui corrisponde la somma realizzata in accordo sia a $C_{v,1}$ che a $C_{v,2}$. Se, al contrario, $\lambda(x^l_{1,k-1}, w) < x^l_k$, allora si può applicare il processo utilizzato nella prova del Teorema 5.2, con $x^l_{1,k-1}$ nel ruolo di l ed a partire dalla partizione p^l_a, ottenuta da p^l_2 eliminando il cutting pattern che genera il valore x^l_k. Si ottiene così una nuova partizione p^l_b per il valore $x^l_{1,k-1}$, per la quale k potrebbe essere cambiata ed il nuovo valore x^l_{k-1} è tale che $\lambda(x^l_{1,k-1}, w) = x^l_{k-1}$. Aggiungendo a p^l_b il cutting pattern estratto da p^l_2 per ottenere p^l_a, si ottiene una partizione finale p^l_3, minore di p^l_2 ed appartenente a $\underline{P}^*_v(l, w)$. Se il nuovo valore di x^l_{k-1} associato a p^l_3 non è minore di x^l_k, allora la partizione desiderata è stata trovata, poiché valgono sia l'uguaglianza $\lambda(x^l_k, w) = x^l_k$ sia la disuguaglianza $\lambda(x^l_{1,k-1}, w) \geq x^l_k$.

Se, al contrario, il nuovo valore x^l_{k-1} è inferiore ad x^l_k, si devono riarrangiare gli elementi di p^l_3 per mantenere il loro ordinamento non crescente. Se p^l_3 non soddisfa le proprietà desiderate, si può riapplicare ad essa il processo relativo alla prova del Teorema 5.1 o quello relativo alla prova del Teorema 5.2, a seconda della proprietà, fra le due associate rispettivamente a $C_{v,1}$ ed a $C_{v,2}$ che non vale. Questo processo combinato non può andare avanti senza avere termine, poiché esiste un limitato numero di partizioni e ad ogni step si passa ad una partizione inferiore alla precedente. Quindi, durante il processo combinato, viene sicuramente trovata una partizione tale che valgano sia $\lambda(x^l_{1,k-1}, w) \geq x^l_k$ che $x^l_k = \lambda(x^l_k, w)$. Si è quindi raggiunto lo scopo di mostrare che le condizioni $C_{v,1}$ e $C_{v,2}$ funzionano bene nel caso

di domanda illimitata. Il modo sfruttato per dimostrarlo risulta utile per analizzare il modo in cui aggiustare la procedura $\mathbf{P}^{++}$ affinché funzioni correttamente nel caso di domanda limitata.

5.4.2.4 *Procedura definitiva*

E' giunto il momento in cui si può affermare che, per aggiustare la procedura $\mathbf{P}^{++}$, è necessario assicurarsi che per almeno una partizione p^l di $\underline{P}_v(l, w)$ e tale che $f_v(l, w) = \sum_{B \in p} f\left(x(B), w\right)$, la corrispondente somma sia realizzata, eliminando il vincolo di considerare solo partizioni per le quali $\lambda(x(B^l_i), w) = x(B^l_i)$ per ogni i. Infatti, l'Esempio 2 ha mostrato che, per poter calcolare correttamente il valore $f_v(l, w)$, è necessario evitare questa limitazione. Si decide di considerare solo la minima di tali partizioni, denotata nel seguito con $p^l_{min} = \{B^l_1, B^l_2, \ldots, B^l_k\}$ ed associata al cutting pattern $B^l = \bigoplus_{i=1}^{n} B^l_i$. Il corrispondente insieme di lunghezze è denotato con $\{x^l_1, \ldots, x^l_k\}$ e la somma $\sum_{i=1}^{k-1} x^l_i$ con $x^l_{1,k-1}$. Si sottolinea che altre partizioni potrebbero essere equivalenti a p^l_{min}, ma ciò non altera la validità delle seguenti considerazioni, basate solo sull'insieme di lunghezze associato.

Seguendo la stessa idea che conduce all'utilizzo delle condizioni $C_{v,1}$ e $C_{v,2}$, e che risulta valida nel caso di domanda illimitata, si spiega come accertarsi che la somma $f\left(x^l_{1,k-1}, w\right) + f\left(x^l_k, w\right)$, denotata nel seguito come "somma minima", sia realizzata. Si suppone induttivamente che il valore $f\left(x^l_{1,k-1}, w\right)$ sia correttamente calcolato pari a $\sum_{i=1}^{k-1} f\left(x^l_i, w\right)$. Un modo banale per assicurarsi che la somma minima sia realizzata potrebbe essere di eliminare l'utilizzo delle condizioni $C_{v,1}$ e $C_{v,2}$, vale a dire permettere tutte le somme, ma ciò non è preferibile. Quindi è necessario introdurre qualche regola che permetta di comprendere se, per una somma non realizzabile secondo le condizioni $C_{v,1}$ e $C_{v,2}$, si possa accettare la proibizione, dato che si è certi che non si tratti della somma minima. Quando tale certezza manca, è necessario rifiutare la proibizione e realizzare comunque la somma.

Si consideri allora che una somma può essere non realizzabile, secondo le condizioni $C_{v,1}$ e $C_{v,2}$, per due possibili motivazioni. La prima, associata a $C_{v,2}$, è che x^l_k sia maggiore di $\lambda(x^l_{1,k-1}, w)$, che potrebbe non essere pari a x^l_{k-1} ma inferiore ad esso ed addirittura, appunto, inferiore ad x^l_k. La seconda, associata a $C_{v,1}$, è che $\lambda(x^l_k, w) < x^l_k$.

Si sottolinea che, per il momento, non si tiene conto dell'errore emerso attraverso l'Esempio 1 per il caso $f(l, w) = f(l - 1, w)$. Tuttavia, al termine della spiegazione, si mostrerà un semplice modo che permette di estendere le modifiche che saranno state proposte, così da superare anche tale errore.

Si suppone per ora che solo una fra le due condizioni sia non soddisfatta, prima di discutere il caso in cui entrambe siano non soddisfatte. Si comincia con il caso in cui $C_{v,2}$ sia non soddisfatta.

Se $x^l_k > \lambda(x^l_{1,k-1}, w)$, si considera la condizione $(\lambda(x^l_{1,k-1}, w), x^l_{1,k-1}, l) \in DRN_v(w)$, cioè l raggiungibile da $\lambda(x^l_{1,k-1}, w)$ ed $l - \lambda(x^l_{1,k-1}, w)$ raggiungibile da $x^l_{1,k-1} - \lambda(x^l_{1,k-1}, w)$, entrambi rispetto a w. Se essa è soddisfatta, si può asserire che la somma non permessa non è la somma minima. Infatti, dalla prima raggiungibilità, se ne ricava che esistono due cutting patterns B^λ e $B^{2,l}$, appartenenti rispettivamente a $\beta_v(\lambda(x^l_{1,k-1}, w), w)$ ed a $\beta_v(l, w)$, e tali che $B^\lambda \subset B^{2,l}$. Si sottolinea che $B^{2,l}$ potrebbe essere diverso da B^l.

La somma associata alla partizione $p^{2,l} = \{B^\lambda, B^{2,l} - B^\lambda\}$ è

$$f\left(\lambda\left(x^{l}_{i,k-1}, w\right), w\right) + f\left(l - \lambda\left(x^{l}_{i,k-1}, w\right), w\right) \tag{5.9}$$

Il valore $f\left(x^{l}_{1,k-1}, w\right)$ è uguale alla somma $f\left(\lambda\left(x^{l}_{1,k-1}, w\right), w\right) + f\left(x^{l}_{1,k-1} - \lambda\left(x^{l}_{1,k-1}, w\right), w\right)$, per definizione di $\lambda(x^{l}_{1,k-1}, w)$, mentre, dalla seconda raggiungibilità, si ricava che il valore $f\left(l - \lambda\left(x^{l}_{1,k-1}, w\right), w\right)$ è maggiore o uguale al risultato della somma $f\left(x^{l}_{1,k-1} - \lambda\left(x^{l}_{1,k-1}, w\right), w\right) + f\left(x^{l}_{k}, w\right)$. Se è strettamente maggiore, allora la somma (5.9) è maggiore di $f\left(\lambda\left(x^{l}_{1,k-1}, w\right), w\right) + f\left(x^{l}_{1,k-1} - \lambda\left(x^{l}_{1,k-1}, w\right), w\right) + f\left(x^{l}_{k}, w\right)$ e quindi maggiore del risultato della somma $f\left(x^{l}_{1,k-1}, w\right) + f\left(x^{l}_{k}, w\right)$, in contraddizione con l'ipotesi secondo cui quest'ultima è la somma minima, fra quelle che danno $f(l, w)$ come risultato. Dunque, il risultato della somma (5.9), associata alla partizione $p^{2,l}$, minore di p^{l}_{min}, è uguale a $f\left(x^{l}_{1,k-1}, w\right) + f\left(x^{l}_{k}, w\right) = f(l, w)$, ma anche questo è in contrasto con l'ipotesi secondo cui p^{l}_{min} è la minima partizione la cui somma associata dia $f(l, w)$ come risultato.

La conseguenza è che se $(\lambda(x_2, w), x_2, l) \in DRN_v(w)$, allora si può essere sicuri che la somma $f(x_2, w) + f(x_1, w)$, che non è realizzata a causa del fatto che la condizione $C_{v,2}$ non è soddisfatta, non può essere la somma minima, per cui la proibizione può essere accettata. Nel seguito si denota con $CM_{v,1}$ la condizione $(\lambda(x_2, w), x_2, l) \in DRN_v(w)$, dove la lettera M è scelta per ricordare il legame con la minima somma.

Un risultato simile può essere ottenuto quando $x^{l}_{k} > x^{l}_{1,k-1} - \lambda(x^{l}_{1,k-1}, w) \geq \lambda(x^{l}_{1,k-1}, w)$ e l'ulteriore condizione, denotata con $CM_{v,2}$ e corrispondente a $(x^{l}_{1,k-1} - \lambda(x^{l}_{1,k-1}, w), x^{l}_{1,k-1}, l) \in DRN_v(w)$, è soddisfatta. La prova è del tutto simile ed utilizza i cutting patterns $B^{3,l}$ e $B^{x-\lambda}$, che generano rispettivamente l ed $x^{l}_{1,k-1} - \lambda(x^{l}_{1,k-1}, w)$, la partizione $p^{3,l} = \{B^{x-\lambda}, B^{3,l} - B^{x-\lambda}\}$ e la somma $f\left(x^{l}_{i,k-1} - \lambda\left(x^{l}_{1,k-1}, w\right), w\right) + f\left(l - x^{l}_{i,k-1} + \lambda\left(x^{l}_{i,k-1}, w\right), w\right)$ nel ruolo prima assunto dalla somma (5.9).

Quindi, se $(\lambda(x_2, w), x_2, l) \notin DRN_v(w)$, la proibizione non verrebbe accettata, ma se, invece, la condizione $(x_2 - \lambda(x_2, w), x_2, l) \in DRN_v(w)$ è soddisfatta, allora essa viene accettata.

Quindi, nel ciclo interno della procedura $\mathbf{P}^{++}$, sulle lunghezze, è necessario utilizzare non solo i valori x_1 di $\Lambda(x_2)$ che non superano $\lambda(x_2, w)$, ma anche, sulla base del discorso relativo alla condizione $CM_{v,1}$, quei valori di $\Lambda(x_2)$ che sono superiori di $\lambda(x_2, w)$, non maggiori di x_2 e tali che, rispetto alla corrente ampiezza w, $x_1 + x_2$ non è raggiungibile da $\lambda(x_2, w)$ oppure $x_1 + x_2 - \lambda(x_2, w)$ non è raggiungibile da $x_2 - \lambda(x_2, w)$, ed inoltre, sulla base del discorso relativo alla condizione $CM_{v,2}$, quei valori di $\Lambda(x_2)$ che sono maggiori di $x_2 - \lambda(x_2, w)$, non superiori di x_2 e tali che, rispetto al corrente valore di w, $x_1 + x_2$ non è raggiungibile da $x_2 - \lambda(x_2, w)$ oppure $x_1 + \lambda(x_2, w)$ non è raggiungibile da $\lambda(x_2, w)$.

Si definisce sia $N_1(x_2)$ come insieme dei valori x_1 per i quali l'eventuale proibizione della somma $f(x_1, w) + f(x_2, w)$ può essere accettata, secondo $CM_{v,1}$, sia $N_2(x_2)$ come insieme dei valori x_1 per i quali la proibizione della stessa somma può essere accettata, secondo $CM_{v,2}$. Dunque tali due insiemi $N_1(x_2)$ ed $N_2(x_2)$ possono essere espressi rispettivamente come $\{x_1 : x_1 + x_2 \in N^{**}_{v}(\lambda(x_2, w), w)$ ed $x_1 + x_2 - \lambda(x_2, w) \in N^{**}_{v}(x_2 - \lambda(x_2, w), w)\}$ e come $\{x_1 : x_1 + x_2 \in N^{**}_{v}(x_2 - \lambda(x_2, w), w)$ ed $x_1 + \lambda(x_2, w) \in N^{**}_{v}(\lambda(x_2, w), w)\}$.

Di conseguenza, l'insieme da usare nel ciclo interno sulle lunghezze è esprimibile formalmente come $\{x_1 \in \Lambda(x_2) : x_1 \leq \lambda(x_2, w)$ oppure $\lambda(x_2, w) < x_1 \leq x_2 - \lambda(x_2, w)$ ed $x_1 \notin N_1(x_2)$ oppure $x_2 - \lambda(x_2, w) < x_1 \leq x_2$ ed $x_1 \notin N_1(x_2) \cup N_2(x_2)\}$. Lo si denota con $\Lambda^{2}_{new}(x_2)$, dove il pedice 2 è utilizzato per ricordare che si è tenuto conto della sola condizione $C_{v,2}$. Da notare

che si è inglobata la condizione classica $C_{v,2}$, cioè $x_1 \leq \lambda(x_2, w)$, all'interno della definizione di $\Lambda^2_{new}(x_2)$.

Sia $N_1(x_2)$ che $N_2(x_2)$ possono essere calcolati utilizzano soltanto i due insiemi $N^{**}_v(\lambda(x_2, w), w)$ ed $N^{**}_v(x_2 - \lambda(x_2, w), w)$. La conseguenza importante che se ne trae è che non vanno usati insiemi diversi al variare dei possibili valori di x_1.

Inoltre, per calcolare $N_1(x_2)$, si può utilizzare l'insieme ridotto $\underline{N}^{**}_v(\lambda(x_2, w), w)$ anziché l'insieme pieno $N^{**}_v(\lambda(x_2, w), w)$ dato che, se $x_1 > \lambda(x_2, w)$, allora valgono le relazioni $x_1 + x_2 \geq 2 \cdot x_1 > 2 \cdot \lambda(x_2, w)$.

In modo simile, per calcolare $N_2(x_2)$, si può utilizzare l'insieme $\underline{N}^{**}_v(x_2 - \lambda(x_2, w), w)$ anziché l'insieme pieno $N^{**}_v(x_2 - \lambda(x_2, w), w)$. Infatti l'insieme $N_2(x_2)$ è utilizzato per evitare di considerare, nel ciclo interno sulle lunghezze, i valori x_1 maggiori di $x_2 - \lambda(x_2, w)$ e, per tali valori, si può affermare che $x_1 + x_2 \geq 2 \cdot x_1$, poiché $x_1 \leq x_2$ e quindi $x_1 + x_2 > 2 \cdot (x_2 - \lambda(x_2, w))$.

Inoltre, se $x_1 > x_2 - \lambda(x_2, w) \geq \lambda(x_2, w)$ allora $x_1 + \lambda(x_2, w) \geq 2 \cdot \lambda(x_2, w)$, da cui si deduce che si può usare l'insieme ridotto l'$\underline{N}^{**}_v(\lambda(x_2, w), w)$ al posto dell'insieme pieno $N^{**}_v(\lambda(x_2, w), w)$.

Si sottolinea ora che il calcolo dell'insieme $\Lambda^2_{new}(x_2)$ necessita di essere realizzato con una apposita procedura ottimizzata. Infatti, in termini di tempo computazionale, la peggiore implementazione potrebbe consistere semplicemente nella verifica di appartenenza di ogni possibile valore x_1 di $N_v(L_S, w)$ all'insieme $\Lambda^2_{new}(x_2)$. L'effetto sarebbe che, per ciascun valore di x_1, verrebbero eseguite molte operazioni, mentre lo scopo ultimo è evitare la realizzazione della somma $f(x_1, w) + f(x_2, w)$, che è un'operazione molto semplice[1]. La cosa importante qui era sottolineare queste considerazioni poiché è su di esse che si basa la scelta spiegata nel seguito, relativamente alla condizione $C_{v,1}$.

Si consideri ora il caso in cui sia $C_{v,1}$ a non essere soddisfatta, mentre $C_{v,2}$ lo sia. Ciò vuol dire che $\lambda(x^l_k, w) < x^l_k \leq \lambda(x^l_{1,k-1}, w)$. L'Esempio 3 ha mostrato che in generale bisognerebbe permettere la realizzazione della somma $f\left(x^l_{i,k-1}, w\right) + f\left(x^l_k, w\right)$ poiché essa potrebbe essere la somma minima, associata alla partizione minima p^l_{min}. Quindi, come nel caso in cui sia $C_{v,2}$ a non essere soddisfatta, l'obiettivo è costruire qualche regola semplice per accertarsi che la somma non realizzata non sia la somma minima.

Utilizzano una prova simile rispetto al caso di $C_{v,2}$, si può mostrare che, se la tripla $(\lambda(x_1, w), x_1, l)$ appartiene a $DRN_v(w)$, allora la somma $f(x_2, w) + f(x_1, w)$ non è la somma minima. Lo stesso risultato è ottenibile nel caso in cui $(x_1 - \lambda(x_1, w), x_1, l) \in DRN_v(w)$. Si denota dunque la prima condizione con $CM_{v,3}$ e la seconda con $CM_{v,4}$.

I relativi insiemi di valori di x_1 per i quali, nel ciclo interno sulle lunghezze, la proibizione può essere accettata sono denotati con $N_3(x_2)$ per $CM_{v,3}$ e con $N_4(x_2)$ per $CM_{v,3}$, e sono rispettivamente esprimibili come $\{x_1 : x_1 + x_2 \in N^{**}_v(\lambda(x_1, w), w)$ ed $x_2 + x_1 - \lambda(x_1, w) \in N^{**}_v(x_1 - \lambda(x_1, w), w)\}$ e come $\{x_1 : x_1 + x_2 \in N^{**}_v(x_1 - \lambda(x_1, w), w)$ ed $x_2 + \lambda(x_1, w) \in N^{**}_v(\lambda(x_1, w), w)\}$. Per calcolare $N_3(x_2)$ ed $N_4(x_2)$ bisogna dunque utilizzare gli insiemi $N^{**}_v(\lambda(x_1, w), w)$ ed $N^{**}_v(x_1 - \lambda(x_1, w), w)$, che sono associati a ciascun possibile valore di x_1. Dunque, stavolta, non può esserci alcun modo ottimizzato per calcolare $N_3(x_2)$ ed $N_4(x_2)$ e di conseguenza non è opportuno utilizzarli.

---

[1] I dettagli implementativi sulle modalità di realizzazione del calcolo di $\Lambda^2_{new}(x_2)$ possono essere trovati in:

Russo M (2007) Knapsack function computation. Technical report n. DIS-RO-MR-2-2007. Università degli Studi di Napoli, Federico II.

Tuttavia, è possibile sfruttare la condizione $l \in BN_v(w)$, denotata nel seguito con $CM_{v,5}$. Infatti è stato prima mostrato che $l \in BN_v(w)$ implica $(x, l, m) \in TRN_v(w)$ e quindi anche che $(l - x, l, m) \in TRN_v(w)$, per una qualsiasi coppia di ulteriori valori x ed m tali che $x \in N^*_v(l, w)$ ed $m \in N^{**}_v(l, w)$. Quindi, se $x_1 \in BN_v(w)$, allora entrambe le triple considerate sopra, con $x_1 = x^l_k$, vale a dire le triple $(\lambda(x_1, w), x_1, l)$ e $(x_1 - \lambda(x_1, w), x_1, l)$, appartengono a $TRN_v(w)$ e quindi anche a $DRN_v(w)$. La conseguenza è che la proibizione della somma $f(x_1, w) + f(x_2, w)$, secondo la condizione $C_{v,1}$, può essere accettata.

La conclusione è che un possibile insieme di valori x_1 da usare nel ciclo interno sulle lunghezze della procedura $\mathbf{P}^{++}$ è $\Lambda^1_{new}(x_2) = \{x_1 \leq \lambda(x_2, w) : x_1 \in \Lambda(x_2) \cup \neg\, BN_v(w)\}$ dove $\neg\, BN_v(w)$ è l'insieme di valori appartenenti ad $N_v(L_S, w)$ ma non a $BN_v(w)$.

Sono stati analizzati i due casi in cui una sola fra $C_{v,1}$ e $C_{v,2}$ è non soddisfatta, mentre l'altra lo è. Se, invece, entrambe non sono soddisfatte, è facile mostrare che, come prima, se almeno una fra le tre condizioni $CM_{v,1}$, $CM_{v,2}$ e $CM_{v,5}$ è soddisfatta, allora la somma $f(x_1, w) + f(x_2, w)$ non può essere la somma minima, con l'accorgimento di considerare $CM_{v,1}$ solo nel caso $x_1 > \lambda(x_2, w)$ e $CM_{v,2}$ solo nel caso $x_1 > x_2 - \lambda(x_2, w)$. Infatti, i ragionamenti fatti circa $\Lambda^2_{new}(x_2)$ restano validi anche se $x_1 \notin \Lambda(x_2)$. Allo stesso modo, anche se $x_1 > \lambda(x_2, w)$, la condizione $x_1 \in BN_v(w)$ continua ad implicare che $f(x_1, w) + f(x_2, w)$ non è la somma minima. Ciò significa che l'insieme da utilizzare nel ciclo interno sulle lunghezze è ottenibile come unione dei seguenti tre insiemi. Il primo è $\{x_1 \in \Lambda(x_2) : x_1 \leq \lambda(x_2, w)\}$, cioè l'insieme classico utilizzato nel caso di domanda illimitata. Il secondo è formalmente esprimibile come $\{x_1 \in N_v(L_S, w) : \lambda(x_2, w) < x_1 \leq x_2 - \lambda(x_2, w)$ ed $x_1 \notin N_1(x_2)$ oppure $x_2 - \lambda(x_2, w) < x_1 \leq x_2$ ed $x_1 \notin N_1(x_2) \cup N_2(x_2)\}$, e tiene conto delle condizioni $CM_{v,1}$ ed $CM_{v,2}$. Il terzo insieme è $\{x_1 \in N_v(L_S, w) : x_1 \leq x_2$ ed $x_1 \notin \Lambda(x_2) \cup BN_v(w)\}$, che tiene conto della condizione $CM_{v,5}$. L'insieme unione così ottenuto è denotato con $\Lambda_{new}(x_2)$.

Si spiega infine, come cambiare la definizione ed il calcolo dell'insieme $\Lambda_{new}(x_2)$ per tener conto anche dell'errore emerso dall'analisi dell'Esempio 1, per il caso in cui valga l'uguaglianza $f(l, w) = f(l - 1, w)$. In tal caso si può usare la posizione $\lambda(l, w) = \lambda(l - 1, w)$. Così facendo, non si può più affermare che, per ogni coppia di valori x_1 ed x_2 coinvolti nella somma $f(x_1, w) + f(x_2, w)$, valgano sia la relazione $\lambda(x_2, w) \leq x_2 - \lambda(x_2, w)$ che la relazione $\lambda(x_1, w) \leq x_1 - \lambda(x_1, w)$. Tuttavia, se per un valore x_2 la prima vale, allora i ragionamenti circa le condizioni $CM_{v,1}$ ed $CM_{v,2}$ restano validi, eccetto un punto trattato nel seguito, mentre se $\lambda(x_2, w) > x_2 - \lambda(x_2, w)$, essi restano validi scambiando il ruolo di $\lambda(x_2, w)$ ed $x_2 - \lambda(x_2, w)$. In modo simile, se per un valore x_1 la seconda relazione vale, le considerazioni circa la condizione $CM_{v,5}$ restano valide, mentre se $\lambda(x_1, w) > x_1 - \lambda(x_1, w)$, esse restano valide a patto di scambiare il ruolo di $\lambda(x_1, w)$ e $x_1 - \lambda(x_1, w)$.

L'unica differenza, per esempio per la condizione $CM_{v,1}$, è nella parte della prova che coinvolge la somma (5.9). Infatti non si può più affermare che $f\left(x^l_{i,k-1}, w\right)$ è uguale a $f\left(\lambda\left(x^l_{i,k-1}, w\right), w\right) + f\left(x^l_{i,k-1} - \lambda\left(x^l_{i,k-1}, w\right), w\right)$, poiché in tal caso si ha $f\left(x^l_{i,k-1}, w\right) = f\left(\lambda\left(x^l_{i,k-1}, w\right), w\right)$. Tuttavia, se $f\left(x^l_{i,k-1} - \lambda\left(x^l_{i,k-1}, w\right), w\right) = 0$, allora vale l'uguaglianza e nella prova non cambia nulla, mentre se $f\left(x^l_{i,k-1} - \lambda\left(x^l_{i,k-1}, w\right), w\right) > 0$, non solo questo non rende comunque possibile che il valore $f\left(l - \lambda\left(x^l_{i,k-1}, w\right), w\right)$ sia inferiore alla somma $f\left(x^l_{i,k-1} - \lambda\left(x^l_{i,k-1}, w\right), w\right) + f\left(x^l_k, w\right)$, ma piuttosto, implica che tale valore è sicuramente maggiore della somma, mentre nella prova sopra si era concluso che valeva l'uguaglianza proprio per il

fatto che il segno di “maggiore” determinava una contraddizione con l’ipotesi che la somma $f\left(x_{i,k-1}^{l}, w\right) + f\left(x_{k}^{l}, w\right)$ dia $f(l, w)$ come risultato, e ciò vuol dire che la prova resta valida.

La conclusione ultima del discorso è che in generale la definizione di $\Lambda_{new}(x_2)$ può essere mantenuta inalterata, dando però al minimo fra $\lambda(x_2, w)$ ed $x_2 - \lambda(x_2, w)$ il ruolo prima assunto dal valore $\lambda(x_2, w)$ ed al massimo fra di essi il ruolo assunto prima dal valore $x_2 - \lambda(x_2, w)$, mentre non c’è bisogno di alcun cambio circa i valori $\lambda(x_1, w)$ ed $x_1 - \lambda(x_1, w)$, poiché solo x_1 è stato usato per definire $\Lambda_{new}(x_2)$, e pure se si fossero usati $\lambda(x_1, w)$ ed $x_1 - \lambda(x_1, w)$, sarebbe bastato far assumere al minimo fra i due il ruolo precedentemente dato a $\lambda(x_1, w)$ ed al massimo fra essi il ruolo precedentemente dato ad $x_1 - \lambda(x_1, w)$.

Si noti che gli stessi risultati si ottengono anche settando $\lambda(l, w)$ uguale ad un qualsiasi valore appartenente ad $N_v(L_S, w)$, ma inferiore a $\lambda(l - 1, w)$ e non superiore ad l. Tuttavia il valore minimo possibile, cioè $\lambda(l - 1, w)$, è quello cui sono associate maggiori probabilità di ridurre, più degli altri, il numero di somme realizzate che coinvolgano l nel ruolo di x_2.

5.5 Risultati sperimentali

I risultati di seguito riportati hanno lo scopo di mostrare i miglioramenti derivanti dall'utilizzo degli insiemi $\Lambda(l)$ ed $\Omega(w)$, in termini di tempo computazionale. A tal fine, sono elencati i tempi computazionali associati a tre gruppi di istanze. Quelle del primo gruppo sono di tipo un-constrained, quelle del secondo di tipo semi-constrained ed infine quelle del terzo di tipo constrained.

Relativamente al primo gruppo, si comparano i risultati ottenuti con la vecchia, ma corretta, procedura di Gilmore e Gomory e denotati in Tabella 5.2 con T_{old} , con quelli, denotati in Tabella 5.2 con T_P ed ottenuti con la procedura **P**, che sfrutta cioè anche gli insiemi $\Lambda(l)$ ed $\Omega(w)$. Relativamente al secondo gruppo, si comparano i risultati, denotati in Tabella 5.3 con T_{P-} ed ottenuti con la procedura $\mathbf{P}^{+}$, ma senza sfruttare gli insiemi $\Lambda(l)$ ed $\Omega(w)$, con quelli, denotati in Tabella 5.3 con T_{P+} ed ottenuti con la procedura $\mathbf{P}^{+}$ completa, cioè sfruttando anche i suddetti insiemi. Infine, per il terzo gruppo di istanze, si comparano i risultati ottenuti con la procedura base del precedente Capitolo e denotati in Tabella 5.4 con T_R, con i risultati ottenuti con la procedura $\mathbf{P}^{++}$ e denotati in Tabella 5.4 con T_{P++} . Tutti i tempi computazionali sono espressi in millisecondi, utilizzando tutte le cifre necessarie. Tutti i risultati sono stati ottenuti su un sistema Pentium IV 1.80 GHz, 1GB Ram, Windows XP SP2, sfruttando un codice C++ appositamente sviluppato.

Le istanze del primo e del terzo gruppo sono state estratte dalla recente collezione di Fekete e van der Veen (2007). Le istanze del secondo gruppo sono state costruire a partire da quelle trattate nel Capitolo precedente ed estratte dalla suddetta recente collezione. In particolare, ciascuna di esse è stata modificata aumentando, per ciascun tipo t_i, la domanda d_i al massimo fra $\lfloor L_S / l_i \rfloor$ e $\lfloor W_S / w_i \rfloor$ se d_i era inferiore ad esso. Le istanze modificate così ottenute sono denotate utilizzando un asterisco.

Fra le istanze di tutti i tre gruppi, alcune sono pesate ed altre non pesate. Inoltre, alcune di esse sono di dimensione medio-piccola ed altre di dimensione elevata.

La Tabella 5.2 riporta i risultati per il primo gruppo di istanze, di tipo un-constrained. Si può notare che per tutte le undici istanze non pesate elencate, la nuova procedura **P** è sempre la più veloce, eccetto che per l'istanza UU1, per la quali il risultato è lo stesso che si ottiene con la procedura classica. Per le istanze pesate, si osserva un comportamento diverso. Solo per cinque istanze, da UW6 a UW10, la nuova procedura è la più veloce. Comunque, per le altre sei istanze elencate, cioè da UW1 ad UW5, ed UW11, la differenza fra i due casi è piccola in percentuale. Si noti inoltre che queste sei istanze sono quelle con tempi computazionali minimi nel sotto-gruppo delle istanze pesate, mente per le cinque istanze per le quali la procedura **P** è più veloce, i tempi computazionali sono i massimi nel sotto-gruppo.

Istanza	T_{old} (msec)	T_P (msec)	***Istanza***	T_{old} (msec)	T_P (msec)
Istanze non pesate			*Istanze pesate*		
UU1	953	953	**UW1**	796	857
UU2	2:641	2:266	**UW2**	1:375	1:415
UU3	4:734	4:187	**UW3**	1:422	1:587
UU4	6:156	4:703	**UW4**	4:531	4:635
UU5	10:890	7:532	**UW5**	5:578	5:791
UU6	15:453	11:562	**UW6**	10:454	10:110
UU7	22:156	11:953	**UW7**	16:985	16:156
UU8	28:046	15:937	**UW8**	36:641	28:859
UU9	24:046	43:968	**UW9**	43:422	38:375
UU10	1.57:906	48:407	**UW10**	1.06:859	49:031
UU11	12.04:296	2.40:750	**UW11**	1:204	1:237

Tabella 5.2 Performances della procedura **P** su istanze di tipo un-constrained

La Tabella 5.3 riporta i risultati per il secondo gruppo di istanze, di tipo semi-constrained. Si può notare che, per tutte tali istanze, lo sfruttamento degli insiemi $\Lambda(l)$ ed $\Omega(w)$ determinano un miglioramento in termini di tempo computazionale. In particolare, per le istanze da CW1* a CW9*, il tempo computazionale è circa la metà di quello associato al caso in cui i suddetti insiemi non vengono utilizzati. Lo stesso avviene per le istanze APT43*, da APT45* a APT48*, ed infine APT35*. Per le istanze APT31* ed APT49* il miglioramento è ancora superiore, mentre per le istanze da APT40* a APT44* e per tutte quelle del sotto-gruppo di non pesate, di dimensioni tanto medio-piccole quanto elevate, eccetto APT35*, il miglioramento è inferiore.

Istanza	T_{P-} (msec)	T_{P+} (msec)	***Istanza***	T_{P-} (msec)	T_{P+} (msec)
Istanze pesate di dimensione medio-piccola			*Istanze pesate di dimensione elevata*		
A1*[a]	94	47	**APT40*[a]**	1.24:812	1.08:453
A2*[a]	78	62	**APT41*[a]**	1.26:766	30:890
A3*[a]	172	156	**APT43*[a]**	3.50:797	1.53:313
CW1*[a]	390	218	**APT44*[a]**	47:235	29:406
CW2*[a]	547	312	**APT45*[a]**	30:750	16:672
CW3*[a]	1:312	750	**APT46*[a]**	1.25:265	40:500
CW4*[a]	2:234	1:265	**APT47*[a]**	1.16:047	34:625
CW5*[a]	3:406	1:781	**APT48*[a]**	2.02:016	1.07:359
CW6*[a]	13:812	5:062	**APT49*[a]**	2.04:469	40:391
CW7*[a]	2:438	1:250			
CW8*[a]	3:703	1:922			
CW9*[a]	3:687	1:797			
Istanze non pesate di dimensione medio-piccola			*Istanze non pesate di dimensione elevata*		
A4*[a]	360	313	**APT30*[a]**	1.01:641	50:453
A5*[a]	625	500	**APT31*[a]**	9.41:672	6.10:172
CU1*[a]	234	203	**APT32*[a]**	19:328	17:672
CU2*[a]	547	406	**APT33*[a]**	1.57:578	1.21:125
CU3*[a]	750	609	**APT34*[a]**	2.18:437	1.29:547
CU4*[a]	3:094	2:266	**APT35*[a]**	4.12:094	2.09:063
CU5*[a]	3:937	2:704	**APT36*[a]**	34:609	26:328
CU6*[a]	4:235	2:719	**APT37*[a]**	3.14:734	2.05:485
CU7*[a]	1:828	1:344	**APT38*[a]**	2.00:265	1.31:062
CU8*[a]	4:016	2:516	**APT39*[a]**	1.10:265	44:312
CU9*[a]	3:234	2:312			

[a] Instanza opportunamente modificata, con domande aumentate, tale da renderla semi-constrained

Tabella 5.3 Performances della procedura $\mathbf{P}^{+}$ su istanze di tipo semi-constrained

La Tabella 5.4 riporta i risultati per il terzo gruppo di istanze, di tipo constrained. Questa volta, la procedura $\mathbf{P}^{++}$ non è sempre la più veloce. In particolare, per le istanze A3, A4, CU1, CU4, CU5, CU7, CU8 e CU9 essa è più lenta. Per le istanze A2, CU2, CU3 e CU6 si può notare un tempo computazionale molto simile fra le due procedure. Per tutte le altre istanze, la procedura $\mathbf{P}^{++}$ è la più veloce. In particolare, per le istanze CW6 ed APT41 il tempo computazionale T_{P++} è circa la metà rispetto a T_R.

Istanza	T_R (msec)	T_{P++} (msec)	***Istanza***	T_R (msec)	T_{P++} (msec)
Istanze pesate di dimensione medio-piccola			*Istanze pesate di dimensione elevata*		
A1	78	62	**APT40**	1.04:344	56:062
A2	78	78	**APT41**	1.17:875	36:437
A3	141	187	**APT43**	3.09:656	1.47:407
CW1	313	235	**APT44**	35:172	25:344
CW2	500	391	**APT45**	26:000	16:953
CW3	1:250	1:016	**APT46**	1.12:297	44:844
CW4	2:250	1:578	**APT47**	1.03:968	40:344
CW5	3:407	2:093	**APT48**	1.31:703	53:766
CW6	13:468	6:312	**APT49**	1.52:312	51:500
CW7	2:391	1:547			
CW8	3:531	2:359			
CW9	3:484	1:985			
Istanze non pesate di dimensione medio-piccola			*Istanze non pesate di dimensione elevata*		
A4	234	282	**APT30**	46:563	40:968
A5	469	407	**APT31**	8.33:391	6.33:657
CU1	219	281	**APT32**	14:000	12:828
CU2	532	531	**APT33**	1.37:641	1.13:031
CU3	719	718	**APT34**	2.03:969	1.29:360
CU4	2:906	3:328	**APT35**	3.56:687	3.01:922
CU5	3:687	4:235	**APT36**	30:828	27:687
CU6	4:109	4:188	**APT37**	2.44:406	2.03:390
CU7	1:656	1:953	**APT38**	1.39:687	1.22:859
CU8	3:812	4:157	**APT39**	1.25:407	56:640
CU9	3:141	3:422			

Tabella 5.4 Performances della procedura $\mathbf{P}^{++}$ su istanze di tipo constrained

5.6 Conclusioni

In questo Capitolo, sono state trattate le knapsack functions in qualità di upper bounds utili nelle versioni constrained del problema di Cutting Stock bidimensionale a ghigliottina. In particolare, dopo un riassunto sulla knapsack function classica e dei miglioramenti proposti nel Capitolo precedente, è stata realizzata un'ampia analisi circa le modalità di unificare tali miglioramenti con l'idea alla base del lavoro di Gilmore e Gomory che ha introdotto la knapsack function. I risultati ottenuti sono incoraggianti in termini di tempo computazionale necessario al calcolo della tabella di valori della knapsack function.

PARTE III

Applicazioni nella realtà industriale

Capitolo 6

Applicazioni nel settore della trasformazione del vetro piano

Questo Capitolo presenta una panoramica degli elementi che costituiscono un ambiente produttivo di trasformazione del vetro piano. In particolare, nel primo paragrafo, vengono elencate le principali tipologie di prodotto. Il secondo paragrafo descrive invece le fasi lavorative di cui si compone il processo. Nel terzo paragrafo si passano brevemente in rassegna le risorse, umane e materiali, a disposizione. Il quarto paragrafo descrive le fasi attraverso cui vengono organizzate le attività produttive, mentre il quinto paragrafo elenca i principali problemi di ottimizzazione che emergono nel contesto industriale descritto, con particolare enfasi agli aspetti correlati al problema di taglio bidimensionale.

6.1 Materie prime, semilavorati e prodotti finiti

Si possono distinguere tre categorie di oggetti, le materie prime, i semilavorati ed i prodotti finiti. Le principali materie prime sono di quattro tipi:

- i vetri
- gli strati di PVB (Polinivil Butirrato)
- le canaline metalliche
- i materiali adesivi (*butile, thiover*)

Sui differenti tipi di vetri si torna più avanti. Relativamente al materiale plastico PVB, nel secondo Capitolo si è già indicato che, unendo strati di vetro puro, intervallati da strati di PVB, si ottengono i cosiddetti vetri stratificati.

Le canaline metalliche servono per l'assemblaggio dei vetri camera, di cui si parla più avanti. Le canaline di lunghezza voluta sono ottenute con taglio mono-dimensionale a partire da barre (o aste) di lunghezza standard. Anche i materiali adesivi sono utilizzati per l'assemblaggio dei vetri camera.

Si chiamano semilavorati quegli oggetti ottenuti per assemblaggio, o per trasformazione, di materie di prime attraverso opportune lavorazioni, ma che non rappresentano ancora l'oggetto ultimo da vendere, cioè il prodotto finito. Spesso la distinzione è solo formale e non sostanziale, nel senso che dipende da ciò che un cliente ha ordinato. Ad esempio, un vetro stratificato può essere ottenuto perché va poi assemblato con un altro vetro così da ottenere il vetro camera ordinato dal cliente. D'altro canto, un secondo cliente può aver ordinato un vetro stratificato. Ciò vuol dire che uno stesso tipo di oggetto può fungere da semilavorato o da prodotto finito, in funzione dell'ordine per il cui espletamento è stato assembrato o trasformato. Inoltre una lastra di vetro stratificato può essere ottenuta sia attraverso una apposita lavorazione di stratifica, interna al contesto industriale, sia attraverso l'acquisto di lastre confezionate. Nel primo caso, la lastra di vetro stratificato è ottenuta con lavorazioni interne, e rappresenta perciò un semilavorato, mentre nel secondo caso è, di fatto, una materia prima.

Talvolta, un'industria può essere incaricata anche soltanto di realizzare una particolare operazione di trasformazione su pezzi di vetro tagliati in altre industrie. In tal caso è l'industria committente a realizzare il taglio e le altre eventuali trasformazioni, e per l'industria incaricata i pezzi di vetro rappresentano una materia prima, mentre in generale essi sono un semilavorato ottenuto internamente attraverso operazioni di taglio a partire dalle lastre.

Nel secondo Capitolo, le tipologie di vetro sono state distine in quattro classi, sulla base delle relative tecnologie di taglio. Nel seguito si presenta una classificazione dal punto di vista dei prodotti vendibili. Si distinugono dunque le seguenti tipologie, schematizzate anche in figura 6.1.

- *vetri float*: le lastre sono realizzate con spessori standard di 2, 3, 4, 5, 6, 8, 10, 12, 15, 19 e 22 mm. Questo tipo di vetro è considerato pericoloso per l'uso in applicazioni architettoniche poiché tende a rompersi in grossi pezzi taglienti (se non temprato) che possono causare gravi incidenti. Le normative edilizie pongono in genere delle limitazioni all'uso di questo vetro in situazioni rischiose quali bagni, pannelli di porte, uscite antincendio e nelle scuole. Per spessori superiori ai 10-12 mm, è necessaria una tecnologia di taglio blindato.

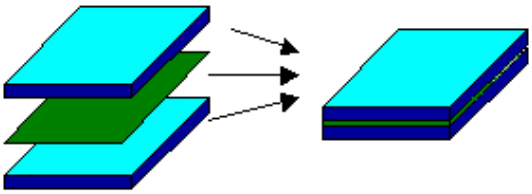
- *vetri stratificati*: sono ottenuti, come già detto, attraverso più strati di vetro float, intervallati da strati di plastico (PVB). In genere gli strati di vetro sono due, come

in Figura 6.1a. Quando il numero di strati è superiore, è necessario utilizzare tecnologie di taglio blindato, sia perché è impossibile incidere gli strati di vetro intermedi, sia per l'elevato spessore che ne deriva.
Questo vetro, se soggetto a violenti urti, si lesiona ma non si frantuma. Tuttavia esso è poco adatto per il montaggio interno-esterno perché le temperature differenti determinano una differente dilatazione del vetro interno rispetto a quello esterno, causando uno squilibrio che comporta la lesione del vetro. In questi casi, si può utilizzare, come vetro sul lato esterno, un vetro temprato, oppure si può utilizzare direttamente un vetro camera.

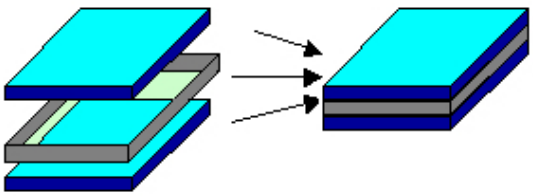
- *vetri camera*: sono ottenuti attraverso due vetri, float o stratificati, con una camera d'aria interposta, racchiusa da una superficie metallica ottenuta piegando una canalina, come in Figura 6.1b. Per aumentare le capacità isolanti di un vetro camera, talvolta, si sigillano vetri con più camere, dunque utilizzando più di due vetri. Le camere possono inoltre essere riempite di gas argon.

- *vetri intelaiati*: sono ottenuti innestando un vetro all'interno di un apposito telaio metallico (Figura 6.1c).

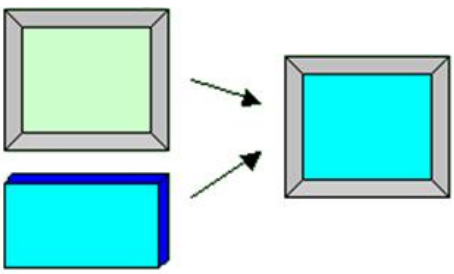
- *vetri trasformati*: si tratta di vetri su cui realizzare una o più lavorazioni di trasformazione su apposita commissione del cliente. Il committente può a sua volta essere un'altra industria che non ha a disposizione gli appositi macchinari. Si parla allora di "lavorazione per conto terzi". In Figura 6.1d si fa l'esempio della tempra, che irrobustisce il vetro, e dell'arrotatura, che leviga i bordi del vetro.



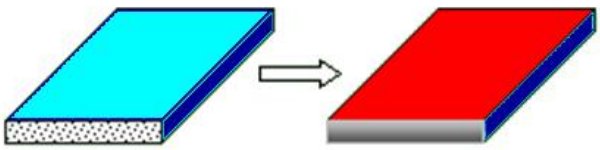
a) vetro stratificati (vetri float + PVB)



b) vetro camera (vetri + canalina per camera d'aria)



c) intelaiatura (vetro + telaio)



d) trasformazioni per conto terzi (esempio: arrotatura + tempra)

Figura 6.1 Classificazione dei vetri per prodotto finito

Per quanto riguarda le tipologie di vetri float, ne esistono di diversi colori, o con particolari proprietà, come quella di avere una bassa emissività, per la riduzione della dispersione termica, o di avere una elevata capacità di riflessione, per ridurre il riscaldamento in ambienti esposti al sole. Alcuni vetri float costano circa 7-8 euro per metro quadro. Altri arrivano a 40-50 euro per metro quadro, per cui nei relativi problemi di taglio bidimensionale, la voce di costo della materia prima è predominante.

6.1.1 Alberi di assemblaggio

Ogni prodotto finito si ottiene attraverso una lavorazione principale di assemblaggio, a partire dai suoi componenti "diretti", da usare ciascuno secondo una quantità di N_i unità (atomiche) al fine di ottenere un'unità (atomica) di prodotto finito. Ciascun componente è una materia prima o a sua volta un semilavorato, ottenuto cioè attraverso l'assemblaggio di ulteriori sub-componenti. Questo scenario determina l'associazione a ciascun prodotto finito di un albero, detto *albero di assemblaggio* o *distinta base*. Per ogni prodotto finito o semilavorato è definita univocamente la lavorazione principale che ne realizza l'assemblaggio.

Su ogni prodotto, semilavorato o materia prima, secondo l'esigenza del cliente, possono essere realizzate lavorazioni aggiuntive, su cui si torna nel paragrafo successivo.
In figura 6.2 è riportato uno schema di albero di assemblaggio per il caso del vetro intelaiato, in cui sono riportate le lavorazioni aggiuntive associate a ciascun nodo, come si ipotizza sia stato indicato dal committente che ne abbia effettuato l'ordine.

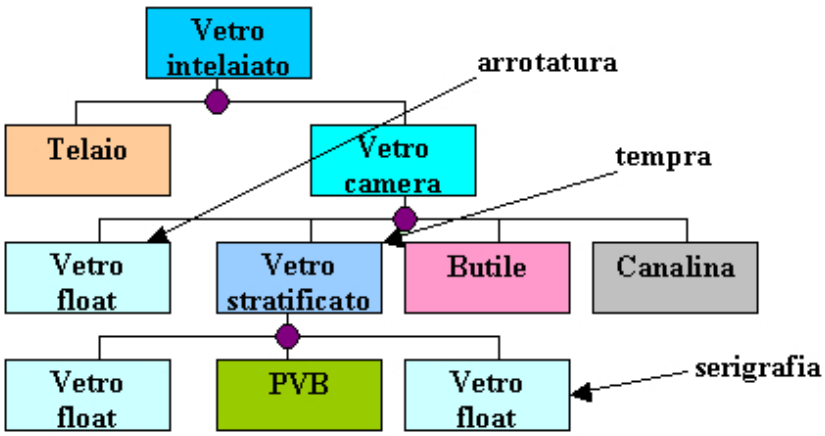


Figura 6.2 Albero di assemblaggio

Le caratteristiche geometriche (base, altezza, spessore) di un'istanza di prodotto finito si riflettono sulle caratteristiche geometriche dei semilavorati associati ai vari nodi dell'albero relativo al tipo di prodotto finito in questione.

6.2 Le lavorazioni

Si distinguono due categorie di lavorazioni: *principali* ed *aggiuntive*. Le prime vanno necessariamente effettuate a seconda del prodotto finito che si vuole ottenere. Le seconde vengono realizzate secondo le richieste del cliente e sono in genere tese ad aggiungere qualità fisiche o decorative al prodotto.

6.2.1 Lavorazioni principali

Le lavorazioni principali sono quelle di taglio e quelle di *assemblaggio* di vario tipo. Per certi versi possono essere accomunate, se viste come lavorazioni che determinano il passaggio da un tipo (o più tipi) di semilavorato ad un altro. La differenza fondamentale consiste nel fatto che, mentre negli assemblaggi è nota e fissata la quantità di semilavorati "in ingresso", nel caso del taglio ciò non avviene. Anzi, le lavorazioni di taglio possono produrre sfridi di materiale riutilizzabile, catalogabili di fatto come materia prima, ma con grandezze geometriche di valore inferiore.

Si possono distinguere cinque lavorazioni di taglio. Per le lavorazioni di taglio bidimensionale, le modalità operative sono state già descritte nel secondo Capitolo.

Taglio float: su vetri non stratificati di spessore limitato, cioè entro i 10-12 mm. Il termine float ha origine dal nome della lavorazione con cui i fornitori producono le lastre di partenza.

Taglio stratificato: su vetri stratificati di spessore limitato, in genere al massimo 8 mm per ciascuno dei due strati, per un totale che può arrivare ai 20 mm se si considera anche lo strato di plastico, che può arrivare a quattro millimetri.

Taglio stampato: su vetri, non stratificati, che hanno la particolarità di essere decorati. Questo comporta la necessità, nei piani di taglio, di seguire un orientamento geometrico prefissato per i pezzi che si vogliono estrarre dal taglio.

Taglio blindato: su vetri, stratificati o non, di spessore elevato.

Taglio canaline: su aste di canaline. Per ogni camera di un vetro-camera si taglia una canalina di lunghezza pari al perimetro del vetro-camera, per poi effettuare le piegature in corrispondenza degli angoli (o anche delle curve nel caso di sagome curvilinee) della sagoma, che risulta quasi sempre essere un rettangolo. E' l'unica lavorazione di taglio mono-dimensionale.

Le lavorazioni di assemblaggio vero e proprio sono tre.

Stratifica: per la realizzazione di un vetro stratificato a partire da due vetri float ed uno o più strati di plastico. L'utilizzo di più strati di plastico ha l'obiettivo di ottenere lo spessore di plastico voluto a partire da quelli disponibili. Più in generale un vetro stratificato, come già detto in precedenza, può essere formato da più di due vetri, con uno strato di plastico fra un vetro e l'altro. In talune applicazioni si arriva anche a spessori totali di 50 mm, con particolare riferimento ai vetri antiproiettile. Questa lavorazione è realizzata all'interno di apposite camere con temperature e pressioni elevate (camere iperbariche), con lo scopo di eliminare l'aria presente fra i vari strati, sia per motivi estetici che per motivi di robustezza.

Lavaggio e sigillatura: per la realizzazione dei vetri camera attraverso l'assemblaggio di vetri, stratificati o non, e canaline. All'interno delle camere vengono inseriti sali disidratanti, per evitare fenomeni di condensa. Una prima sigillatura è effettuata con il butile, posto fra la canalina e le due lastre di vetro. In una

seconda fase, il thiovil (che è un polisolfuro) viene posto lungo il perimetro esterno, per coprire le canaline. All'interno della camera possono essere introdotti elementi decorativi, o tendine parasole, pilotate da dispositivi elettrici. Inoltre vi si può introdurre argon per aumentare le capacità isolanti.

Intelaiatura (o *Incollaggio cellule*): per la realizzazione di un vetro intelaiato.

6.2.2 Lavorazioni aggiuntive

Le lavorazioni aggiuntive per la maggior parte sono effettuate sui pezzi di vetro non stratificati, ottenuti da lavorazioni di taglio. Le lavorazioni aggiuntive sono delle seguenti tipologie: *sagomatura, molatura, foratura, decorazioni, tempra, incollaggio cellule.*

La *sagomatura* consiste nell'estrazione di un pezzo di vetro di forma non rettangolare, a partire da un pezzo rettangolare appositamente tagliato con dimensioni sufficientemente elevate. In termini di taglio bidimensionale, esistono macchinari in grado di operare incisioni di forma qualsiasi. Quando questo non è possibile, è necessario decidere le dimensioni del pezzo rettangolare da cui estrarre la sagoma desiderata attraverso la sagomatura. In tal caso, le dimensioni del rettangolo possono non essere univoche, dato che dipendono dalla rotazione decisa per la sagoma, supponendo che essa non sia perfettamente circolare. In modo simile a quanto fatto per la gestione delle dipendenze fra taglio bidimensionale ed arrotatura, nel secondo Capitolo, è possibile utilizzare diverse rotazioni, come per esempio quelle che determinano minimi locali per la relativa area rettangolare, per poi esplodere il tipo di pezzo in più sotto-tipi, con domanda condivisa.

I principali tipi di *molatura* sono descritti nel seguito.

Sfilettatura: eliminazione meccanica o manuale del filo o degli spigoli taglienti al bordo delle lastre.

Molatura a filo grezzo: abrasione dei bordi di una lastra, ottenuta con nastri o mole di pietra, di diamante, di grana grossolana per eliminare le irregolarità del taglio.

Molatura a filo lucido: fase successiva alla precedente, realizzata con lo scopo di eliminare le disomogeneità più piccole, ottenendo inoltre una lucidatura con mole diamantate a grana finissima o polveri di pomice o di ossido di cerio.

Molatura a filo lucido industriale: ottenuto dalla molatura del bordo delle lastre con mole di adeguata finezza, senza successive lavorazioni. L'aspetto del vetro diventa semiopaco, ma il bordo ha una buona finitura.

Molatura a smussi e a bisello: lavorazione dei bordi di una lastra che interessa non solo i bordi della lastra ma anche la sua superficie, poiché viene eseguita con un angolo inferiore ai 90° rispetto ad essa. E' una lavorazione molto delicata e appariscente, e viene usata per prodotti di pregio nell'arredamento di interni.

Bisellatura: asportazione a mezzo di limatura o di tornitura degli spigoli vivi in seguito ad una lavorazione di rifinitura estetica riservata a prodotti di elevata qualità.

Le *forature* e le *incisioni*, nonché le *tacche*, determinano fori di forma variegata nel vetro e sono realizzate per poter applicare maniglie, viti, bulloni, e cose simili, al fine di produrre porte di vario genere. Vengono eseguite con apparecchiature automatiche o montate su flessibili manuali, come pure con getti di sabbia o con acido specifico.

Le decorazioni possono essere realizzate attraverso le seguenti lavorazioni:

Sabbiatura: decorazioni a getto di sabbia abrasiva che permette di ottenere dei disegni più o meno complessi a seconda delle specifiche esigenze.

Satinatura: decorazioni all'acido, con le quali si possono ottenere effetti di luce diffusa o parziale, di opacizzazione della lastra, oppure incisioni, più o meno profonde.

Smaltatura: processo mediante il quale si colora completamente la faccia di un vetro, attraverso un'applicazione a rullo o a spruzzo. Il colore può essere stabilizzato con un ulteriore trattamento termico.

Serigrafia: è utilizzata per riprodurre, attraverso l'applicazione di vernici e smalti, disegni, scritte, decorazioni a scopo artistico o funzionale con l'uso di telai e appropriata tecnica. Anche in tal caso può essere utile un ulteriore trattamento termico di stabilizzazione.

Opacizzazione: attraverso l'applicazione di un deposito opacizzante, inalterabile anche se esposto agli agenti atmosferici, consente anche l'incollaggio di pannelli coibentanti.

Altre lavorazioni aggiuntive sono i trattamenti termici**,** cioè la *tempra*, l'*indurimento* e la *stratifica*. La tempra del vetro è necessariamente effettuata a valle delle altre lavorazioni e consiste in un processo termico basato su elevate temperature, allo scopo di conferire al vetro migliori caratteristiche di resistenza, in particolare relativamente alla flessione.

Al raggiungimento di una temperatura prossima a quella che determina il rammollimento del vetro, questo viene estratto dal formo e bruscamente raffreddato. Le tensioni così indotte sono di trazione all'interno e di compressione sullo strato superficiale esterno. Il vetro temprato ha la caratteristica di ridursi in piccoli frammenti non taglienti quando si rompe, e perciò è considerato un vetro di sicurezza.

Il processo di *indurimento* è simile alla tempra, con la differenza che il raffreddamento viene eseguito più lentamente rispetto ad un pari spessore di vetro temprato. Il vetro indurito ha il vantaggio di non risentire di problemi di rottura spontanea, mantenendo dunque una resistenza meccanica maggiore rispetto al vetro temprato. Lo svantaggio consiste nel fatto che, in caso di rottura, questa produce pezzi grossolani e, per tale motivo, il vetro indurito non è classificato come vetro di sicurezza.

Le ultime lavorazioni aggiuntive qui descritte sono la *curvatura* e l'*argentatura*.

Curvatura: la curvatura o bombatura delle lastre, adottata nel campo dell'edilizia e dell'arredamento, si ottiene mediante un processo di fabbricazione complesso che implica una attenta precisione delle misure sia del pezzo di vetro che del telaio di contenimento. Infatti il pezzo di vetro viene tagliato secondo la forma e le dimensioni dello sviluppo che assumerà una volta curvato.

Argentatura: è un trattamento di deposito, sulla superficie del vetro, di argento metallico (per precipitazione di nitrato d'argento) al fine di renderla perfettamente riflettente (a specchio). La protezione della superficie si ottiene mediante successiva verniciatura.

Quasi tutte le lavorazioni aggiuntive vanno effettuate su vetri non stratificati. Fa eccezione la *sabbiatura*, che può essere fatta anche su vetri stratificati e comunque non su vetri-camera. In ogni caso la sabbiatura non può essere effettuata prima della stratifica.

Le lavorazioni aggiuntive sono soggette a dipendenze di natura tecnologica. Ciò implica la necessità che una data coppia di lavorazioni sia realizzata in un ordine ben preciso. Dunque l'insieme delle dipendenze definisce una relazione d'ordine parziale sull'insieme delle lavorazioni aggiuntive.

Nella Figura 6.3 si riporta uno schema delle dipendenze, tenendo però conto solo delle lavorazioni realizzate con maggiore frequenza, per motivi di semplicità. Sono inoltre inserite anche le lavorazioni di assemblaggio, indicate in modo sottolineato.

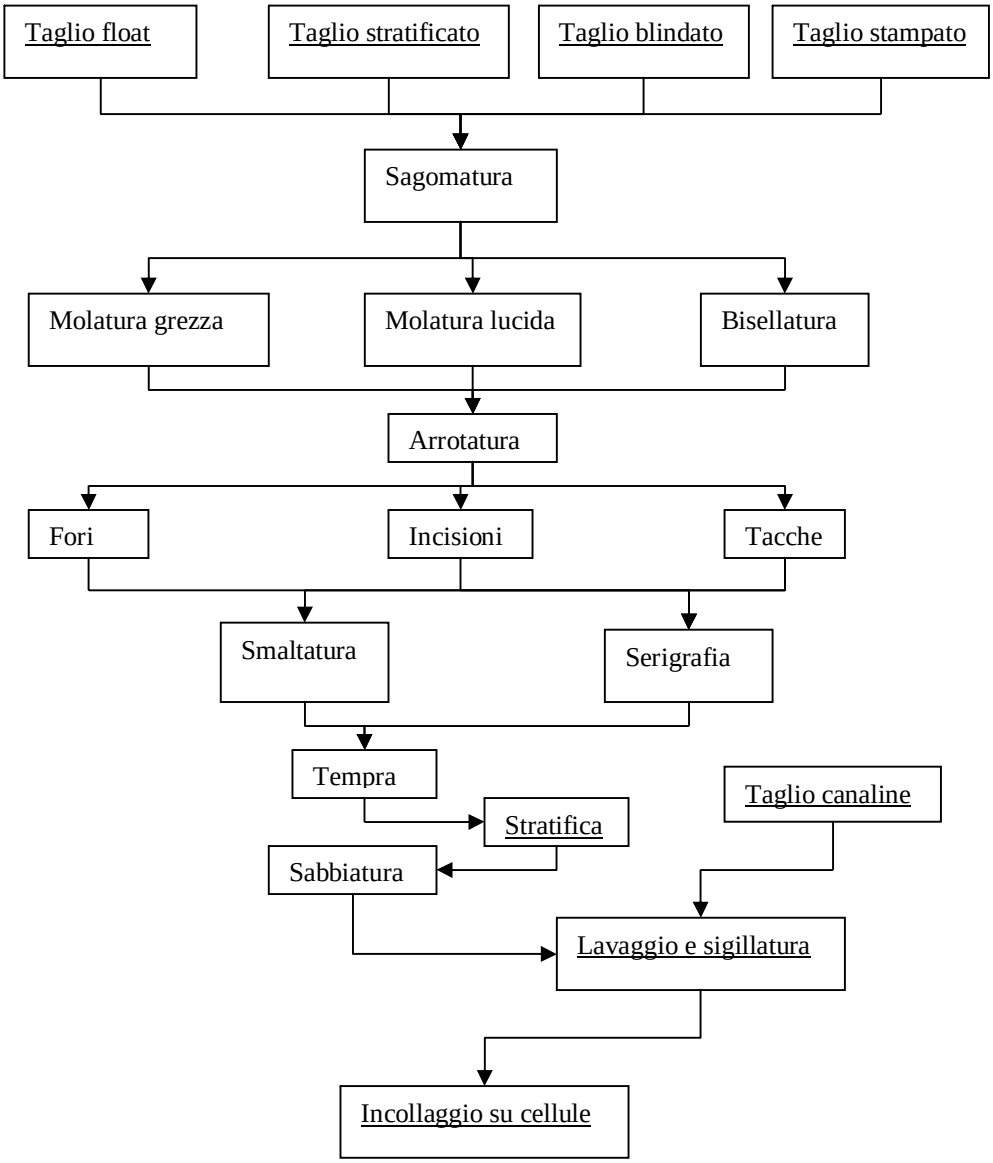


Figura 6.3 Dipendenze fra le lavorazioni

6.3 Le risorse

Vi sono tre tipi di risorse permanenti, vale a dire uomini, macchinari ed arnesi, dove la distinzione fra macchinari ed arnesi è quella che i primi sono fissi ed i secondi mobili.

6.3.1 Uomini e macchinari

Relativamente alle risorse umane ed ai macchinari, l'aspetto rilevante riguarda la possibilità di assegnare tali risorse alle diverse lavorazioni. Infatti ogni lavorazione necessità, per il suo espletamento, di un insieme di risorse, non necessariamente prestabilito, soprattutto in termini di risorse umane.

Talune fasi lavorative possono non necessitarne, come nel caso di asciugatura all'aria aperta di un vetro camera appena assemblato.

In termini di risorse umane, è interessante notare che le lavorazioni possono richiedere tempi differenti a seconda di quale sia l'insieme di risorse umane ad esse associato, sia relativamente al numero che all'esperienza e alle capacità delle singole persone. Questo ha ovviamente degli effetti sui problemi di assegnazione e di schedulazione che emergono nell'ambiente produttivo.

Relativamente ai macchinari, vi sono due caratteristiche di interesse, nel senso che esse hanno effetti sulla modellazione dei problemi di ottimizzazione. Un primo elemento riguarda i costi di setup, come nel caso di un forno di tempra, che ha bisogno di tempi di setup per riscaldarsi, con conseguenti costi energetici.

Un secondo elemento riguarda il fatto che, per una data lavorazione, possono esservi più macchinari associabili, anche con caratteristiche molto diverse e quindi con differente impatto sui tempi di realizzazione nonché sulla qualità dei prodotti.

Inoltre, soprattutto in relazione ai tavoli di taglio bidimensionale, è possibile modellare i macchinari o in modo atomico, o come costruiti da più moduli. Infatti, come accennato nel secondo Capitolo, questi macchinari constano in genere di un tavolo di taglio automatico, ed uno o più tavoli ausiliari per le operazioni manuali. Nel caso del vetro stratificato, questi tavoli ausiliari possono essere sfruttati per realizzare schemi di taglio con un maggiore livello di nesting dei tagli. D'altro canto, se uno dei tavoli non viene associato ad una lavorazione di taglio, da un lato questa potrà risentirne in termini di qualità delle soluzioni di taglio bidimensionale, e dall'altro il tavolo libero può essere sfruttato per operazioni di taglio completamente manuali, ad esempio su lastre residue di vetro ottenute come sfrido da precedenti lavorazioni di taglio. Sorge cioè da un lato un trade-off fra parallelismo di più lavorazioni e capacità di utilizzo della materia prima, e dall'altro una dipendenza fra la schedulazione delle attività e le istanze di problemi di taglio bidimensionale.

6.3.2 I carrelli

In termini di arnesi, è molto interessante fare una digressione sui *carrelli* (o *cavalletti*) mobili, utili per lo stazionamento e la movimentazione dei vetri all'interno dell'industria, nonchè per le consegne ai clienti.

Le modalità di posizionamento dei vetri sui carrelli dipendono da vari fattori, e principalmente dal tipo di vetro, dalle sue dimensioni geometriche, pezzo per pezzo, e dal tipo di carrello.

I carrelli sono generalmente a forma di T capovolta, così da avere uno schienale, inclinato rispetto alla direzione verticale con una pendenza di almeno il 6%, e due piani di appoggio che formano con i relativi schienali degli angoli ottusi. Ci sono poi i carrelli ad 'L', che hanno un solo piano di appoggio.

Per entrambi i tipi, vi sono estensioni che prevedono la possibilità di aggiungere piani di appoggio ad altezza, a volte regolabile, superiore a quella dei piani di base. Per ciascuna forma poi vi sono dei sottotipi determinati dalle dimensioni e dal materiale di costruzione, che implicano fondamentalmente dei limiti sul peso sopportabile da ciascun piano del carrello e dal carrello nel suo insieme.

Riguardo al limite di insieme, questo non è quasi mai un problema, poiché il vincolo di ingombro geometrico risulta il più delle volte quello più stringente. Le modalità di riempimento dei carrelli devono soddisfare vincoli geometrici che tengano conto tra l'altro dei seguenti aspetti:

a) la base dei carrelli è una griglia e non un piano 'pieno'.
b) i vetri devono appoggiarsi fra loro in modo da essere stabili, ma evitando il rischio che ciascuno possa graffiare gli altri. In particolare i vetri-camera sono più delicati per la presenza della camera d'aria. Ciò impone di non poter appoggiare su di essi altri vetri con qualche vertice che finisca sulla zona centrale del vetro camera.

Un ulteriore tipo di carrelli sono quelli detti "ad arpa", che hanno dei pannelli planari e verticali di appoggio, che dividono lo spazio sul piano di appoggio in tanti piccoli interstizi, entro ciascuno dei quali si può appoggiare un vetro (float o stratificato).

Tale tipo di carrelli ha una logica di appoggio più semplice e non vincolata ad alcuna sequenza con cui inserire i vetri, cosa che invece avviene per i carrelli precedentemente descritti, dove ciascun vetro si appoggia su altri vetri, salvo quelli che si appoggiano direttamente sullo schienale del carrello.

Schematizzando, si può dire che i carrelli ad arpa hanno una modalità di accesso *random* mentre quelli a 'T' o ad 'L' ne prevedono uno *a stack*, pensando però ad una relazione di ordine-appoggio non necessariamente totale, pur essendo tale caso quello con maggiore frequenza. Infatti, in alcuni casi, risulta migliore uno schema di posizionamento "ad albero di fette" dove ciascuna fetta può essere costituita da più vetri affiancati. In figura 6.4 è riportato un esempio, con un carrello ad 'L', con tre "fette" impilate a stack (albero senza ramificazioni), con la prima fetta, appoggiata allo schienale, costituita da due vetri e la terza costituta da tre vetri.

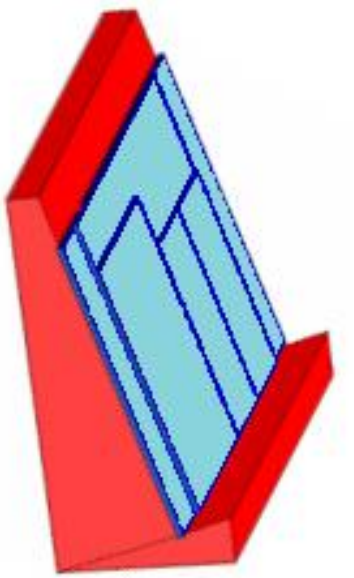


Figura 6.4 Schema di carrello con vetri

La scelta degli insiemi di vetri da inserire sui singoli carrelli può necessitare di un apposito modulo di ottimizzazione. Tale necessità non sussiste per i carrelli ad arpa, che però sono più ingombranti e vengono dunque utilizzati solo in presenza di un elevato numero di misure diverse, non posizionabili sui carrelli ad ‘L’ o ‘T’ in maniera efficace. I carrelli ad arpa sono utili anche come stazioni di appoggio durante le lavorazioni di taglio. In pratica, man mano che i pezzi di vetro vengono estratti dalle lastre, essi vengono stazionati nei carrelli ad arpa, e quando è il loro turno di essere appoggiati sul relativo carrello a ‘T’ o ad ‘L’, vengono rimossi dal carrello ad arpa e posti appunto sull’altro tipo di carrello.

6.4 L'organizzazione delle attività

Un *ordine* di un cliente consiste in uno o più prodotti finiti, ciascuno con le particolari lavorazioni aggiuntive da associare a ciascun nodo del relativo albero di assemblaggio.

Dall'insieme di ordini da espletare deriva un insieme di lavorazioni *effettive L* i cui elementi hanno associati: il tipo di lavorazione, la quantità ed il tipo di semilavorato o prodotto finito su cui insistono. Una lavorazione effettiva può essere vista come l'istanza di un modello, rappresentato dal tipo di lavorazione in quanto tale. Esiste poi una relazione d'ordine parziale ottenuta sulla base di quella definita per i tipi di lavorazione e sulla base dell'albero di assemblaggi per ciascun tipo di prodotto da realizzare.

Infatti, se due semilavorati A e B devono essere assemblati per realizzare il prodotto finito C, gli assemblaggi di A e di B, nonché tutte le lavorazioni aggiuntive su A e su B, devono essere realizzati prima dell'assemblaggio di C, mentre tutte le lavorazioni aggiuntive su C vanno realizzate dopo l'assemblaggio di C. Inoltre, per ciascuno dei tre, A, B e C, le relative lavorazioni aggiuntive devono seguire le regole di dipendenza di cui si è parlato nel paragrafo 6.2.2. In principio tutte queste dipendenze valgono per ogni singolo pezzo di A, di B o di C.

E' opportuno poter dividere un insieme di unità di uguale prodotto finito (relative alla stessa coppia cliente-ordine) in lotti. Tale possibilità serve a rispecchiare la naturale opzione adottata in presenza di un numero elevato di pezzi, quando essi debbano essere sottoposti a più lavorazioni. Se ad esempio 100 pezzi vanno sia arrotati che temprati, può essere opportuno cominciare a temprare i primi 50 pezzi quando i secondi 50 non siano stati ancora arrotati. L'idea è lo stessa delle catene di montaggio, come schematizzato in Figura 6.5.

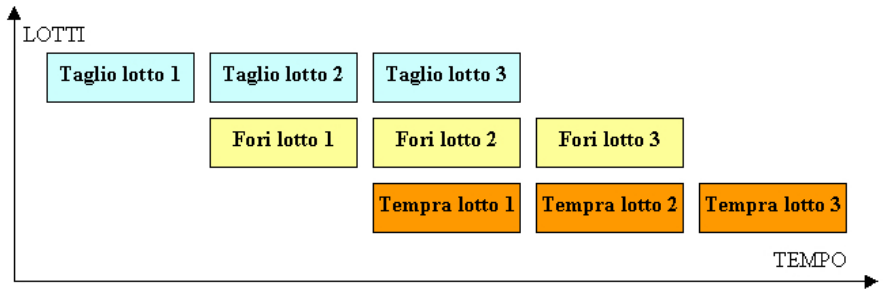


Figura 6.5 Lavorazioni contemporanee

Tale lottizzazione è denominata *lottizzazione di componente* e determina la possibilità di spezzare dipendenze temporali altrimenti da soddisfare, pur non essendo logicamente necessarie. Infatti rende indipendenti gli insiemi di lavorazioni effettive "gemelle" da realizzare su ciascuno dei lotti. Ciascun lotto di componente è considerato *propagante*, nel senso che l'assemblaggio dei pezzi di quel lotto può cominciare quando siano pronti i soli pezzi semilavorati da assemblare, indipendentemente da quelli relativi ad altri lotti. In pratica si induce una lottizzazione sui semilavorati componenti e, per ciascun semilavorato componente, la divisione può essere fatta solo scindendo ulteriormente i lotti indotti. Fissato un lotto di componente, per ogni lavorazione effettiva (di assemblaggio o aggiuntiva) da realizzare su tale lotto, si identifica una coppia *lotto-lavorazione*.

Più coppie *lotto-lavorazione* possono essere agglomerate. E' il caso in cui, riprendendo l'esempio precedente, i primi 50 pezzi possono essere arrotati insieme a pochi altri pezzi di un altro cliente e per i quali la consegna non è imminente (altrimenti conviene ovviamente lavorarli prima e separatamente dai 50). Tali agglomerati costituiscono una *attività lavorativa* (o semplicemente *attività*) atomica, così come vista dagli operatori.

La lottizzazione di lavorazione ha in genere senso soltanto se esistono dei tempi di setup o comunque di passaggio fra un gruppo e l'altro i pezzi, poiché altrimenti costituirebbero in genere solo una complicazione. Il caso più eclatante in cui si può parlare di tempi di setup è per la lavorazione di tempra, per la quale esiste un forno che viene comunque portato a temperatura indipendentemente dal numero di pezzi presenti nel forno.

Va anche detto che l'agglomerazione in genere può avvenire solo se il tipo di lavorazione è lo stesso. In casi specifici deve essere lo stesso anche il tipo di prodotto, come per la lavorazione di taglio, dato che un'attività di taglio è realizzata su una lastra di un fissato tipo di vetro. In altri casi ancora, meno restrittivi, il tipo di prodotto deve ricadere in una stessa categoria, come nel caso della lavorazione di tempra in cui conta lo spessore dei vetri.

La condizione di avere lo stesso tipo di lavorazione viene meno in casi particolari, come conseguenza delle capacità dei macchinari. In particolare si fa riferimento alle lavorazioni di taglio float e di sagomatura, che possono essere realizzati in maniera combinata se il macchinario in dotazione è capace di operare incisioni curvilinee e nella stessa fase di quelle rettilinee, che servono ad estrarre i pezzi rettangolari finiti e quelli in cui sono racchiuse le sagome.

Un altro caso riguarda l'insieme delle lavorazioni: *foratura, incisione* e *tacche* per le quali esiste un unico macchinario capace, e con grosso guadagno di tempo, di realizzarle contemporaneamente. Esso infatti è dotato di uno strumento CAD con cui disegnare le parti del vetro da consumare, così da ottenere i fori circolari, quelli rettangolari per le tacche e le incisioni.

Esiste un caso in cui l'agglomerazione di diverse coppie lotto-lavorazione ha degli effetti non esigui. E' il caso delle lavorazioni di taglio, poiché gruppi con un maggior numero di pezzi possono giovare all'ottimizzazione dei piani di taglio, determinando cioè un migliore utilizzo della materia prima.

Una volta definite le attività, le dipendenze fra i lotti di lavorazione si traducono in dipendenze sulle attività. Infatti se la lavorazione effettiva A deve precedere la lavorazione effettiva B, allora una qualunque attività che comprende uno dei lotti di A deve precedere l'attività che comprende l'equivalente lotto della lavorazione B. La Figura 6.6 schematizza un esempio con cinque lotti-lavorazione agglomerati in tre attività. Le tre frecce marcate rappresentano altrettante dipendenze fra lotti-lavorazione, che dunque diventano dipendenze fra le attività che li agglomerano. In principio, esiste il rischio che un insieme di agglomerazioni possano determinare delle dipendenze cicliche fra le attività, e ciò va evitato.

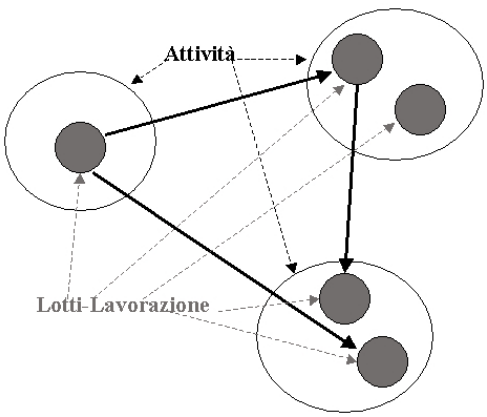


Figura 6.6 Agglomerazione di lotti-lavorazione

A valle delle fasi di scissione in lotti e di agglomerazione in attività, vi è una terza fase di eventuale scissione di ogni attività in differenti finestre temporali, cosa che può essere utile per diversi motivi, come ad esempio per le capacità di un forno di tempra, che può contenere un limitato numero di pezzi per ogni infornata.

6.5 I problemi di ottimizzazione

Sulla base degli elementi descrittivi dati in questo Capitolo, in una industria del vetro emergono diversi problemi di ottimizzazioni, fra di loro interdipendenti, nel senso che le condizioni al contorno per ciascuno di essi dipendono dalle soluzioni scelte per gli altri. In particolare, i problemi principali sono sei, elencati e brevemente descritti nel seguito.

- *Scheduling*
- *Assignment*
- *Cutting Stock* mono-dimensionale e bidimensionale
- *Packing* tridimensionale
- *Sequencing*
- *Routing*

Innanzitutto vi è la schedulazione delle attività (*Scheduling*), supponendo di inglobare nel problema sia le scelte circa le divisioni in lotti e le agglomerazioni in attività, sia la schedulazione vera e propria delle attività, che tenga conto dei vincoli di utilizzo dei macchinari in una sola attività per volta.

Vi è poi la fase di assegnazione delle risorse umane alle attività, per cui si profila un problema di *Assignement*.

Per le lavorazioni di taglio, è necessario generare gli schemi di taglio risolvendo un problemi di *Cutting*, sia mono-dimensionali, per le canaline, sia bidimensionali per i vetri.

A valle del taglio, come già spiegato in precedenza, i vetri vengono appoggiati su appositi carrelli. In genere su ogni carrello vengono appoggiati pezzi relativi alla stessa tripla ordine-cliente-prodotto, in modo che tali pezzi debbano seguire lo stesso iter produttivo, e si possa sfruttare la stessa modalità di appoggio scelta a valle del taglio, durante tutto l'iter.

Si sottolinea che i vetri di un gruppo associato ad una tripla possono essere in numero tale non poter essere appoggiati tutti sullo stesso carrello, necessitando cioè di una suddivisione fra più carrelli. In generale la suddivisione avviene sulla base della sequenza con cui i vetri vengono estratti, con particolare riferimento al caso in cui i vetri del sottogruppo associato ad un carrello siano sparsi sulle diverse lastre relative all'attività di taglio. E' dunque opportuno risolvere il corrispondente problema di *Sequencing*, che consiste nella scelta della sequenza con cui lavorare le lastre.

Infine, quando si consegnano i vetri ai clienti, emerge da un lato un ulteriore problema di Packing e dall'altro un problema di *Routing*. Il Packing è collegato all'esigenza di ridurre la probabilità di danni durante il tragitto, mentre il problema di Routing consiste nell'ottimizzazione dei percorsi sia relativamente ai tempi di percorrenza sia relativamente ad obiettivi aggiuntivi, quale può essere la necessità di recupero di carrelli in comodato dai clienti che hanno precedentemente ricevuto altre consegne.

La Figura 6.7 schematizza i suddetti problemi, rappresentando inoltre le dipendenze esistenti. Le dipendenze più forti sono fra i quattro problemi di Scheduling, di Cutting, di Packing e di Sequencing.

Infatti, le agglomerazioni in attività decise nella schedulazione determinano la generazione delle istanze di problemi di Cutting, in quanto ogni istanza corrisponde ad una attività di taglio. D'altronde, se vengono generati schemi di taglio bidimensionale che estraggono pezzi con dimensioni aumentate, per i quali è quindi necessaria l'arrotatura, come spiegato nel secondo Capitolo, allora bisogna aggiungere nuove attività. Inoltre una nuova attività di taglio può essere dovuta al danneggiamento dei vetri, che dunque devono essere nuovamente estratti da altre lastre. La probabilità con cui ciò può avvenire è correlata, come spiegato nel secondo Capitolo, a quanto i parametri legati al rischio di *finte* siano stati utilizzati in modo stressato, magari per ottenere schemi di taglio migliori.

Altre considerazioni peculiari circa la dipendenza fra Cutting e Scheduling sono quelle già riportate nel paragrafo 6.3.1 circa le modalità di modellazione dei macchinari di taglio.

Anche il problema del Packing è influenzato dallo Scheduling, poiché se lotti diversi hanno pochi vetri associati, è auspicabile che su di un carrello vengano posizionati vetri di lotti diversi, purché debbano successivamente subire la stessa lavorazione. La dipendenza fra Cutting e Packing è allora determinata dal fatto che, come detto prima, la necessità di attività di arrotatura inizialmente non previste sono dovute alle soluzioni scelte per il Cutting.

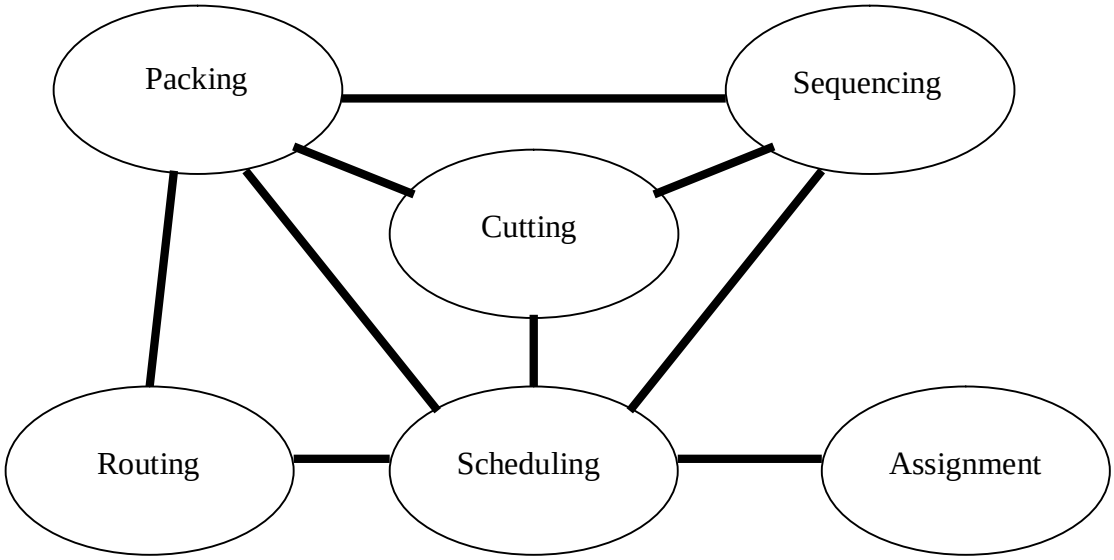


Figura 6.7 Problemi di ottimizzazione

Inoltre, i tempi di inizio delle ulteriori lavorazioni cui devono essere sottoposti i pezzi di vetro estratti dal taglio influenzano le priorità con cui estrarre i gruppi di vetri associati a lotti diversi. Questo ha un impatto sul problema di Sequencing, in quanto, supponendo fissati gli schemi di taglio generati dal cutting, può essere opportuno ottimizzarne la sequenza di esecuzione, sulla base delle priorità con cui estrarre i diversi gruppi di pezzi. La Figura 6.8 schematizza un esempio con almeno due lastre necessarie per il taglio, in cui i pezzi di ciascun lotto sono identificati da un colore diverso.

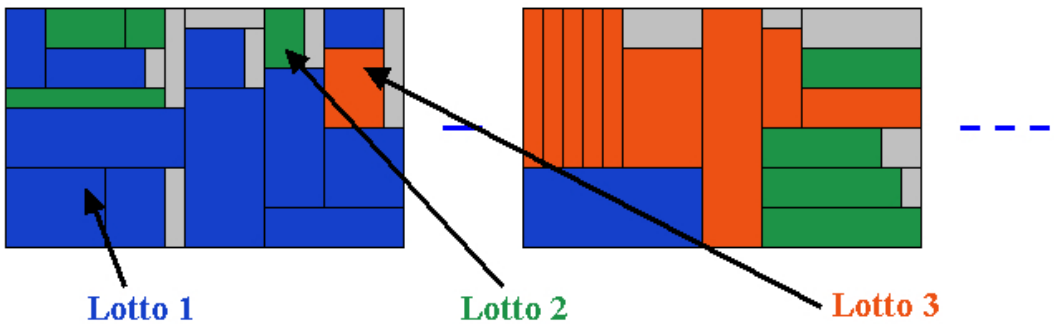


Figura 6.8 Lotti con diversa priorità

Queste considerazioni spiegano anche la stretta dipendenza fra Cutting e Sequencing. Infatti può essere opportuno ottimizzare in modo combinato gli schemi di taglio e la loro sequenza, dando maggiore importanza all'uno o all'altro sulla base della relazione che esiste fra i costi della materia prima, rispetto ai quali conviene dare importanza al Cutting, ed i costi

diretti ed indiretti derivanti dalla qualità delle sequenze di estrazione dei pezzi, rispetto ai quali conviene dare importanza al Sequencing.

La dipendenze del problema di Packing rispetto al Cutting ed al Sequencing sono basate sulle seguenti considerazioni. Si supponga che il gruppo di vetri associato ad un lotto sia tale da necessitare di più carrelli. In tal caso va scelto quale sottogruppo di vetri associare ad ogni carrello. Se tale scelta è fatta solo secondo i vincoli di appoggio, cioè risolvendo il problema del Packing, allora il Sequencing deve tener conto dei singoli sottogruppi piuttosto che dei gruppi. La motivazione è che un elevato numero di carrelli semi-pieni nei pressi dell'isola di lavoro può determinare difficoltà di movimentazione degli operatori, come schematizzato in Figura 6.9, con conseguente aumento sia dei tempi di realizzazione dell'attività sia delle probabilità di danneggiare i vetri. Si ricordi infine le utili considerazioni fatte nel paragrafo 6.3.2 sull'utilizzo dei carrelli ad arpa relativamente a questo tipo di problematiche.

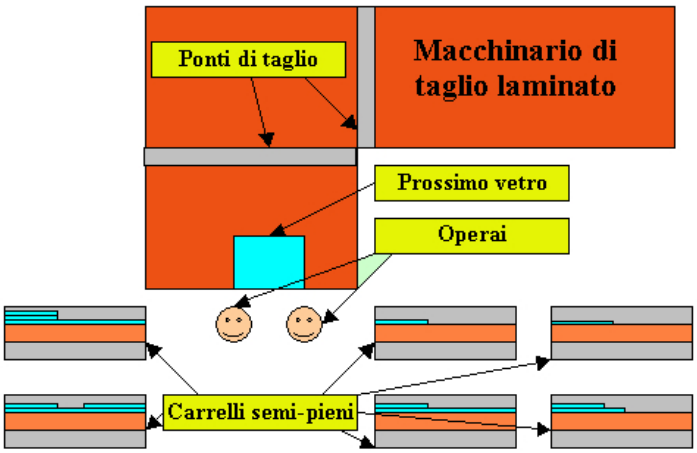


Figura 6.9 Ingombro dovuto ai carrelli semi-pieni

D'altro canto, se il sequencing tiene conto solo dei gruppi di vetri, allora il problema del Packing può essere risolto sulla base della sequenza con cui vengono estratti i vetri all'interno di ogni gruppo. Ciò vuol dire che in tal caso la sequenza di risoluzione dei problemi è inversa, cioè prima Sequencing e poi Packing.

Il legame fra Scheduling ed Assignment è evidente poiché l'assegnazione delle risorse umane alle attività presuppone di partire dalla conoscenza di quali siano le attività, e ciò è deciso risolvendo il problema dello Scheduling.

La relazione fra Scheduling e Routing è legata al fatto che l'ottimizzazione del percorso di un mezzo di trasporto può cambiare se si aggiunge o si rimuove una tappa, sulla base dei tempi di realizzazione dei prodotti finiti per un dato cliente.

Infine, il legame fra Packing e Routing è legato al fatto che le probabilità di danneggiamento dei vetri quando siano appoggiati su un carrello, posto in un mezzo di trasporto, dipendono anche dal tempo di percorrenza fino al luogo destinazione, nonché dal tipo di tragitto. Tali probabilità hanno impatto sui vincoli del problema di Packing.

In relazione al problema di Cutting, oggetto di questo lavoro, è dunque evidente la stretta correlazione che esiste con diversi altri problemi di ottimizzazione. In generale, come già accennato in precedenza, l'importanza da dare a ciascun aspetto è legata alle voci di costo associate. In casi estremi, con vetri che costano poco o che costano molto, la sequenza di risoluzione dei problemi può vedere il Cutting rispettivamente come ultimo nel caso di basso costo e come primo nel caso di costo elevato. Il costo, a sua volta, va considerato alterato anche in funzione della disponibilità del relativo tipo di materiale. Ad esempio, se un materiale, pur costando poco, è momentaneamente disponibile in bassa quantità, e da esso

debbono essere estratti dei pezzi che un cliente considera urgenti, allora è opportuno ridurre al minimo le probabilità di danneggiamento, onde evitare di dover attendere i tempi di consegna dei fornitori.

Più in generale è opportuno ottimizzare i diversi problemi in modo combinato, quanto meno con fasi di feedback a valle della soluzione di ciascun problema, per affinare le soluzioni associate ai problemi precedentemente risolti. Nei successivi due Capitoli, viene trattato il problema del Sequencing, prima in termini assoluti, e poi in combinazione con il problema del Cutting.

Capitolo 7

Il Sequencing

Il Capitolo introduce i problemi di Sequencing, che differiscono fra loro per la funzione obiettivo. Essi sono noti anche come *Pattern Sequencing Problems*, o come *Pattern Allocation Problems*. Ne viene data una descrizione indipendente dai problemi di Cutting Stock, sebbene lo studio di tali problemi nasca, salvo qualche eccezione, proprio per determinare la sequenza degli schemi di taglio generati come soluzione di un problema di Cutting associato a più stocks. In pratica si suppone di avere a disposizione gli schemi di taglio, già generati e non alterabili.

Il primo paragrafo descrive un modello comune per i problemi di Sequencing, mostrando anche alcune delle funzioni obiettivo più utilizzate, introducendo così i problemi MOSP, MORP ed MDP. Il secondo paragrafo spiega invece alcuni dei metodi risolutivi proposti in letteratura, descrivendo alcuni metodi utilizzati per il MORP per poi passare, con maggior dettaglio, a quelli proposti per il MOSP, trattato dalla maggioranza degli inerenti lavori di letteratura. Viene pure descritto un modello lineare intero tratto dall'equivalenza con un altro problema simile, quello della minimizzazione del numero di cambi utensile, o MTSP.

7.1 I problemi di Sequencing

Il problema del Sequencing viene più propriamente indicato con il nome di *Pattern Sequencing Problem* (Fink e Voβ, 1999), poiché consiste nello scegliere una permutazione ottima di *m* pattern rispetto ad una data funzione obiettivo, in relazione a quelli che, nel caso del Cutting Stock, sono *n* gruppi di pezzi.

I problemi di Sequencing sono anche noti con il nome di *Pattern Allocation Problem*. Tale nome è stato inizialmente utilizzato in lavori come quelli di Dyson e Gregory (1974), di Madsen (1988) e di Yuen e Richardson (1995).

La sua definizione, in termini generali, richiede pochi elementi. Data una matrice *C*, di dimensioni $m \times n$, con valori interi c_{ij}, per $1 \le i \le m$ ed $1 \le j \le n$, ogni riga corrisponde ad un pattern ed ogni colonna ad un gruppo di pezzi. L'obiettivo consiste nella scelta di una permutazione $\Pi = (\pi_1, \ldots, \pi_m)$ delle righe, al fine di ottimizzazione di una data funzione obiettivo. L'*i*-mo elemento π_i del vettore Π identifica l'indice della riga scelta in *i*-ma posizione per cui, a valle della permutazione, la matrice *C* passa da $(c_1, \ldots, c_m)$ a $(c_{\pi_1}, \ldots, c_{\pi_m})$.

Il problema si specializza a seconda della funzione obiettivo scelta. La sua definizione generale somiglia a quella del ben noto problema del commesso viaggiatore (*Traveling Salesman Problem* o TSP, Applegate, Bixby, Chvàtal e Cook, 2007), ma la differenza consiste nel fatto che, mentre nel TSP gli elementi che entrano nella funzione obiettivo sono funzione solo delle coppie di elementi consecutivi (π_i, π_{i+1}) del vettore Π, nel caso del Sequencing non è sempre così, come sarà più chiaro a breve.

Da quanto detto, si può affermare che il Sequencing racchiude una classe di problemi, ciascuno identificato dalla particolare funzione obiettivo utilizzata. Le funzioni obiettivo più utilizzate sono tre, tutte significative dal punto di vista dei problemi di Cutting Stock, sulla base delle relazioni descritte nel Capitolo precedente con gli altri problemi di ottimizzazione. Prima di elencarle, si sottolinea infatti che il problema è spesso referenziato come Sequencing Cutting Patterns problem (Foerster e Wäscher, 1998; Pezzella e De Giovanni, 2007) o come Cutting Sequencing problem (Faggioli e Bentivoglio, 1998).

Yanasse (1997b) e Linhares e Yanasse (2002) descrivono le suddette tre funzioni obiettivo, qui introdotte utilizzando termini relativi al caso in cui i pattern siano dei cutting pattern. Comunque, nel seguito, per brevità, si utilizzerà il termine pattern. Come detto, in relazione al Cutting Stock, il valore *n* rappresenta il numero di gruppi di pezzi, ciascuno ad esempio destinato ad un diverso carrello, come spiegato nel precedente Capitolo. Il concetto di carrello è espresso in letteratura con il termine *stack*.

Ciascun elemento c_{ij} della matrice *C* rappresenta il numero di pezzi del *j*-mo gruppo presenti nell'*i*-mo pattern. Per una data permutazione $\Pi = (\pi(1), \ldots, \pi(m))$ dei pattern, si dice che il *j*-mo stack è aperto a valle del completamento dei primi *k* pattern, se il prodotto $\left(\sum_{i=1}^{k} c_{\pi(i),j}\right) \cdot \left(\sum_{i=k}^{m} c_{\pi(i),j}\right)$ è positivo (Fink e Voβ, 1999). Infatti, se ciò avviene, allora vuol dire che almeno un pezzo del *j*-mo gruppo è stato estratto da uno dei primi *k* pattern, come si evince dalla positività del primo fattore, e che almeno un pezzo di tale gruppo è stato estratto proprio dal *k*-mo pattern o sarà estratto in uno dei pattern successivi, come si evince dalla positività del secondo fattore.

Dunque il *j*-mo stack viene aperto prima di processare il pattern di indice $p^{\Pi}_{\min,j} = \min_{i=1,\ldots,m} \{i : c_{\pi(i),j} > 0\}$, cioè prima di processare il primo pattern che contiene pezzi del *j*-mo gruppo. Esso viene chiuso dopo aver processato il pattern di indice

$p^{\Pi}_{\max,j} = \max_{i=1,\ldots,m} \{i : c_{\pi(i),j} > 0\}$, cioè dopo aver processato l'ultimo pattern che contiene pezzi del *j*-mo gruppo.

Si supponga, come nell'esempio riportato in Yanasse (1997b), di avere $n = 6$ gruppi ed $m = 6$ pattern, e di identificare i gruppi con le lettere *a*, *b*, *c*, *d*, *e*, *f*. La Tabella 7.1 riporta i dati relativi all'esempio.

Pattern	**Pezzi**
1	*a, a, a, b, d, d*
2	*b, d, e, e, f*
3	*c, c, d, d, d, d*
4	*a, c, c, e*
5	*a, a, f, f*
6	*e, e, e, f, f*

Tabella 7.1 Un esempio di Sequencing

Dunque il primo pattern contiene tre pezzi del gruppo *a*, uno del gruppo *b* e due del gruppo *d*, il secondo pattern contiene un pezzo del gruppo *b*, un pezzo del gruppo *d*, due pezzi del gruppo *e* ed un pezzo del gruppo *f*, e così via.

Se si suppone di processare i pattern nello stesso ordine con cui sono stati indicati, cioè con la permutazione $\Pi = (1, 2, 3, 4, 5, 6)$, allora dopo aver processato il primo pattern, risultano aperti gli stack relativi ai gruppi *a*, *b* e *d*. Dopo il secondo pattern sono aperti gli stack relativi ai gruppi *a*, *b*, *d*, *e* ed *f*, e così via, come indicato in Tabella 7.2.

Pattern	**Stack aperti**
1	*a, b, d*
2	*b, d, d, e, f*
3	*a, c, d, e, f*
4	*a, c, e, f*
5	*a, e, f*
6	*e, f*

Tabella 7.2 Stack aperti

Si può notare come, dopo aver processato sia il secondo che il terzo pattern, il numero di stack aperti sia massimo e pari a 5. Una prima versione del problema del Sequencing consiste nel trovare la permutazione che minimizzi il massimo numero di stack aperti. Tale problema viene infatti denotato come *Minimization of Open Stack Problem*, o MOSP.

In termini applicativi, si può notare che tale obiettivo corrisponde a ridurre l'ingombro dei carrelli aperti presso l'isola di lavoro, come descritto nel Capitolo precedente.

Formalmente (Linhares e Yanasse, 2002), si può definire la matrice Q^{Π}, di dimensioni $m \times n$, pari cioè a quelle della matrice *C*, ed i cui elementi q^{Π}_{ij} sono definiti, in funzione della permutazione Π, dalla relazione

$$q^{\Pi}_{ij} = \begin{cases} 1 & \text{if } \exists h,k : h \le i \le k \text{ and } c_{\pi(h),j} \cdot c_{\pi(k),j} > 0 \\ 0 & \text{else} \end{cases}$$

cone la seguente semantica: se $q^{\Pi}_{ij} = 1$, allora il *j*-mo stack è ancora aperto dopo aver processato il pattern in posizione *i*-ma nell'ordine stabilito dalla permutazione.

Ne consegue che il massimo numero di stack aperti $Z_{MOSP}(\Pi)$ è pari a $\max_{1\le i\le m}\sum_{j=1}^{n} q_{ij}^{\Pi}$. Il problema dunque può essere formalizzato nel seguente modo:

$$MOSP\begin{cases}\min \quad Z(\Pi)=\max\limits_{1\le i\le m}\sum\limits_{j=1}^{n} q_{ij}^{\Pi} \\ \text{over all permutations } \Pi\end{cases}$$

Fink e Voβ (1999) referenziano questo problema con la sigla PSP-SOS, dove PSP sta per Pattern Sequencing Problem e SOS, che identifica la specifica funzione obiettivo, sta per Simultaneously Open Stacks.

Una seconda funzione obiettivo, molto simile, è relativa al numero di pattern che intercorrono dal primo all'ultimo pattern da cui siano estratti pezzi per un determinato gruppo. Tale valore è chiamato *Order Spread* per quel gruppo. L'obiettivo, sotto questo aspetto, consiste nella minimizzazione dell'order spread medio al variare dei gruppi. Tale problema è identificato dalla sigla MORP.

Per una data permutazione Π dei pattern, e supponendo che per il generico *i*-mo pattern, vi siano α_i "copie" da processare, il valore di funzione obiettivo $Z_{MORP}(\Pi)$ associato può essere ottenuto con l'espressione $\frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=p_{\max,j}^{\Pi}}^{p_{\min,j}^{\Pi}} \alpha_{\pi_i} - 1\right)$, dove le quantità $p_{\min,j}^{\Pi}$ e $p_{\max,j}^{\Pi}$ sono state definite sopra.

Il generico valore α_i prende il nome di "frequenza" dell'*i*-mo pattern. Se le frequenze sono tutte pari ad 1, l'espressione di $Z_{MORP}(\Pi)$ si semplifica e diventa pari alla sommatoria $\sum_{j=1}^{n}\left(p_{\max,j}^{\Pi} - p_{\min,j}^{\Pi}\right)$, dove il fattore costante 1/*n* è stato rimosso per semplicità.

Riprendendo l'esempio della Tabella 7.1, e supponendo i valori α_i tutti pari ad 1, per il gruppo *a* l'order spread vale 4 poiché il primo pattern da cui sono estratti pezzi di tale gruppo è il primo della sequenza e l'ultimo analogo pattern è il quinto della sequenza. Per il gruppo *b* l'order spread vale 1 e così via, come riportato in Tabella 7.3.

Gruppi	Order Spread
a	5 – 1 = 3
b	2 – 1 = 1
c	4 – 3 = 1
d	3 – 1 = 2
e	6 – 2 = 4
f	6 – 2 = 4

Tabella 7.3 Spread degli ordini

In Fink e Voβ (1999), il MORP è chiamato PSP-AOS, dove AOS sta per Average Order Spread. Foerster e Wäscher (1998) lo chiamano OSMP, che sta per *Order Spread Minimization Problem*. Ancora, in Respicio e Captivo (2005), è invece chiamato AOR.

Una generalizzazione di questo problema consiste nel sostituire la media pura degli order spread con una media pesata. Questa generalizzazione trova applicazione nella minimizzazione del costo da spendere per gli attori di un film. In tal caso ogni pattern è

associabile ad una scena ed ogni gruppo è costituito da un solo attore (Cheng, Diamond e Lin, 1993; Nordström e Tufekci, 1994; Fink e Voβ, 1999). Fink e Voβ referenziano tale problema con l'acronimo PSP-AC, dove AC sta per Actor Cost. Un nome più opportuno potrebbe essere WMORP, che sta per *Weighted* MORP, in analogia, tra l'altro, alle versioni del problema di Cutting Stock, ma sembra che nessuno ancora abbia utilizzato questo acronimo.

Se γ_j rappresenta il costo associato al *j*-mo gruppo, allora il valore di funzione obiettivo $Z_{WMORP}(\Pi)$ vale $\sum_{j=1}^{n}\left(\gamma_j \cdot \sum_{i=p^{\Pi}_{\max,j}}^{p^{\Pi}_{\min,j}} \alpha_{\pi_i}\right)$.

In termini delle correlazioni con il Cutting degli altri problemi in un ambito industriale, il problema MORP è da associare alla priorità con cui determinati gruppi di pezzi debbono essere estratti. Un basso order spread medio vuol infatti dire che è possibile estrarre quanto prima il gruppo di pezzi a maggiore priorità, perché presenti su pattern vicini nella sequenza stabilita attraverso la permutazione, poi quello di pezzi con priorità immediatamente inferiore e così via.

Un terzo tipo di funzione obiettivo è legato al concetto di *discontinuità*. Si dice che si ha una discontinuità per il *j*-mo gruppo nel passare dal pattern $\pi(i)$ al pattern $\pi(i+1)$ quando almeno un pezzo di quel gruppo è estratto dal pattern $\pi(i)$, ma nessuno dal pattern $\pi(i+1)$. Il numero di discontinuità fra i pattern $\pi(i)$ e $\pi(i+1)$ è pari al numero di gruppi per i quali vi è la discontinuità. Formalmente, si può definire per ogni gruppo *j* una matrice D^j, di dimensioni $m \times m$, con elementi $d^j{}_{h,k}$ così definiti

$$d^j_{h,k} = \begin{cases} 1 & \text{if } \ c_{h,j} > 0 \text{ and } c_{k,j} = 0 \\ 0 & \text{else} \end{cases}$$

Ne scaturisce la definizione di una matrice *D*, di dimensioni $m \times m$, con elementi $d_{h,k} = \sum_{j=1}^{n} d^j_{h,k}$, ciascuno dei quali rappresenta il numero di discontinuità fra i pattern con i corrispondenti indici *h* e *k*, nell'opportuno ordine reciproco. Il valore di funzione obiettivo associato ad una fissata permutazione Π è $Z_{MDP}(\Pi) = \sum_{i=1}^{m+1} d_{\pi(i),\pi(i+1)}$, dove si considera un pattern fittizio aggiuntivo, privo di pezzi, ed imposto in $(m+1)$-ma posizione nella sequenza.

Il problema che ne scaturisce, noto come MDP, che sta per *Minization of Discontinuities Problem*, consiste nel trovare la permutazione per la quale sia minimo tale valore. Per tale problema, la funzione obiettivo dipende solo da termini di costo associati a pattern consecutivi nella sequenza. Dunque il problema diventa equivalente al problema del commesso viaggiatore. Madsen (1988) sfrutta la seguente definizione complementare

$$d^{*j}_{h,k} = \begin{cases} 1 & \text{if } \ c_{h,j} > 0 \text{ and } c_{k,j} > 0 \\ 0 & \text{else} \end{cases}$$

di elementi che valgono 1 se il gruppo *j* ha pezzi sia sul pattern *h* che sul pattern *k*. Ne scaturisce la definizione di una matrice D^*, con elementi $d^*_{h,k} = d^*_{h,k} = -\sum_{j=1}^{n} d^{*j}_{h,k}$, cioè

simmetrica, al contrario di quanto in generale non avviene per *D*, salvo nel caso in cui ogni pattern ha pezzi dello stesso numero di gruppi. La minimizzazione del suddetto valore $Z_{MDP}(\Pi)$ equivale alla minimizzazione del valore $Z^*_{MDP}(\Pi) = \sum_{i=1}^{m} d^*_{\pi(i),\pi(i+1)}$. Per lavorare con valori positivi, basta aumentare di una opportuna costante tutti gli elementi della nuova matrice D^*.

Esistono altri tipi di funzione obiettivo, correlati alle specifiche applicazioni, come ad esempio in biologia molecolare per sequenze di DNA (Alizadeh, Karp, Newberg e Weisser, 1995) o, meno recentemente, per l'ordinamento di elementi di "informazione" (Deutsch e Martin, 1971; Morse, 1972; McCormick, Schweitzer e White, 1972).

Si può notare come i tre problemi, MOSP, MORP ed MDP siano molto simili fra loro, ed infatti, in generale, buone soluzioni per uno di essi sono altrettanto buone per gli altri due.

In Yanasse (1997b) si era formulata la congettura che, per una qualunque coppia di questi tre problemi, esistesse sempre una soluzione contemporaneamente ottima per entrambi. Tuttavia il lavoro di Linhares e Yanasse (2002) ha dimostrato che ciò è falso per ciascuna delle tre possibili coppie, attraverso le due seguenti matrici C_1 e C_2. La matrice C_1 rappresenta una istanza per la quale la congettura risulta falsa per la coppia di problemi MOSP e MORP, mentre la matrice C_2 rappresenta una istanza per la quale la congettura risulta falsa sia per la coppia MOSP ed MDP sia per la coppia MORP ed MDP.

$$C_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Relativamente alla complessità di questi problemi, Yanasse (1997b) e Linhares e Yanasse (2002) mostrano che essi sono tutti NP-hard. Per il problema MOSP, ciò viene fatto discendere dall'equivalenza con il *Gate Matrix Layout Problem*, o GMLP (Möhring, 1990). Già Andreatta, Basso, Caumo e Desserti (1989) avevano già mostrato che il problema MOSP è NP-hard, per l'equivalenza con il min-cutwidth problem su ipergrafi, utilizzando il risultato di Gavril (1977). Altri problemi, equivalenti al MOSP, e studiati nell'ambito del progetto di circuiti VLSI o nell'ambito della teoria dei grafi, sono elencati in Linhares e Yanasse (2002). Per il MORP, è mostrato che esso è un caso speciale del *bandwidth minimization problem* (Garey e Johnson, 1979). Tale caratteristica è stata inizialmente sfruttata in letteratura in lavori come quelli di Madsen (1979, 1980).

L'MDP è NP-hard per l'equivalenza con il TSP. Linhares e Yanasse (2002) estendono ai problemi MOSP ed MDP una proprietà mostrata in Deo, Krishnamoorty e Langston (1987) per il GMLP, secondo cui non esiste per tale problema un algoritmo di approssimazione assoluta a meno che non valga l'uguaglianza **P** = **NP**. La prova non è applicabile al caso del MORP.

Le tre funzioni obiettivo elencate non esauriscono l'insieme di quelle studiate in letteratura, come indicato, ad esempio, in Pezzella e De Giovanni (2007).

7.2 Metodi risolutivi

Si descrivono nel seguito alcuni dei metodi risolutivi proposti per i problemi MORP e MOSP. Per il problema MDP non si riporta alcun particolare lavoro, data l'equivalenza rispetto al problema del commesso viaggiatore (TSP). Oltretutto le istanze di TSP corrispondenti alle istanze dell'MDP non hanno una struttura particolare, per cui nessun algoritmo, tanto euristico quanto esatto, è stato appositamente sviluppato in letteratura. Del resto le istanze di TSP derivanti da istanze di MDP non hanno neanche, in generale, matrici di costo-distanza di tipo euclideo. Qui ci si limita nel dire che il lavoro di Dyson e Gregory (1974) sfrutta, per la risoluzione dell'MDP, l'algoritmo proposto per il TSP da Little, Murty, Sweeney e Karel (1963), mentre Madsen (1988) sfrutta la tecnica euristica 3-opt di Lin (1965). In realtà Madsen risolve l'MDP per ottenere una soluzione approssimata per il problema MORP.

Il primo paragrafo descrive l'approccio risolutivo per il MORP di Foerster e Wäscher (1998), che consiste in un approccio meta-euristico, e precisamente con una tecnica di Simulated Annealing (Aarts e Korst, 1989; Dowsland, 1993; Aarts e Lenstra, 1997).

In seguito, sono spiegate tecniche risolutive per il problema MOSP, sia esatte che euristiche. Tale problema sembra essere quello più affrontato e studiato in letteratura (De Giovanni, Pezzella, Pfetsch, Rinaldi e Ventura, 2007). Si descrive inoltre il modello proposto per il MOSP da Yanasse (1997b). Altri modelli, qui non riportati, sono stati proposti in Faggioli e Bentivoglio (1998) ed in Madsen (1988), che si basa sulla definizione della matrice D^* descritta nel paragrafo precedente. Esistono inoltre altri tipi di approcci, anche recenti, ad esempio basati su algoritmi genetici (Pezzella e De Giovanni, 2007) o su programmazione dinamica (De la Banda e Stuckey, 2007). Recentemente è stato anche proposto un approccio bi-obiettivo per MOSP e MORP (Respicio e Captivo, 2005).

7.2.1 Un approccio Simulated Annealing per il MORP

In relazione a quanto detto sopra, Foerster e Wäscher (1988) hanno proposto di utilizzare una tecnica meta-euristica per il fatto che l'euristica 3-opt utilizzata da Madsen rischia, per la sua natura, di fermare la ricerca in minimi locali. La tecnica di Simulated Annealing, come altre tecniche meta-euristiche, accettano invece di spostarsi da una soluzione *a* ad una soluzione *b*, anche se la soluzione *b* è peggiore della soluzione *a*, realizzando lo spostamento (o *mossa*), nel caso della Simulated Annealing, con una probabilità che si riduce al ridursi della qualità della generica soluzione *b* in relazione a quella della soluzione *a* da cui ci si sposta. La struttura dell'algoritmo utilizzato è del tipo classico per un approccio Simulated Annealing ed è riportata nel Box 7.1, come data dagli autori del lavoro.

Selezione di una soluzione iniziale Π_0 e di una temperatura iniziale T
Inizializzazione del valore intero *rep*
Repeat
    For *rc* = 1 **To** *rep* **Do**
    Begin
        Selezione randomatica di una soluzione Π_1 "vicina" a Π_0
        Valutazione della differenza $\delta = v(\Pi_1) - v(\Pi_0)$
        If $\delta < 0$ **Then** $\Pi_0 = \Pi_1$
        Else If ***random***(0, 1) < exp($-\delta / T$) **Then** $\Pi_0 = \Pi_1$
    End
    Riduci la temperatura T
    Aumenta il valore di *rep*
Until Criterio di stop

Box 7.1 Schema dell'approccio Simulated Annealing per il MORP

In particolare, la soluzione iniziale Π_0, viene scelta in modo randomatico. Il valore iniziale della temperatura T viene calcolato attraverso il prodotto $\gamma \cdot v(\Pi_0)$, dove $v(\Pi_0)$ è i valore associato alla soluzione $v(\Pi_0)$. Il valore del parametro γ, scelto positivo, rappresenta un fattore di scala per l'esponente $-\delta / T$ utilizzato più avanti nell'algoritmo.

Il valore iniziale di *rep* è posto pari al numero *m* di pattern del problema. Per ogni ciclo principale, la temperatura è ridotta moltiplicando il suo valore per un fattore α compreso strettamente fra 0 ed 1, mentre il valore del parametro *rep* è aumentato moltiplicandolo per un fattore β strettamente maggiore di 1. Il criterio di stop utilizzato consiste nel fermarsi quando, per una data temperatura, la soluzione corrente non viene aggiornata in nessuno dei tentativi, fatti in numero di *rep*. Per la selezione randomatica della soluzione Π_1, si utilizzano solo mosse 2-opt. In pratica, selezionati in modo random due pattern, la loro eliminazione genera tre corrispondenti sottosequenze in cui viene spezzata la sequenza associata alla corrente soluzione Π_0. Si costruisce poi la sequenza che mantiene la sottosequenza centrale ed inverte la posizione delle altre due sottosequenze, ponendo cioè la prima come terza e la terza come prima. Ovviamente, se viene selezionato il primo o l'ultimo pattern, le sole due sottosequenze che ne scaturiscono, vengono invertite.

7.2.2 Un modello per il MOSP derivato dal problema MTSP

Il lavoro di Yanasse (1997b) presenta nei seguenti termini l'analogia modellistica fra il MOSP ed il *Minimizing the number of Tool Switches Problem* (o MTSP), studiato da Tang e Dendardo (1988).

Il problema MTSP consiste nel minimizzare il numero di cambi utensile su un macchinario flessibile, in grado cioè di realizzare differenti lavorazioni, ciascuna delle quali ha però bisogno di utensili diversi. Formalmente, i dati di ingresso sono:

m	il numero di jobs da processare
n	numero totale di utensili
$\mathbf{A}_i$	un n-vettore di 0 ed 1. $A_{ij} = 1$ a se il job i richiede l'utensile j
t	l'istante di tempo dopo che si è processato il t-mo job, ma prima di effettuare i cambi utensile
c	la capacità, cioè il numero massimo di utensili contemporaneamente caricabili
$\mathbf{e}$	un n-vettore di 1
x_{it}	vale 1 se il job i è il t-mo job nella sequenza, 0 altrimenti
$\mathbf{W}_t$	un n-vettore di 0 ed 1. $W_{tj} = 1$ se l'utensile j è caricato sulla macchina al tempo t, cioè dopo aver eseguito il t-mo job nella sequenza e prima di eseguire alcuno switch
$\mathbf{P}_t$	un n-vettore di 0 ed 1. $P_{tj} = 1$ se al tempo t è necessario caricare sulla macchina l'utensile j

La formulazione del problema prevede dunque la minimizzazione del seguente valore

$$v = min \sum_{t=1}^{m-1} e \cdot P_t$$

soggetta ad i seguenti vincoli

$$P_{tj} \geq W_{t+1,j} - W_{t,j} \quad t = 1, \ldots, m-1 \; ; \; j = 1, \ldots, n \qquad (7.1)$$

$$P_{tj} \geq 0 \quad t = 1, \ldots, m \; ; \; j = 1, \ldots, n \qquad (7.2)$$

$$e \cdot W_t \leq c \quad t = 1, \ldots, m \qquad (7.3)$$

$$x_{it} A_{ij} \leq W_{tj} \quad i, t = 1, \ldots, m \; ; \; j = 1, \ldots, n \qquad (7.4)$$

$$\sum_{i=1}^{m} x_{it} = 1 \quad t = 1, \ldots, m \qquad (7.5)$$

$$\sum_{t=1}^{m} x_{it} = 1 \quad i = 1, \ldots, m \qquad (7.6)$$

$$x_{it} \in \{0, 1\} \quad i, t = 1, \ldots, m \qquad (7.7)$$

$$W_{tj} \in \{0, 1\} \quad t = 1, \ldots, m \; ; \; j = 1, \ldots, n \qquad (7.8)$$

Poiché, per tale problema, non c'è alcun vincolo su quanto tempo prima un utensile possa essere caricato sul macchinario rispetto all'istante in cui servirà, e poiché per ridurre i cambi utensile, non è necessario rimuovere un utensile, salvo che il suo posto debba essere preso da un altro utensile, nel vincolo (7.3) si può usare il segno di uguaglianza.

Ipotizzando, senza perdere di generalità, che ogni utensile sia necessario ad almeno un job, un lower bound per il numero di cambi utensile è il valore $m - c$. Infatti, supponendo di caricare c utensili prima di processare il primo job, cioè al tempo 0, allora ciascuno degli altri $m - c$ utensili deve essere caricato almeno una volta sul macchinario, e ciò dimostra la validità dell'affermazione.

L'analogia con il MOSP consiste nel fatto che si può far corrispondere logicamente ogni job ad un pattern ed ogni utensile ad uno stack. I due problemi, in generale, non sono equivalenti, ma valgono le seguenti proprietà.

Innanzitutto, se il valore ottimo (minimo) per il MOSP, denotato con c^*, non è maggiore di c, allora vale l'uguaglianza $v = m - c$. Infatti l'apertura di uno stack corrisponde al caricamento dell'utensile e la chiusura di uno stack alla sua rimozione definitiva. Se dunque si tengono caricati sul macchinario gli utensili corrispondenti agli al più c^* stack aperti associati alla soluzione ottima per il MOSP, e si precaricano al tempo 0 i c utensili corrispondenti ai primi c stack che vengono aperti nella soluzione ottima del MOSP, allora nessun utensile viene rimosso per poi essere ricaricato e, quindi, $v = m - c$.

Inoltre, se $c^* > c$, allora $v > m - c$. Se, infatti, per assurdo, vi fosse una soluzione per l'MTSP con valore $m - c$, allora nessun utensile verrebbe rimosso senza dover poi essere ricaricato e ciò corrisponde, dal punto di vista del MOSP, alla chiusura di uno stack, mentre la presenza di un utensile sul macchinario corrisponde al fatto che uno stack sia aperto. Dunque la sequenza ottima per l'MTSP determina per il MOSP un massimo numero di stack aperti pari a c, contro l'ipotesi che c^* è il minimo possibile ed è maggiore di c.

Da questa seconda proprietà deriva il fatto che un modello di programmazione lineare intera per il MOSP può essere ricavato da quello per l'MTSP, introducendo c come variabile, da minimizzare, e non come costante, ed aggiungendo il vincolo $\sum_{t=1}^{m-1} e \cdot P_t = m - c$ ai vincoli (7.1)-(7.8) del modello dell'MTSP.

Yanasse indica che gli scarsi risultati computazionali ottenuti da Tang e Dendardo (1988) per l'MTSP, utilizzando il modello lineare intero descritto, si riflettono in risultati di altrettanta qualità per il MOSP, come confermato anche in Becceneri (1999). Di conseguenza, ha proposto un apposito algoritmo branch and bound, descritto nel paragrafo successivo.

Il lavoro di Linhares e Yanasse (2002) ha mostrato che è falsa anche la congettura secondo cui esiste sempre una soluzione contemporaneamente ottima per il MOSP e per l'MTSP, che era stata formulata nel lavoro di Yanasse (1997b) insieme alle altre congetture discusse nel paragrafo precedente, circa le relazioni fra il MOSP, il MORP e l'MDP. Lo stesso lavoro ha però introdotto una ulteriore congettura, circa la possibilità che esista sempre una soluzione contemporaneamente ottima per l'MDP e per l'MTSP. Ha inoltre esteso anche all'MTSP la prova che non può esistere un algoritmo di approssimazione assoluta, salvo che le due classi di problemi **P** ed **NP** siano equivalenti.

Relativamente al MOSP, un secondo possibile modello lineare è quello presentato da Belov (2003), discusso nel Capitolo successivo poiché combina sia il problema di Cutting che quello di Sequencing.

7.2.3 Un approccio esatto branch and bound per il MOSP

Tale algoritmo si basa sulla seguente considerazione. Ogni sequenza Π di pattern per un problema P è equivalente ad una sottosequenza ridotta, associata ad un problema con un minor numero di pattern, contenente solo i pattern di P che, nella sequenza Π, corrispondono alle chiusure degli stack aperti. Si supponga ad esempio, per il problema i cui dati sono stati riportati in Tabella 7.1, di considerare la sequenza $\Pi = (1, 2, 3, 4, 5, 6)$. Come già evidenziato nella Tabella 7.3, i pattern corrispondenti alle chiusure degli stack sono quelli di indice 2, 3, 4, 5, 6. Infatti, dopo aver processato il pattern 2, si chiude lo stack relativo al gruppo *b*. Analogamente il pattern 3 corrisponde alla chiusura dello stack per *d*, il pattern 4 per *c*, il pattern 5 per *a* ed il pattern 6 per *e* ed *f*. Alla sottosequenza (2, 3, 4, 5, 6) corrisponde lo stesso valore massimo di numero di stack aperti della sequenza (1, 2, 3, 4, 5, 6).

Questa considerazione permette di passare, senza perdita di generalità, dalla ricerca della sequenza di pattern, alla ricerca della equivalente sequenza di chiusura degli stack. Ad una fissata sequenza di chiusura degli stack, denotata ad esempio con $\Phi = (\varphi(1), \ldots, \varphi(n))$, corrisponde la sequenza di pattern Π_Φ costruita come spiegato nel seguito. Inizialmente si pongono tutti i k_1 pattern che contengono pezzi del gruppo associato allo stack $\varphi(1)$. Si sottolinea che si suppone implicitamente che, qualunque sia la sottosequenza $(\pi(1), \ldots, \pi(k_1))$ con cui vengono posti tali pattern, non si produca alcuna chiusura di pattern se non dopo il pattern $\pi(k_1)$. Inoltre, si suppone implicitamente che, se altri stack si chiudono oltre a $\varphi(1)$, questi siano gli stack immediatamente consecutivi a $\varphi(1)$ nella sequenza Φ. Dopo tali primi k_1 pattern, si aggiungono tutti i pattern residui che contengono pezzi relativi allo stack $\varphi(2)$, in numero pari a k_2. Per quanto detto prima, k_2 potrebbe essere pari a zero. Anche per tali k_2 pattern valgono le stesse considerazioni fatte per i primi k_1 pattern, associati a $\varphi(1)$. Si prosegue in tal modo, per i pattern ulteriormente residui associati a $\varphi(3)$, e così via, fino agli ultimi pattern rimasti, sicuramente associati solo a $\varphi(n)$.

La considerazione finale da fare, che Yanasse rende implicitamente, è che non è detto che siano possibili tutte le sequenze di chiusura. Infatti, per una data ipotetica sequenza Φ di chiusura degli stack, non è detto che, per qualche valore di i, non vi sia una sottosequenza dei k_i pattern aggiunti, tale da far chiudere uno stack $\varphi(h)$, con $h > i$, prima dell'aggiunta dell'ultimo di tali pattern, ed anzi è possibile che, per qualche valore di i, nessuna sottosequenza dei k_i pattern aggiunti sia tale da non far chiudere alcun altro stack prima dell'aggiunta dell'ultimo di tali pattern. Tuttavia, questo non è un problema, sulla base delle seguenti due considerazioni, non rese esplicitamente da Yanasse.

Innanzitutto, ogni sottosequenza di k_i pattern non "buona", determina la generazione di una sequenza parziale che è associata ad una sequenza di chiusura Φ' diversa da Φ, e poiché

l'algoritmo deve verificare tutte le possibili sequenze di chiusura degli stack, esso risulta corretto, dato che verifica anche la sequenza Φ′.

In secondo luogo, per una eventuale sequenza di chiusura Φ, impossibile da realizzare, basta considerare come atomici gli insiemi di k_i pattern aggiunti, per ogni *i*, come se fossero agglomerati in un unico macro-pattern. Con ciò si vuol dire che la preventiva chiusura di uno stack diverso da φ(*i*) può solo determinare una riduzione del massimo numero di stack aperti associati alla sequenza Π, rispetto allo stesso numero associato alla sequenza fittizia Π_Φ, che contiene gli *n* macro-pattern relativi a Φ. Di conseguenza, nella costruzione di una sequenza Φ, basta considerare il massimo numero di stack aperti associato alla sequenza fittizia di macro-pattern Π_Φ. Se per qualche valore di *i*, esiste una sottosequenza dei k_i pattern che determina la chiusura preventiva di uno stack φ(*h*), con *h* > *i*, ciò vuol dire che esiste una sequenza di chiusura Φ′, ottenuta, a partire da Φ, spostando l'indice *h* nella posizione precedente a quella dell'indice *i*, cui sia associata una sequenza di pattern Π' con un massimo numero di stack aperti sicuramente non maggiore di quello associato a Π_Φ. Laddove i due valori dovessero essere uguali ed un algoritmo di enumerazione implicita evitasse di esplorare un insieme di sequenze contenente Φ′, ciò vorrebbe dire che la posizione relativa di *h* rispetto ad *i*, nella sequenza di chiusura Φ, è ininfluente ai fini del massimo numero di stack aperti, e che all'atto concreto della costruzione della sequenza di pattern associata, si può finire per costruire comunque la sequenza Π′, se cioè si sceglie la sottosequenza dei k_i pattern che determina una chiusura preventiva di φ(*h*), rispetto alla chiusura di φ(*i*).

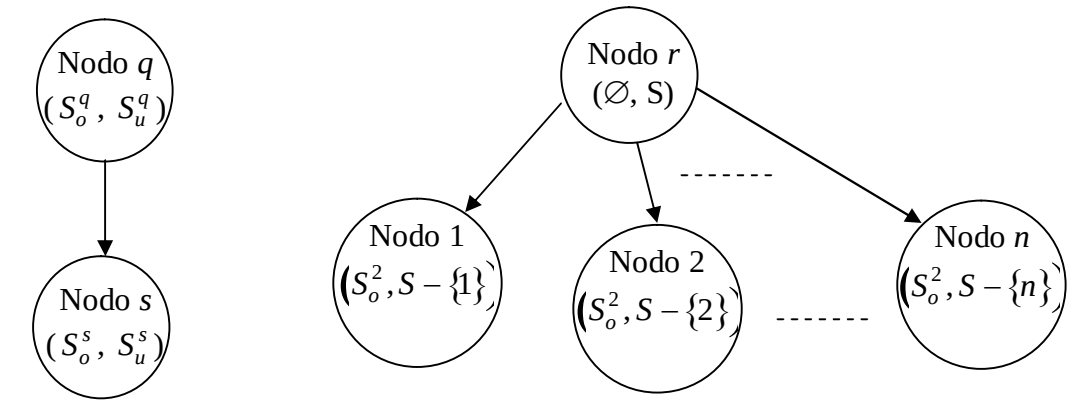
Nell'esempio di Tabella 7.1, dalla sequenza di chiusura Φ = (*a*, *f*, *c*, *d*, *e*, *b*), si genererebbe la sequenza di macro-pattern Π_Φ = ((1, 4, 5), (2, 6), (3)). I primi tre pattern 1, 4 e 5, determinano la chiusura dello stack *a* e l'apertura di tutti gli altri stack. Per i due pattern aggiuntivi 2 e 6 avviene il caso delineato prima. Se infatti si sceglie la sottosequenza (2, 6), il pattern 2 determina la chiusura dello stack *b*, prima della chiusura dello stack *f*, che avviene poi con il pattern 6. Anche se si sceglie la sequenza (6, 2) si ottiene comunque la chiusura dello stack *b* prima della chiusura dello stack *c*, al contrario di quanto stabilisce la sequenza Φ, che dunque risulta impossibile.

Tornando alla descrizione dell'algoritmo, l'esplorazione dell'insieme di sequenze di chiusura avviene associando, ad ogni nodo *q* dell'albero di ricerca, due insiemi, denotati rispettivamente con S_u^q ed S_o^q. Il primo è l'insieme degli stack non ancora chiusi. Il secondo, sottoinsieme del primo, è l'insieme degli stack aperti. Con *S* si denota l'insieme di tutti gli stack. Dunque, nel nodo radice *r* si parte con le posizioni $S_u^r = S$ ed $S_o^r = \varnothing$.

Per ogni nodo *q*, si considerano tanti successori quanti sono gli elementi di S_u^q. Ad ogni nodo successore *s* corrisponde la chiusura di uno degli stack di S_u^q. Se, per un nodo successore *s*, si indica con *g* lo stack chiuso, allora al nodo *s* si associa l'insieme $S_u^s = S_u^q - \{g\}$, mentre l'insieme S_o^s è ottenuto, di conseguenza, come l'insieme degli stack che si aprono a valle dell'aggiunta, rispetto ai pattern già utilizzati per la chiusura degli elementi di $S - S_u^q$, di tutti i pattern necessari alla chiusura dello stack *g*. La Figura 7.1, ripresa dal lavoro di Yanasse, mostra quanto detto. La Figura 7.1a schematizza il passaggio da un nodo *q* ad un nodo successore *s*, mentre la Figura 7.1b schematizza la generazione dei nodi successori a partire dal nodo radice.

Dalla relazione fra S_u^s ed S_u^q si evince che non vengono considerate altre chiusure oltre quella di *g*. Tuttavia, attraverso un esempio, Yanasse mostra implicitamente che da S_o^s vengono tolti gli altri stack eventualmente chiusi insieme a *g*. Relativamente ad S_o^s, toglierli o

non toglierli non ha alcun effetto sull'algoritmo. Si coglie però l'occasione per affermare che li si potrebbe togliere anche da S_u^s, risparmiandosi così la generazione di nodi dummy, in cui non si aggiungono pattern alla sequenza parziale. Nell'algoritmo di Yanasse è invece possibile, per qualche stack *d* di S_u^s, che l'eventuale numero di pattern da considerare per la chiusura di *d* è pari a zero. Inoltre, secondo quanto spiegato in precedenza, potrebbe non esistere alcuna sottosequenza dei pattern aggiunti per la chiusura di *g*, che permetta la non preventiva chiusura di *d* rispetto a *g*. In pratica, la sequenza parziale di pattern che si costruisce implicitamente in relazione alla sequenza parziale di chiusura associata al path dal nodo radice al nodo corrente *q*, è una sequenza di macro-pattern. Comunque, per quanto spiegato in precedenza, l'algoritmo risulta corretto.



a) nodo successore b) branching nel nodo radice

Figura 7.1 Schematizzazione del branch and bound per il MOSP

Per accelerare la procedura, l'algoritmo fa uso di alcuni lower bound. Un primo lower bound consiste nel numero di gruppi associati a ciascuno dei pattern iniziali. Infatti, se un pattern contiene *l* gruppi, allora non è possibile avere meno di *l* stack aperti, dato che dopo aver processato quel pattern, e prima delle chiusure, vi saranno almeno *l* stack aperti. Tale lower bound è indicato con LB_1. Esso era già stato introdotto nel lavoro di Yuen e Richardson (1995), i quali hanno stabilito, come semplice criterio di ottimalità per metodi euristici, la verifica dell'eventuale uguaglianza fra tale lower bound ed il valore della soluzione ottenuta dall'euristica.

Si definisce inoltre, per il nodo *q*, in relazione ad un suo nodo successore *s*, il valore v_q^s pari al numero di stack aperti dopo aver processato il (macro-)pattern aggiunto per la chiusura del relativo stack *g*, e prima delle conseguenti chiusure di stack, fra cui certamente quella dello stack *g*. Vale la relazione $v_q^s \geq \left|S_o^q\right|$, cioè v_q^s è almeno pari al numero di stack rimasti aperti nel nodo *q*, ed anche la relazione $v_q^s > \left|S_o^s\right|$. La disuguaglianza stretta è dovuta al fatto che almeno uno stack viene chiuso, e cioè lo stack *g* corrispondente al passaggio dal nodo *q* al nodo *s*.

Per ogni nodo *q*, una volta generati tutti i nodi successori, si può calcolare un secondo lower bound, come minimo dei valori v_q^s al variare dei nodi successori *s*. Tale lower bound è indicato con $LB^q{}_2$. Si noti che tale lower bound rappresenta il minimo possibile per il massimo numero di stack aperti associato ad una qualunque sequenza parziale (benché eventualmente fittizia, nel senso di sequenze di macro-pattern) contenente i soli pattern necessari per la chiusura degli stack corrispondenti al path che va dal nodo radice *r* al nodo corrente *q* per poi

passare attraverso uno dei nodi successori *s* di *q*. Tale lower bound per queste sequenze parziali è ovviamente un lower bound anche per le sequenze complete che se ne possono ricavare aggiungendo i pattern ancora non presenti.

Si noti inoltre che i valori v_q^s tengono in realtà conto, in relazione alle suddette sequenze parziali, solo della situazione a valle del processo del macro-pattern aggiunto, senza avere memoria delle situazioni associate a valle del processo dei macro-pattern aggiunti nei passaggi dal nodo radice *r* fino al nodo corrente *q*. Per tenerne conto, si utilizza sia un terzo lower bound, indicato con LB_3 e pari al valore v_p^q relativo al passaggio dal nodo predecessore *p* al nodo corrente *q*, sia la propagazione del lower bound ottenuto per il nodo predecessore *p*. In definitiva, si utilizzano le seguenti relazioni generali

$$\begin{cases} LB^r = \max\left\{LB_1^r, LB_2^r\right\} \\ LB^q = \max\left\{LB_2^q, LB_3^q, LB^p\right\} \quad if \quad j \neq r \end{cases} \tag{7.9}$$

Si sottolinea che in principio sembrerebbe più opportuno utilizzare la "propagazione" dei lower bound dal predecessore al nodo corrente, solo del valore di LB_3, utilizzando cioè le seguenti relazioni, in cui si utilizza la notazione LB_{3+} al posto di LB_3, dato che il valore può essere diverso rispetto a quello di LB_3, per l'inglobamento della propagazione.

$$\begin{cases} LB^r = \max\left\{LB_1^r, LB_2^r\right\} \\ LB_3^r = 0 \\ LB_{3+}^q = \max\left\{LB_{3+}^p, v_p^q\right\} \quad if \quad j \neq r \\ LB^q = \max\left\{LB_1, LB_2^q, LB_{3+}^q\right\} \quad if \quad j \neq r \end{cases} \tag{7.10}$$

Infatti, il lower bound per il nodo corrente *q* non dovrebbe essere influenzato da ciò che poteva avvenire passando dal suo predecessore *p* ad un nodo successore di *p* ma diverso da *q*, mentre il valore di LB_2 associato al nodo *p*, e dunque il valore di LB^p, è correlato alle situazioni associate a tutti i successori di *p*. Si noti inoltre che ne scaturisce la necessità di aggiungere LB_1, indipendente dal nodo, nella formula del lower bound per qualunque nodo.

Tuttavia, per le relazioni (7.9), valgono le seguenti considerazioni. Il valore LB_2^p è ottenuto come minimo fra un insieme di valori fra cui c'è anche v_p^q, cioè LB_3^q. Dunque, LB_2^p è minore o uguale di LB_3^q, ed allora è come se si fossero propagati solo LB_1 ed LB_3^p, cioè in modo equivalente alle relazioni (7.10), dato che il calcolo di LB^q prevede di utilizzare il massimo fra LB_3^q, pari a v_p^q, ed LB^p. Ciò vuol dire che il risultato è lo stesso, ma si risparmia memoria, dovendo mantenere solo un valore di lower bound per ciascun nodo.

L'algoritmo utilizza anche un upper bound per il massimo numero di stack aperti associato alle possibili sottosequenze parziali finali che si potranno aggiungere a quella parziale iniziale già costruita in relazione al path dal nodo radice al nodo corrente *q*. Tale upper bound è posto semplicemente pari ad $\left|S_u^q\right|$, cioè pari al numero di stack ancora non chiusi, indipendentemente dal fatto che siano stati aperti o meno. Esso è denotatao con UP^q. Infatti il caso peggiore possibile è quello in cui tutti i relativi stack siano aperti, prima che se ne cominci a chiudere qualcuno. Si sottolinea anche che la critica originale fatta sopra circa

l'eventuale eliminazione dall'insieme S_u^q di più di uno stack per volta produrrebbe anche un migliore upper bound.

Quando in un nodo q risulta $LB^q \geq UP^q$, il nodo viene tagliato, ed il valore di LB^q è eventualmente utilizzato per aggiornare la migliore soluzione nota. Infatti, per quanto spiegato in precedenza, e con un'eccezione indicata nel seguito, LB^q è pari esattamente al massimo di stack aperti associato alla sequenza parziale (eventualmente fittizia, cioè di macro-pattern) costruita in base al path dal nodo radice al nodo corrente q, mentre UP^q è il massimo numero di stack aperti associato alle eventuali sequenze parziali da aggiungere in coda a quella già generata. Dunque, qualunque sia la sequenza parziale aggiunta, il massimo numero di stack aperti per la sequenza totale resterà pari ad LB^q e potrà dunque essere scelta a piacere. Un caso particolare è quello in cui LB^q sia pari al lower bound LB_1. In questo caso, indipendentemente dal fatto che un numero di stack aperti pari ad LB_1 sia stato già raggiunto o meno in relazione alla sequenza parziale già costruita, è certo che qualunque sequenza aggiuntiva dei pattern ancora non considerati produrrà una sequenza completa con massimo numero di stack aperti pari ad LB_1. Come nota si indica che se il valore LB_1 non è stato ancora raggiunto, ciò implica che $UP^q = LB_1$ e che c'è un pattern, ancora non aggiunto, contenente tutti i gruppi relativi agli stack del corrente insieme S_u^q.

La strategia con cui vengono esplorati i nodi è di tipo *depth-first*, selezionando ogni volta il successore più promettente in modo greedy, sulla base del lower bound LB_3. Infatti il lower bound LB_1 è lo stesso per tutti i nodi ed il lower bound LB_2 è calcolabile solo dopo aver generato gli ulteriori successori.

Come miglioramento rispetto a questo algoritmo, nel lavoro di Yanasse si introduce il concetto di dominanza fra gruppi di pezzi, cioè fra gli stack. Se T_a è l'insieme di pattern contenenti pezzi del gruppo a e T_b è l'insieme di pattern contenente pezzi del gruppo b, allora si dice che il gruppo b domina il gruppo a se vale la relazione $T_a \subseteq T_b$. In tal caso, il gruppo dominante b non viene mai scelto per il branching, a meno che il gruppo dominato a non sia già stato eliminato dall'insieme S_u associato al nodo corrente. E' infatti certo che le sequenze di chiusura Φ in cui il gruppo b assume una posizione precedente rispetto a quella del gruppo a, sono impossibili, salvo nel caso $T_a = T_b$, e generano sequenze di pattern certamente generabili da sequenze di chiusura in cui il gruppo a si trova prima del gruppo b.

Si conclude indicando che il passaggio dalla ricerca di una sequenza di pattern alla ricerca di una sequenza di chiusura degli stack determina certamente un miglioramento in termini di tempi computazionali quando il numero di gruppi di pezzi sia inferiore al numero di pattern, cioè quando $n < m$. In realtà, il lavoro di Yanasse (1998) mostra che un attento schema di enumerazione implicita delle sequenze di chiusura degli ordini non è mai peggiore rispetto a schemi di enumerazione delle sequenze di pattern, come quelli utilizzati, ad esempio, in Yuen e Richardson (1995), Faggioli e Bentivoglio (1998), Linhares, Yanasse e Torreão (1999) ed Oliveira e Lorena (2002).

7.2.4 Raffinamenti dell'approccio branch and bound per il MOSP

Due lavori, quello di Yanasse e Limeira (2004) e quello di Becceneri, Yanasse e Soma (2004), propongono miglioramenti per l'approccio branch and bound. I raffinamenti vengono descritti nel seguito in due corrispondenti sotto-paragrafi.

7.2.4.1 *Raffinamenti basati sulla decomposizione del problema*

Il lavoro di Yanasse e Limeira (2004) riprende lo schema branch and bound di Yanasse (1997b), apportando miglioramenti basati su un modo differente di modellare il problema.

In particolare, ad ogni istanza del problema, si associa un grafo non orientato, denominato *MOSP graph*. Esso ha un nodo per ogni gruppo di pezzi (o stack) e, per ogni coppia di gruppi che abbiano pezzi su almeno uno stesso pattern, vi è un arco fra i nodi corrispondenti ai due gruppi. Ad ogni pattern corrisponde dunque un clique, e quindi il MOSP graph è un'unione di cliques.

Il MOSP graph associato all'esempio di Tabella 7.1 è riportato in Figura 7.2.

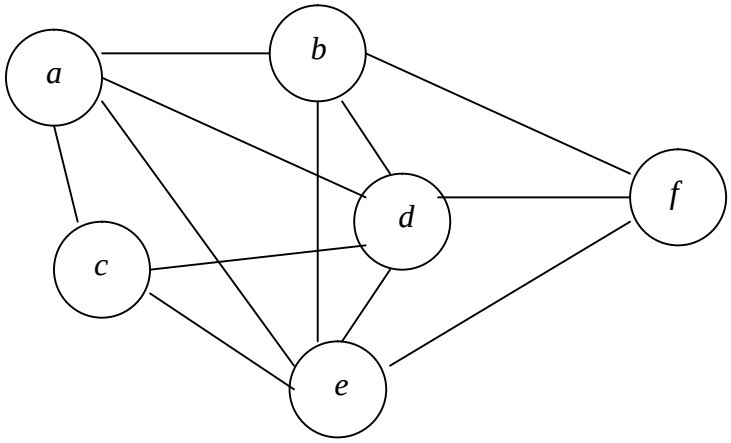


Figura 7.2 Un esempio di MOSP graph

La generazione di questo grafo generalizza quella già proposta in Lins (1989) per il caso in cui su ogni pattern vi siano pezzi associati ad al più due gruppi diversi.

I miglioramenti si basano innanzitutto su alcune considerazioni, ottenute nel lavoro di Yanasse (1997c), che ha appunto introdotto la suddetta rappresentazione tramite grafo.

La prima proposizione ivi dimostrata è che, data una sequenza di pattern, la sequenza che processa i pattern in ordine esattamente inverso determina lo stesso massimo numero di stack aperti.

La seconda proposizione afferma che, aggiungendo pattern all'istanza di un problema, si ottiene una istanza per la quale il valore ottimo è non minore rispetto a quello dell'istanza di partenza. Ciò è stato sfruttato anche in Yanasse (1997b) nell'utilizzo dei lower bounds, e si traduce nel fatto che, se per una sottosequenza di pattern, il massimo numero di stack aperti assume un certo valore, esso non può ridursi per sequenze complete che contengono quella sottosequenza.

La terza proposizione identifica le connessioni fra il problema MOSP ed il seguente problema sul MOSP graph. Fissata una permutazione degli archi del grafo, si realizza una sequenza di step. Ad ogni step viene aggiunto il successivo arco, seguendo l'ordine stabilito dalla permutazione. Quando un nodo, estremo di un arco aggiunto, è raggiunto per la prima volta, si dice che il nodo viene *etichettato*. Quando, a valle dell'eventuale nuova etichettatura, si verifica che, per il nodo etichettato, tutti gli archi che vi incidono sono stati già aggiunti, allora il nodo viene chiuso. Il problema consiste allora nel trovare la permutazione che minimizza il massimo numero di nodi etichettati, al variare degli step. Si sottolinea che si considerano tutte le possibili permutazioni degli archi, senza la necessità di avere, per due archi consecutivi, un nodo intermedio condiviso su cui passare. Si parla di *Graph Traversing Problem*.

la tecnica costruttiva con cui ottenere una soluzione per il MOSP a partire da una permutazione degli archi del relativo MOSP graph si basa sulla seguente regola. Ogni pattern, detto ad esempio P_i, è aggiunto in corrispondenza del primo step dopo il quale tutti i nodi corrispondenti ai gruppi che hanno almeno un pezzo sul pattern P_i, sono stati etichettati. Nel lavoro di Yanasse (1997c) si dimostra che la soluzione ottima per il problema del MOSP

graph può essere trasformata, attraverso tale procedimento costruttivo, in una delle soluzioni ottime per il problema MOSP di partenza.

Si sottolinea che il MOSP graph potrebbe avere più componenti, non connesse fra loro. In tal caso, basta risolvere il problema indipendentemente per ciascuna componente, per poi ottenere la sequenza totale associata al grafo iniziale, ponendo in un ordine qualsiasi le sottosequenze relative a ciascun componente. Nel seguito si suppone dunque, senza perdita di generalità, che il grafo sia connesso.

Le innovazioni apportate nel lavoro di Yanasse e Limeira (2004) si basano sulle due seguenti proposizioni.

La prima fa riferimento al caso speciale in cui il MOSP graph sia costituito nel modo schematizzato in Figura 7.3.

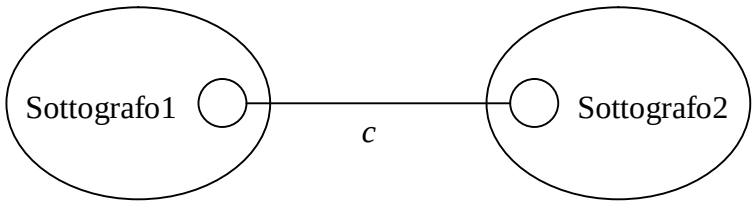


Figura 7.3 Caso speciale di MOSP graph

In questo caso particolare, vi è un arco *c*, che congiunge, attraverso i due nodi su cui incide, due sottografi che, senza l'arco *c*, sarebbero disconnessi l'uno dall'altro. La proposizione afferma che esiste una soluzione ottima in cui tutti gli archi di ciascun sottografo siano posti, nella relativa permutazione, in sequenza l'uno dietro l'altro. La dimostrazione si basa sull'idea intuitiva per cui conviene chiudere tutti i nodi di un sottografo prima di etichettare nodi dell'altro sottografo. Ovviamente ciò non è possibile per i nodi speciali, estremi dell'arco *c*, ma la proposizione vale ugualmente.

Applicando la stessa idea, in modo iterativo e con considerazioni speciali per gli analoghi dei due nodi speciali del caso di Figura 7.3, si ottiene un risultato simile per il caso di grafo "a stella" schematizzato in Figura 7.4, in cui c'è un nodo *p*, a partire dal quale partono archi che congiungono il nodo *p* con *k* sottografi che, escludendo gli archi incidenti in *p*, risultano disconnessi l'uno dall'altro.

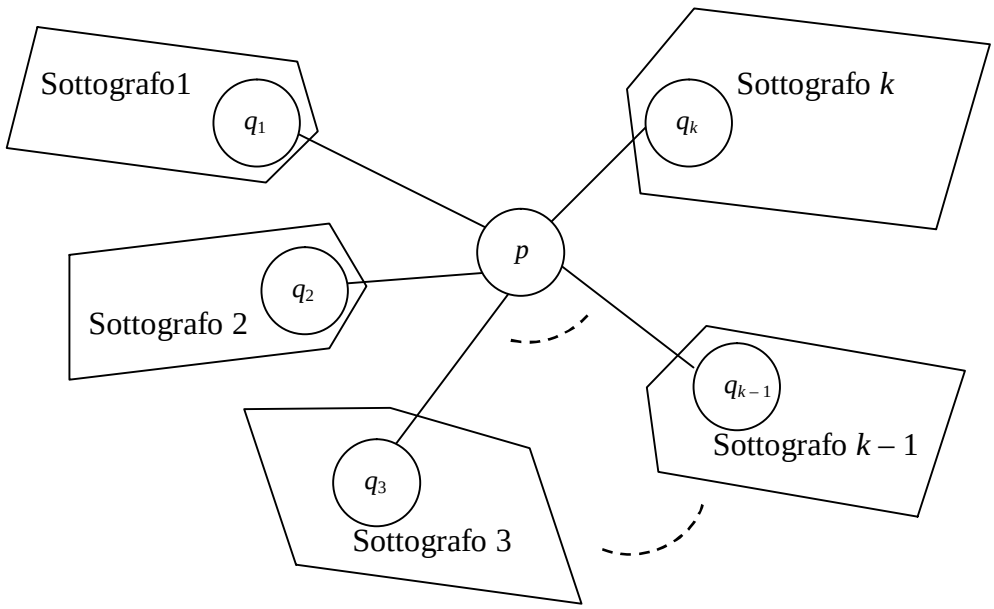


Figura 7.4 MOSP graph a stella

Siano $\Gamma_1, \Gamma_2, \ldots, \Gamma_k$ le permutazioni ottime relative a ciascuno dei k sottografi, cui corrispondono valori parziali ottimi $v_1, \ldots, v_k$ relativamente a ciascuno dei sottografi. Siano inoltre $\Gamma^+_1, \Gamma^+_2, \ldots, \Gamma^+_k$ le permutazioni relative ai sottografi aumentati con il corrispondente arco (p, q_i) ed il nodo p e siano $v^+_1, \ldots, v^+_k$ i relativi valori parziali ottimi associati.

La proposizione afferma che, supponendo, senza perdita di generalità, che valgano le due relazioni $v_1 \geq v_k \geq v_2 \geq v_i$ per $i = 3, \ldots, k-1$, e $v^+_1 \geq v^+_k \geq v_i$ per $i = 2, \ldots, k-1$, una permutazione ottima per il grafo completo si ottiene con entrambe le sequenze, similari, $\neg\Gamma^+_1, \Gamma^*_{j(1)}, \ldots, \Gamma^*_{j(k-2)}, \Gamma^+_k$ e $\neg\Gamma^+_k, \Gamma^*_{j(1)}, \ldots, \Gamma^*_{j(k-2)}, \Gamma^+_1$, di valore $\max\{v^+_1, v_2 + 1\}$ dove $\neg\Gamma$ rappresenta la sequenza inversa rispetto a Γ, la sequenza $(j(1), \ldots, j(k-2))$ è una permutazione qualsiasi degli elementi da 2 a $k-2$, e Γ^*_i è la sequenza di archi ottenuta a partire da Γ_i ed inserendo l'arco (p, q_i) subito dopo il primo arco che determina l'etichettatura del nodo speciale q_i. In pratica, non è rilevante l'ordine con cui vengono inserite le sottosequenze $\Gamma^*_2, \ldots, \Gamma^*_{k-2}$.

In generale, la verifica dell'esistenza di nodi che possano assumere il ruolo di nodo centrale, o separatore, non è una cosa semplice (Ellis, Sudborough e Turner, 1994), per cui l'algoritmo verifica soltanto la presenza di sottografi con strutture ad albero, visto che tutti i nodi di un tale sottografo possono assumere il ruolo di centro di una stella. A valle della scrematura del MOSP graph ottenuta per rimozione dei sottografi ad albero, trovati verificando iterativamente se vi siano nodi con grado di connessione 0 oppure 1 (a valle della rimozione dei nodi precedenti), si può risolvere il problema sui sottografi non connessi del grafo scremato, sfruttando la procedura branch and bound di Yanasse (1997b) per ciascuno di essi. Il fatto che tale procedura sia applicata probabilmente a sottografi separati riduce il numero totale di nodi generati nelle rispettive procedure branch and bound, rispetto all'applicazione direttamente sul grafo di partenza.

Inoltre, e questo vale anche più in generale, in presenza di una struttura a stella qualsiasi, cioè non limitata al caso di sottografi ad albero, si può sfruttare il fatto che non è necessario ottenere per ciascun sottografo la soluzione ottima. Se infatti è noto il valore ottimo di un sottografo *a*, e si trova una soluzione non necessariamente ottima per un altro sottografo *b*, di valore strettamente minore di quello relativo al sottografo *a*, allora si può accettare tale soluzione, visto che determina una sottosequenza che certamente non implicherà, per le sequenze complete che lo contengono, un aumento del valore rispetto al valore associato alla sottosequenza per il sottografo *a*. Inoltre, se il valore di una soluzione non necessariamente ottima per il sottografo *b* non è maggiore dei valori ottimi, già noti, di almeno altri tre sottografi, vale la stessa conclusione.

Proprietà analoghe possono essere utilizzate sia nel caso più semplice di componenti non connesse del grafo di partenza, sia nel caso già schematizzato in Figura 7.3. La stessa considerazione, relativa al caso di componenti non connesse, è stata fatta nel lavoro di Yuen e Richardson (1995). Tale lavoro si focalizza sulla ricerca di una sequenza di pattern, come anche in Faggioli e Bentivoglio (1998), piuttosto che di una sequenza di chiusura degli stack. Sfrutta inoltre una caratterizzazione del problema basata su una trasformazione in un grafo, complementare rispetto al caso esaminato, e cioè associando i nodi ad i pattern ed introducendo archi fra i nodi corrispondenti a coppie di pattern che condividono l'associazione ad almeno uno stesso gruppo di pezzi.

Infine, si sottolinea che, per strutture particolari del grafo, o di uno dei sottografi, i relativi problemi di MOSP sono risolvibili in tempo polinomiale (Ellis, Sudborough e Turner, 1994; Yanasse, 1996, 1997a; Yanasse, Becceneri e Soma, 1998), come ad esempio, oltre al caso dell'albero, nel caso di grafi con una struttura circolare.

7.2.4.2 *Raffinamenti basati sulla riduzione del problema*

Il lavoro di Becceneri, Yanasse e Soma (2004) sfrutta, come il lavoro di Yanasse e Limeira (2004) precedentemente descritto, la trasformazione di una istanza di MOSP in un problema su grafo.

Vengono introdotti i seguenti nuovi elementi. Innanzitutto fasi di riduzione del problema, e poi nuovi lower bounds globali, uno basato su una procedura euristica costruttiva, ed altri basati su semplici considerazioni. Si rimanda la descrizione dell'euristica costruttiva al paragrafo successivo, proseguendo nel seguito con la descrizione delle tecniche di riduzione del problema.

Vi è innanzitutto una fase di pre-processing in cui vengono eliminati i pattern che comprendono pezzi di gruppi, per i quali esiste un altro pattern che pure comprende almeno un pezzo per ciascuno di essi. Cioè, se si definisce, per ogni pattern i, l'insieme P_i i cui valori sono gli indici dei gruppi con pezzi compresi nel pattern i, allora il pattern i viene eliminato se esiste un pattern k tale che $P_i \subseteq P_k$. A valle della risoluzione del problema ridotto che non ha il pattern i, questo viene reinserito immediatamente dopo il pattern P_k, per ottenere una soluzione per il problema iniziale. Nel caso in cui $P_i = P_k$, uno solo di essi viene eliminato. In pratica si elimina, finché è possibile, un pattern alla volta, per poi reinserirli a valle della risoluzione del problema ridotto, per ottenere una soluzione al problema originario.

Dopo la riduzione, l'algoritmo verifica se il grafo associato al problema ridotto è di tipo speciale, ad esempio un albero o un semplice ciclo, poiché in tal caso, come detto in precedenza, è possibile ottenere la sequenza ottima in tempo polinomiale. Ad esempio, nel caso di grafo a poligono, basta costruire, in tempo lineare, una sequenza di pattern associata alla visita del grafo realizzata per archi consecutivi.

Sul problema ridotto, e nel caso in cui non ci si trovi in uno dei casi speciali, si operano ulteriori scremature basate sulle due seguenti proposizioni relative a dominanze ed equivalenze fra nodi.

Si definisce come $N(j)$ l'insieme dei nodi adiacenti ad un nodo j (nel grafo associato al problema ridotto) e di definisce anche, per ogni nodo j, l'insieme $N'(j) = N(j) \cup \{j\}$.

Il primo criterio di dominanza fra due nodi j ed h si basa sulla relazione $N'(j) \subseteq N'(h)$. In tal caso, data una sequenza di chiusura degli stack, se in tale sequenza j è successivo ad h, è possibile costruire una sequenza in cui j appare prima di h, con un massimo numero di stack aperti non maggiore di quello associato alla sequenza iniziale.

La prova consiste nel fatto che, poiché $N'(j) \subseteq N'(h)$, la processazione di tutti i pattern necessari per chiudere h implica automaticamente che anche tutti i pattern necessari per chiudere j siano processati.

In realtà gli autori affermano che tale criterio di dominanza comporta un costo computazionale che non bilancia i vantaggi che ne derivano, per cui esso non viene utilizzato.

Il contrario avviene per il secondo criterio, di equivalenza, secondo cui, se vale una delle due uguaglianze $N(j) = N(h)$ oppure $N'(j) = N'(h)$, allora, a partire da una qualunque sequenza di chiusura fattibile, in cui i due stack j ed h non sono posti consecutivamente, se ne può costruire un'altra in cui ciò avviene, senza aumentare il massimo numero di stack aperti, ed indipendentemente dal fatto che si ponga il pattern j esattamente prima del pattern h o il pattern h esattamente prima del pattern j. Ciò vuol dire che si può eliminare uno dei due nodi, si supponga il nodo h, e generare un problema ulteriormente ridotto, per poi aggiungere la chiusura dello stack h alla sequenza di chiusura ottima per il problema ulteriormente ridotto, in posizione immediatamente precedente alla chiusura dello stack j. Il fatto di porlo in posizione immediatamente precedente è fissato per convenzione.

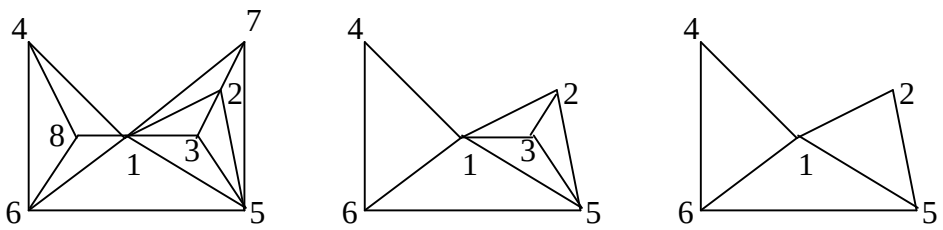
Tale criterio di equivalenza può essere riapplicato ogni volta che, a valle di una sua applicazione che determini l'eliminazione di alcuni nodi, si ottenga un grafo ulteriormente

ridotto. Su di esso si rivalutano gli insiemi N(j) ed è possibile che emergano ulteriori equivalenze. Si supponga, ad esempio, di trattare l'istanza del problema usata dagli autori, con i dati in Tabella 7.4, cioè con m = 14 pattern ed n = 8 gruppi di pezzi.

Pattern	**Insiemi di gruppi associati**	**Pattern**	**Insiemi di gruppi associati**
1	{1, 2, 3}	8	{2, 7}
2	{1, 4}	9	{3, 5}
3	{1, 5}	10	{4, 6}
4	{1, 6}	11	{4, 8}
5	{1, 7}	12	{5, 6}
6	{1, 8}	13	{5, 7}
7	{2, 5}	14	{6, 8}

Tabella 7.4 Dettagli di un esempio

Il MOSP graph associato è riportato in Figura 7.5. Nessuna riduzione avviene nella fase di pre-processing.



a) grafo iniziale b) prima riduzione c) grafo definitivo

Figura 7.5 Esempio di riduzione del MOSP graph

Nel grafo iniziale (Figura 7.5a), dal criterio di equivalenza, discende l'equivalenza sia dei nodi 2 e 7 che dei nodi 4 ed 8. Eliminando i nodi 7 ed 8, si ottiene il grafo ridotto in Figura 7.5b. Su tale grafo, risultano equivalenti i nodi 2 e 3 per cui, dall'eliminazione del nodo 3, si ricava il grafo finale in Figura 7.5c sul quale non emergono altre equivalenze.

Si conclude indicando che, durante l'esecuzione dell'algoritmo branch and bound, si sfrutta anche il criterio di dominanza descritto indipendentemente in Limeira (1998).

Per quanto riguarda i lower bounds globali utilizzati, il primo equivale al lower bound LB_1 introdotto già nel lavoro di Yanasse (1997b). Tale lower bound viene denotato con lb_p e, nei termini qui introdotti, può essere espresso come $\max\{|P_i| : i = 1, \ldots, m\}$.

Il secondo lower bound, che deriva dalla trasformazione nel MOSP graph, è denotato con lb_d e pari a $1 + \min\{g(q) : q$ è un nodo del MOSP graph $G\}$, dove $g(q)$ è il grado del nodo generico q, cioè il numero di archi in esso incidenti. Infatti, se per un dato nodo q, il relativo stack è quello scelto per primo nella sequenza di chiusura, se ne ricava che il numero di stack aperti a valle dell'inserimento, in testa alla sequenza, di tutti i pattern necessari per chiudere q è proprio ad $1 + g(q)$. Per il fatto che, necessariamente, uno degli stack deve essere scelto per primo, si ricava la validità del lower bound lb_d.

Il terzo lower bound è basato su un'analisi fatta nel lavoro di Yanasse, Becceneri e Soma (1999). Esso è denotato con lb_c ed è pari a $\max\{K_1 : K_1$ è un minore del grafo $G\}$.

Questi tre bounds vengono combinati attraverso una apposita procedura denominata "euristica di contrazione degli archi", i cui dettagli qui non vengono riportati.

7.2.5 Approcci euristici e meta-euristici per il MOSP

Le euristiche presentate nel seguito sono quelle proposte nei lavori di Yuen (1995) e di Becceneri, Yanasse e Soma (2004). Inoltre si descrivono tre tecniche proposte da Faggioli e Bentivoglio (1998), una euristica, una seconda meta-euristica, basata sull'approccio Tabu Search ed una terza, pure meta-euristica, semplificata. Per gli ultimi due lavori, la relativa (meta-)euristica è proposta per trovare velocemente una buona soluzione, come step iniziale di una procedura che poi prosegue con una ricerca esaustiva per risolvere ottimamente il problema. Il lavoro di Yuen e Richardson (1995) sfrutta in modo analogo le euristiche dei lavori di Yuen (1991, 1995), nell'ambito di una procedura esaustiva, di cui si è già fatto cenno nel paragrafo precedente. Si sottolinea che approcci meta-euristici sono proposti anche in Fink e Voβ (1999), che utilizzano la Simulated Annealing e la Tabu Search, mentre in Pezzella e De Giovanni (2007) si utilizza un approccio basato su Algoritmi Genetici (Goldberg, 1989).

Nel lavoro di Yuen (1995) sono presentate quattro euristiche, con l'intento di migliorare due euristiche proposte nel lavoro di Yuen (1991), e per questo chiamante Yuen3, Yuen4, Yuen5 e Yuen6. Tutte e quattro prevedono di costruire una soluzione per ogni possibile pattern iniziale della sequenza, selezionando poi di volta in volta il pattern successivo secondo una specifica regola.

L'euristica Yuen3 seleziona il pattern successivo basandosi su due parametri. Per ciascun potenziale pattern i, si valutano i due valori C_i ed N_i. Il valore C_i è pari al numero di gruppi di pezzi coinvolti nel pattern e tali che il relativo stack risulta già aperto. Il valore N_i è invece il numero di stack aggiuntivi che si aprirebbero se si aggiungesse quel pattern. I due valori vengono combinati nel valore $M_j = C_i - N_i$. Si sottolinea che le due euristiche del lavoro di Yuen (1991) utilizzavano solo il valore C_i.

L'euristica seleziona, di volta in volta, il pattern cui corrisponde il massimo valore M_i. In casi di parità, viene selezionato il pattern con il valore minimo di N_i. Se c'è ulteriore parità, viene selezionato il pattern elencato per primo nel problema, e questa regola finale vale anche per le altre euristiche, in caso di totale parità sulla base dei relativi criteri.

L'euristica Yuen4 utilizza invece il solo valore N_i, selezionando il pattern che corrisponde al suo minimo valore.

L'euristica Yuen5 ha la stessa regola iniziale dell'euristica Yuen3, basata cioè sul valore M_i ma, in caso di partirà, calcola il valore $M^*_{i,k}$ per tutti i possibili ulteriori pattern di indice k posti a valle di ogni pattern i per cui si ottenga il minimo valore M_i. Ciò vuol dire che, se ad esempio p e q sono gli unici due pattern con valore massimo $M_p = M_q$, si suppone prima di fissare il pattern p e si calcola il massimo valore $M^*_{p,k}$ corrispondente al massimo valore M_k che si avrebbe per la scelta del pattern k da porre dopo il pattern p, considerando dunque come possibile ulteriore pattern anche il pattern q. Analogamente, si suppone di fissare il pattern q e si calcola il massimo valore $M^*_{q,k}$ corrispondente al massimo valore M_k che si avrebbe per la scelta del pattern k da porre dopo il pattern q, considerando dunque come possibile ulteriore pattern anche il pattern p. Si seleziona infine, fra p e q, il pattern cui corrisponde il massimo valore fra $M^*_{p,k}$ e $M^*_{q,k}$.

L'euristica Yuen6 sviluppa ulteriormente l'idea dell'euristica Yuen5, considerando tutte le possibile coppie di pattern i, k, associando a ciascuna la somma del valore M_i e del valore M_k che, nelle altre euristiche, verrebbe calcolato per il pattern k a valle del posizionamento del pattern i. Tale somma è denotata con $M_{i,k}$. Fissata la coppia cui corrisponde il massimo valore $M_{i,k}$, si seleziona il primo pattern della coppia. Poiché, per costruzione, $M_{i,k} = M_{k,i}$, per qualunque coppia di pattern i e k, e data la regola su indicata, secondo cui, in caso di parità, si seleziona il pattern con indice minore, basta calcolare solo i valori $M_{i,k}$ con $i < k$.

L'euristica di Faggioli e Bentivoglio (1998) pure costruisce una sequenza per ogni possibile pattern iniziale. Nei termini della stessa simbologia utilizzata per la descrizione delle euristiche di Yuen, la selezione del pattern successivo si basa sulle seguenti tre regole, verificate in successione. Quando una regola determina la selezione di un unico pattern, esso viene selezionato senza verificare le regole successive. In caso di parità, la selezione, fra quelli che ottimizzano il criterio della regola corrente, avviene sulla base del criterio legato alla regola successiva. Se a valle delle tre regole vi è ancora più di un potenziale pattern, si opera come per l'euristiche di Yuen, scegliendo cioè il pattern elencato per prima nella definizione del problema.

La prima regola si basa sulla minimizzazione del valore N_i di nuovi stack aperti. Laddove uno o più pattern condividono un valore minimo pari a zero, tutti tali pattern vengono aggiunti alla sequenza, senza cioè verificare le regole successive.

La seconda regola si basa sulla massimizzazione del numero di stack che vengono chiusi a valle del pattern aggiunto. La terza regola, infine, seleziona il pattern che massimizzi il valore C_i.

Faggioli e Bentivoglio (1998) propongono inoltre due approcci meta-euristici. Il primo è un approccio Tabu Search, per il quale la soluzione di partenza è quella ottenuta con la suddetta euristica. Il secondo è una semplificazione del primo ed è basato su una procedura di ricerca locale generalizzata.

Tornando all'approccio Tabu Search di Faggioli e Bentivoglio, lo schema dell'algoritmo è riportato nel Box 7.2, come dato dagli autori.

Costruisci una soluzione iniziale Π_0 e poni $\Pi_c = \Pi^* = \Pi_0$
Setta $i = 0$
Repeat
  Trova la migliora soluzione Π_1 ottenibile a partire dalla soluzione corrente Π_c attraverso le *mosse* consentite, cioè quelle i cui *attributi* non sono proibiti secondo le *tabu restrictions*, e quelle che soddisfano un determinato criterio di aspirazione.
  Aggiorna $\Pi_c = \Pi_1$
  Se Π_c è migliore della migliore soluzione nota Π^*, aggiorna $\Pi^* = \Pi_c$
  Aggiorna le *tabu restrictions*
  $i = i + 1$
Until Criterio di stop

Box 7.2 Schema dell'approccio Tabu Search per il MOSP

Le *mosse* possibili sono quelle corrispondenti all'estrazione di un pattern dalla sequenza ed al suo reinserimento in una posizione diversa qualsiasi.

Come criterio di aspirazione per le mosse, viene utilizzato quello denominato *search-direction* (Glover e Laguna, 1993, 1997).

Relativamente alle *tabu restrictions*, si utilizza una matrice T di dimensioni $m \times m$, dove m è il numero dei pattern, con elementi T_{hk} pari ad 1 se il passaggio di un pattern dalla posizione h alla posizione k è correntemente proibito. Quando si opera una mossa che determina il passaggio di un pattern dalla posizione h alla posizione k, l'elemento della matrice T_{hk} è posto ad 1 e viene azzerato dopo un determinato numero p di iterazioni, pari al numero di pattern m per le prime $3m$ iterazioni. Per le iterazioni di indice i da $3m$ a $6m$, il valore di p cresce linearmente da m a $4m$, Per le iterazioni di indice i da $6m$ a $9m$, il valore di p aumenta ancora linearmente nello stesso range $[m, 4m]$, ripartendo dal valore m quando $i = 6m$. L'uguaglianza $i = 9m$ è utilizzata come criterio di stop.

La procedura di ricerca locale generalizzata proposta da Faggioli e Bentivoglio prevede di eseguire più volte una procedura di ricerca locale, partendo da soluzioni iniziali diverse. Una ricerca locale, a differenza che nell'approccio Tabu Search, accetta l'aggiornamento della soluzione corrente solo in caso di miglioramento, fermandosi quando nessun miglioramento è più possibile secondo l'insieme di mosse utilizzato.

L'algoritmo seguito è schematizzato nel Box 7.3, come dato dagli autori.

Setta $\Pi^* = \{1, \ldots, m\}$
For $i = 1$ **To** m **Do**
Begin
  Si pone $\pi_1 = i$
  For $k = 2$ **To** m **Do**
  Begin
    Si sceglie π_k secondo le tre regole dell'euristica greedy e secondo l'ordinamento in Π^*
    Si pone π_k non in coda ai precedenti k–1 pattern, ma nella posizione che minimizzi il corrispondente massimo numero di stack aperti
  End
  Si pone $\Pi_c = (\pi_1, \ldots, \pi_m)$
  Repeat
    Si costruisce la migliore soluzione Π_1 ottenibile a partire da Π_c attraverso un insieme di mosse predefinito
    Se Π_1 è migliore di Π_c, si aggiorna $\Pi_c = \Pi_1$
  Until Π_c non è stata aggiornata
  Se Π_c è migliore della migliore soluzione nota Π^*, aggiorna $\Pi^* = \Pi_c$
End

Box 7.3 Schema dell'approccio di ricerca locale generalizzato per il MOSP

La costruzione della soluzione $(\pi_1, \ldots, \pi_m)$ avviene basandosi sulla euristica greedy precedentemente spiegata, ma controllando quale sia la migliore posizione in cui inserire un pattern quando viene selezionato per l'inserimento.

Si noti che la soluzione Π^*, eventualmente aggiornata in ciascuno degli m cicli realizzati, influenza l'euristica greedy. Infatti, in casi di parità a valle dell'applicazione dei criteri associati alle tre regole dell'euristica, viene selezionato il pattern elencato per primo in Π^*. Ciò vuol dire che la procedura di ricerca locale interagisce con l'euristica.

La soluzione ottenuta con questo terzo approccio, è poi utilizza per avere a disposizione un upper bound globale nella esecuzione di un approccio esatto che verifica, implicitamente dove possibile, tutte le possibili sequenze di pattern. Tale approccio esatto, i cui dettagli non vengono forniti, rappresenta una evoluzione rispetto a quello di Yuen e Richardson (1995) ed è basata sull'aggiunta nuovi di criteri di chiusura dei nodi dell'albero di ricerca.

L'ultima procedura euristica qui descritta è quella proposta nel lavoro di Becceneri, Yanasse e Soma (2004). Come già indicato nel paragrafo precedente, essa viene proposta per ottenere rapidamente una prima buona soluzione, da utilizzare per ridurre il numero di nodi generati nella procedura branch and bound esatta.

Tale euristica prende il nome di *Minimal Cost Node*. Essa viene applicata sul MOSP graph associato all'istanza del problema, eventualmente ridotto secondo la strategia di pre-processing già descritta nel paragrafo precedente.

Relativamente all'approccio esatto discusso nel paragrafo 7.2.4.2, si sottolinea che l'euristica va applicata al grafo prima dei criteri di equivalenza, poiché l'applicazione di tali

criteri su grafi via via ridotti, implica regole di ordinamento di chiusura dei nodi al fine di rispettare la gerarchia con cui i nodi del grafo sono stati man mano eliminati per equivalenza.

Si ricorda che il MOSP graph è generato associando un nodo ad ogni possibile gruppo di pezzi ed aggiungendo un arco per ogni coppia di nodi i cui gruppi hanno pezzi su almeno uno stesso pattern.

Si fanno dunque le seguenti definizioni. L'insieme dei nodi del grafo è denotato con V, mentre E denota l'insieme degli archi. Per ciascun nodo p di V, $g(p)$ denota il suo grado, cioè il numero degli archi in esso incidenti. Si definisce inoltre come A la lista ordinata degli archi già traversati. L'insieme dei nodi di V in cui non incide nessuno degli archi della lista A è denotato con S_V e rappresenta l'insieme dei nodi non ancora chiusi. Esso viene ordinato per valori non decrescenti del grado ridotto dei nodi, considerando cioè solo gli archi non appartenenti ad A. L'insieme dei nodi etichettati e non ancora chiusi, cioè appartenenti ad S_V, ma tali che almeno un arco in essi incidenti appartenga alla lista degli archi traversati A, è denotata con *OPEN*.

Sulla base di tali definizioni, l'euristica può essere schematizzata come nel Box 7.4, così come indicato dagli autori.

Inizializza $S_V = V$, ordinato secondo i gradi dei nodi
$OPEN = \varnothing$
$A = \varnothing$
While $|A| \neq n$ **Do**
Begin
Sia q il primo nodo di S_V, cioè con grado minimo.
Si sceglie la coppia (p_1, p_2) di nodi di S_V, con $g(p_1) = g(q)$, tale da minimizzare $g(p_2)$
Si aggiungono p_1 ed p_2 ad *OPEN* (nel caso in cui non vi appartenessero già)
Si aggiunge l'arco (p_1, p_2) in coda ad A, aggiornando di conseguenza i gradi $g(p_1)$ e $g(p_2)$
Sia per p_1 che per p_2, se il grado è diventato 0, si cancella il nodo da *OPEN* e da S_V
Si costruisce l'insieme A' di archi non ancora traversati ed incidenti in due nodi già appartenenti ad *OPEN*
Gli archi di A' vengono aggiunti in coda ad A, in ordine qualsiasi, aggiornando di conseguenza i gradi dei nodi in cui essi incidono, eliminando da *OPEN* e da S_V quei nodi che dovessero ritrovarsi con grado nullo.
Si riordina l'insieme S_V secondo i valori aggiornati dei gradi dei suoi nodi
End

Box 7.4 Schema dell'euristica Minimal Cost Node

Il nome dell'euristica è dovuto al fatto che, ad ogni passo, viene selezionato il nodo collegato al minor numero di altri nodi ancora non chiusi. Se si seleziona un nodo p, allora, prima di poterlo chiudere, bisognerà necessariamente etichettare tutti i nodi, ancora non etichettati, collegati a p attraverso un arco. L'euristica cerca allora di selezionare un nodo, per la cui chiusura sia necessario eventualmente etichettare il minor numero di altri nodi. Dovendo però scegliere un arco, si seleziona un arco incidente in uno dei nodi a grado minimo, ma tale da minimizzare il grado dell'altro nodo in cui l'arco incide. A parità di grado di p_2, si seleziona l'arco con nodo p_1 di indice minore e, in caso di uguale nodo p_1, quello con nodo p_2 di indice minore.

Si noti che, una volta aggiunto l'arco (p_1, p_2) ad A, si verifica se esistono archi incidenti in due nodi già etichettati. Questo è dovuto al fatto che l'aggiunta di tali archi non può far aumentare il massimo numero di nodi etichettati e che, anzi, conviene inserirli subito, proprio perché non determinano l' etichettatura di alcun altro nodo e possono evitare di

chiudere con ritardo uno dei nodi in cui incidono. Per costruzione, basta guardare solo gli archi incidenti in p_1 oppure in p_2, dato che tutti gli altri nodi di *OPEN* vi appartenevano già al ciclo precedente.

Si riporta nel seguito un esempio di applicazione dell'euristica al grafo di Figura 7.5a, utilizzato nel paragrafo 7.2.4.2 e relativo all'istanza i cui dati (m = 14 pattern ed n = 8 gruppi di pezzi) sono già stati riportati in Tabella 7.4.

Inizialmente $S_V = V = \{1, 2, 3, 4, 5, 6, 7, 8\}$, mentre A ed *OPEN* sono vuoti. I gradi dei nodi sono, rispettivamente, $g(1) = 7$, $g(2) = 4$, $g(3) = 3$, $g(4) = 3$, $g(5) = 5$, $g(6) = 4$, $g(7) = 3$ e $g(8) = 3$, per cui i nodi a grado minimo, che vale 3, sono 3, 4, 7 ed 8. Viene selezionato l'arco (4, 8), ponendo cioè $p_1 = 4$ e $p_2 = 8$, così da minimizzare $g(p_2)$, con il vincolo $g(p_1) = 3$. La lista A diventa dunque pari a {(4,8)} e l'insieme *OPEN* diventa {4, 8}, mentre i gradi diventano: $g(1) = 7$, $g(2) = 4$, $g(3) = 3$, $g(4) = 2$, $g(5) = 5$, $g(6) = 4$, $g(7) = 3$ e $g(8) = 2$, cioè i gradi del grafo ridotto ottenuto per rimozione dell'arco (4, 8) di A, come mostrato in Figura 7.6a.

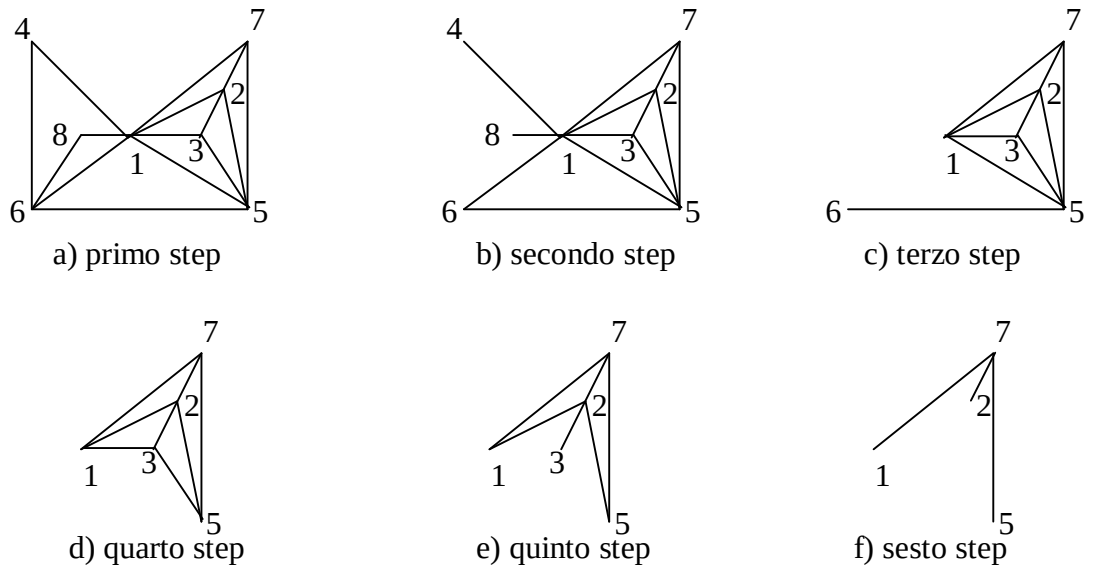


Figura 7.6 Esempio di applicazione dell'euristica Minimal Cost Node

A questo punto i nodi di grado minimo sono 4 ed 8, e l'arco selezionato è (4,6), dato che i possibili archi fra cui scegliere sono (4, 1), con $g(1) = 7$, (4, 6), con $g(6) = 4$, (8, 1) ed (8, 6). Una volta aggiunto l'arco (4, 6), il nodo 6 risulta etichettato, per cui l'arco (6, 8) risulta incidente in due nodi già etichettati. Quindi viene anch'esso aggiunto alla lista A, determinando così la riduzione del grafo come in Figura 7.6b, con A = {(4, 8), (4, 6), (6, 8)} ed *OPEN* = {4, 6, 8}.

I gradi dei nodi diventano dunque: $g(1) = 7$, $g(2) = 4$, $g(3) = 3$, $g(4) = 1$, $g(5) = 5$, $g(6) = 2$, $g(7) = 3$ e $g(8) = 1$, per i cui i nodi di grado minimo sono ancora 4 ed 8 ed i due potenziali archi sono (4, 1) ed (8, 1) che condividono lo stesso secondo nodo. Viene dunque selezionato l'arco (1, 4) che determina l'etichettatura del nodo 1. Di conseguenza, gli archi (1, 6) ed (1, 8) risultano incidenti in nodi già etichettati e vengono perciò aggiunti ad A, determinando la situazione schematizzata in Figura 7.6c, con A = {(4, 8), (4, 6), (6, 8), (1, 4), (1, 6), (1, 8)} ed *OPEN* = {1, 6} dato che i nodi 4 ed 8 diventano chiusi e vengono dunque eliminati anche dall'insieme S_V. I gradi dei nodi di S_V diventano: $g(1) = 4$, $g(2) = 4$, $g(3) = 3$, $g(5) = 5$, $g(6) = 1$ e $g(7) = 3$.

A questo punto il nodo di grado minimo è il 6 e l'unico arco possibile da scegliere è (6, 5), che determina l'etichettatura del nodo 5. Dunque anche l'arco (1, 5) viene aggiunto ad

A, poiché 1 e 5 risultano entrambi etichettati, mentre il nodo 6 diventa chiuso e quindi viene tolto dall'insieme S_V. Il grafo ridotto che ne consegue è quello di Figura 7.6d, con *OPEN* uguale ad {1, 5} ed *A* uguale a {(4, 8), (4, 6), (6, 8), (1, 4), (1, 6), (1, 8), (5, 6), (1, 5)}.

I gradi dei nodi di S_V diventano: $g(1) = 3$, $g(2) = 4$, $g(3) = 3$, $g(5) = 3$ e $g(7) = 3$. L'arco selezionato è dunque (1, 3) che congiunge due nodi a grado minimo, determinando così l'etichettatura dal nodo 3 e la conseguente aggiunta ad *A* anche dell'arco (3, 5). Ne consegue l'ulteriore riduzione del grafo rappresentata in Figura 7.6e, con *OPEN* = {1, 3 ,5} e con la lista *A* pari a {(4, 8), (4, 6), (6, 8), (1, 4), (1, 6), (1, 8), (5, 6), (1, 5), (1, 3), (3, 5)}.

I gradi dei nodi di S_V diventano dunque $g(1) = 2$, $g(2) = 4$, $g(3) = 1$, $g(5) = 2$ e $g(7) = 3$, per cui viene selezionato l'unico arco possibile (3, 2), con etichettatura del nodo 2, conseguente aggiunta degli ulteriori archi (1, 2) e (2, 5) e chiusura del nodo 3. La situazione diventa quella di Figura 7.6f, con *OPEN* = {1, 2, 5} ed *A* = {(4, 8), (4, 6), (6, 8), (1, 4), (1, 6), (1, 8), (5, 6), (1, 5), (1, 3), (3, 5), (2, 3), (1, 2), (2, 5)}.

I gradi dei doni dell'insieme S_V = {1, 2, 5, 7} diventano: $g(1) = 1$, $g(2) = 1$, $g(5) = 1$ e $g(7) = 3$. Quindi i nodi a grado minimo sono 1, 2 e 5 ed i relativi archi sono (1, 7), (2, 7) e (5, 7), che condividono il secondo nodo. Viene quindi selezionato l'arco (1, 7), che determina l' etichettatura del nodo 7 e la chiusura del nodo 1. Di conseguenza i due altri archi (2, 7) e (5, 7) risultano incidenti in nodi già etichettati e vengono aggiunti in coda alla lista *A*, determinando la chiusura dei nodi 2, 5 e 7 e la fine della procedura con la lista definitiva *A* pari a {(4, 8), (4, 6), (6, 8), (1, 4), (1, 6), (1, 8), (5, 6), (1, 5), (1, 3), (3, 5), (2, 3), (1, 2), (2, 5), (1, 7), (2, 7), (5, 7)}. A tale lista corrisponde, secondo la regola di trasformazione in una sequenza di pattern, spiegata nel paragrafo 7.2.4.1, la sequenza di pattern Π = (11, 10, 14, 2, 4, 6, 12, 3, 9, 1, 7, 5, 8, 13), con un massimo numero di stack aperti pari a 4, dopo la processazione dei pattern 2 in quarta posizione, 1 in decima posizione e 5 in dodicesima posizione.

In conclusione del paragrafo, si sottolinea che pure nel lavoro di Yanasse (1997b) viene proposta una euristica. Si tratta di applicare gli step iniziali della procedura branch and bound proposta nello stesso lavoro e descritta nel paragrafo 7.2.3, fino alla costruzione della prima soluzione, cioè fino al primo backtracking.

Capitolo 8

Il problema combinato di

Cutting e Sequencing

L'approccio metodologico proposto nel precedente Capitolo per il problema del Sequencing suppone di avere già a disposizione l'insieme dei pattern generati come soluzione per il problema di Cutting nel caso multi-stock.

Tuttavia, la soluzione del problema di Cutting mira alla minimizzazione dello scarto totale, nel caso di stock differenti, ovvero alla minimizzazione del numero di stock utilizzati, nel caso di stock identici.

Ciò vuol dire che, seguendo tale approccio, i pattern vengono generati senza tener conto degli obiettivi relativi al problema del Sequencing. Come evidenziato nel sesto Capitolo, è invece auspicabile risolvere i problemi di Cutting e di Sequencing in modo combinato. Nel seguito si discutono alcune possibili modalità attraverso cui realizzare tale obiettivo.

Il primo paragrafo descrive gli approcci proposti in lavori di letteratura relativi a tale questione, tra l'altro affrontata solo di recente. Il secondo paragrafo presenta due approcci euristici originali, uno basato sull'euristica presentata nel terzo Capitolo ed uno basato sull'idea innovativa di sovragenerazione di pattern, che può essere utilizzata in alternativa o in combinazione con quelle già presenti in letteratura.

8.1 Approcci di letteratura

L'approccio più semplice che si possa pensare, già discusso nei precedenti Capitoli, consiste nel risolvere il problema di Cutting con l'obiettivo di minimizzare lo scarto, per poi utilizzare i cutting pattern, così generati, come dati di ingresso per il problema di Sequencing.

Tale metodo per la soluzione del problema combinato di Cutting e Sequencing è noto come two-stage approach (Dyson e Gregory, 1974; Madsen, 1988). Metodi che affrontino i problemi in modo combinato sono abbastanza recenti in letteratura.

Si coglie l'occasione per dire che in letteratura si affronta anche un ulteriore problema, diverso dai tre identificati nel Capitolo precedente, cioè il MOSP, il MORP e l'MDP. Si tratta del problema denominato PMP nel lavoro di Vanderbeck (2000). Esso consiste nella minimizzazione del numero di differenti cutting pattern utilizzati. Sebbene non si tratti di un vero e proprio problema di Sequencing, dato che non c'è alcun ordinamento da cercare, esso ha interessanti risvolti applicativi.

Il motivo pratico per cui tale problema è importante consiste nel fatto che, in alcuni settori industriali, è necessario un tempo di setup quando si alteri lo schema di taglio, cioè il tipo di cutting pattern. Si parla infatti di minimizzazione dei setup. Questo aspetto è particolarmente presente in settori con problemi di taglio mono-dimensionale. Supponendo di avere stock mono-dimensionali di lunghezza L, uno schema di taglio è identificato dalle lunghezze $l_1, \ldots, l_k$ in cui scindere lo stock, supponendo cioè che valga $\sum_{i=1}^{k} l_k = L$.

Può avvenire che, per realizzare un tale schema di taglio, è necessario porre manualmente $k - 1$ coltelli sul macchinario, il primo a distanza l_1 rispetto ad un estremo dello stock, il secondo a distanza l_2 rispetto al primo e così via fino all'ultimo. Ciò implica tempi di setup e dunque maggiori costi di esecuzione.

Nel caso del taglio del vetro, gli automatismi presenti sui macchinari rendono meno interessante questo problema, per cui nel seguito esso non verrà analizzato. Si indica soltanto che la maggior parte dei lavori di letteratura che affrontano il problema, fra cui McDiarmind (1999), Foerster e Wäscher (2000), Belov (2003), Yanasse e Limeira (2006) e Belov e Scheithauer (2007), oltre al già citato lavoro di Vanderbeck, lo fanno in relazione al caso mono-dimensionale.

Va comunque sottolineato che, anche nel caso del taglio del vetro, avere pochi tipi di schemi di taglio rappresenta un vantaggio in termini di velocizzazione delle operazioni di apertura dei tagli. Tuttavia l'obiettivo di minimizzare il numero di differenti schemi di taglio può implicare dei risultati molto scarsi nei termini degli obiettivi dei problemi di Sequencing, in quanto con molta probabilità ciascuno schema di taglio coinvolge un elevato numero di gruppi di pezzi, determinando così sia un elevato numero di stack aperti, sia un elevato order spread medio, a causa dell'elevata frequenza associata a ciascun cutting pattern.

Il lavoro di Yanasse e Pinto Lamosa (2007) sottolinea che approcci combinati per Cutting e MOSP problem sono presenti in scarso numero in letteratura e, nell'ambito di una significativa bibliografia sui lavori relativi al Cutting ed al Packing, identifica i soli lavori di Armbruster (2002) e di Pileggi (2002). Questo secondo lavoro viene ripreso in Pileggi, Morabito ed Arenales (2006). Anche Belov (2003) ha però proposto una procedura per la combinazione dei due problemi.

I suddetti lavori affrontano una combinazione fra Cutting e MOSP usando come obiettivo la minimizzazione dello scarto, e come vincolo una limitazione sul massimo numero di stack aperti, ad eccezione di Belov, come si spiegherà in seguito. In effetti pare essere la modalità di approccio che meglio si applica alla realtà. Infatti, in termini di MOSP, non è realmente necessario minimizzare il più possibile il massimo numero di stack aperti, bensì

tenerlo opportunamente limitato. Ricordando la schematizzazione di un'isola di lavoro per il taglio del vetro, riportata alla fine del sesto Capitolo, appare infatti chiaro che l'importanza è evitare un ingombro eccessivo, e non c'è necessità di una sua riduzione spinta.

In alcune realtà industriali, viene progettato lo spazio nei pressi dell'isola di lavoro, determinando quindi un valore limite c al numero di stack che possono essere aperti. Se dunque si sfora tale numero, sono necessari degli switch fra i contenitori relativi agli stack, sempre nell'ipotesi che ciascun pattern non coinvolga più di c gruppi di pezzi.

Il lavori di Armbruster e di Belov sono indirizzati al caso mono-dimensionale, mentre i lavori di Pileggi e di Yanasse e Pinto Lamosa sono indirizzati sia al caso mono-dimensionale che a quello bidimensionale, sebbene l'ultimo lavoro è stato testato solo su casi mono-dimensionali, e tra l'altro nel caso di stock identici.

Un altro lavoro recente è quello di Rinaldi e Franz (2007), che però utilizza stock con una dimensione infinita, cioè strip, e si pone in ipotesi restrittive sul massimo numero di gruppi aperti. I relativi risultati trovano interessante applicazione nell'ambito della produzione delle lastre di vetro. Infatti il processo di generazione prevede, come accennato nel secondo Capitolo, il raffreddamento del vetro fuso. I macchinari che generano le lastre partono da un flusso continuo di vetro fuso, assimilabile di fatto ad una strip. In questo caso i pezzi sono, di fatto, quelle che poi saranno le lastre di partenza nelle industrie di trasformazione del vetro, ed il limite sul numero di differenti tipi di lastre-pezzo è legato a vincoli di gestione della produzione.

Nel seguito si dà una descrizione dei lavori di Armbruster, di Pileggi, di Belov e di Yanasse e Pinto Lamosa, soprattutto con l'obiettivo di sottolineare le idee alla base delle relative metodologie.

8.1.1 L'approccio euristico di Armbruster

Si sottolinea innanzitutto che il problema affrontato nel lavoro di Armbruster (2002) è leggermente diverso da una pura combinazione di Cutting e MOSP.

In particolare l'autore fa riferimento ad un problema mono-dimensionale che sorge nell'industria dell'acciaio. I macchinari di riferimento sono fatti in modo da avere un certo numero di *compartimenti* per lo stazionamento dei gruppi di pezzi. Ciascun gruppo si suppone costituito da un solo tipo di pezzi, identificato cioè dalla lunghezza dei pezzi che vi appartengono. Il numero di compartimenti è limitato. Dunque, in termini di combinazione fra Cutting e MOSP, l'obiettivo consiste nella minimizzazione dello sfrido, cioè dello scarto, e c'è un limite massimo al numero di stack aperti.

La particolarità del problema consiste nel fatto che i compartimenti sono organizzati in gruppi ed i compartimenti dello stesso gruppo sono in fila l'uno dietro l'altro, così da costituire una *via*. Tutti i compartimenti sono di uguale lunghezza l_c e tutte le vie hanno lo stesso numero di compartimenti. Dunque, se c è il numero totale di compartimenti, e b è il numero totale di vie, allora ci sono c / b compartimenti per ogni via. Se i pezzi di un gruppo hanno lunghezza l_j maggiore di l_c, allora per lo stazionamento dei pezzi di quel gruppo, è necessario utilizzare un numero di compartimenti consecutivi, in una stessa via, pari ad $\lceil l_j / l_c \rceil$ cioè al rapporto l_j / l_c , fra lunghezza dei pezzi e lunghezza dei compartimenti, approssimato per eccesso.

Se, per ogni gruppo, la relativa lunghezza non è maggiore di l_c, allora il problema perde la sua particolarità. Comunque, per tenerne conto, nel modello proposto da Armbruster si rappresenta l'effetto di una sequenza di pattern attraverso w matrici, Y^1, …, Y^k, …, Y^w, supponendo che w sia il numero di pattern utilizzati. Ciascuna matrice Y^k è di dimensioni (c / b) × b, con una via associata ad ogni colonna, mentre ad ogni riga è associata una posizione relativa di compartimento all'intero di ciascuna via. Dunque l'elemento $y^k_{p,q}$

identifica il gruppo associato al *p*-mo compartimento della *q*-ma via, a valle dell'esecuzione del *k*-mo pattern, la cui posizione nella sequenza è denotata con $f(k)$.

Supposta nota la sequenza di pattern, ciascuna matrice Y^k è costruita in due passi. Al primo passo si eredita la situazione a valle dell'esecuzione del pattern precedente. Dunque, la matrice $Y^{\underline{k}}$, con $f(\underline{k}) = 1$, parte da valori tutti nulli, dato che prima dell'esecuzione del primo pattern, nessun gruppo è associato a nessuno compartimento. Invece, per tutti gli altri valori di *k*, cioè tali che $f(k) > 1$, la matrice è inizialmente pari alla matrice $Y^{g(f(k)-1)}$, dove *g* è la funzione inversa di *f*.

A partire dalla situazione ereditata, la matrice viene aggiornata associando uno o più compartimenti consecutivi, secondo i rapporti l_j / l_c, a tutti i gruppi con pezzi sul *k*-mo pattern ed a cui non risultava ancora nessun compartimento associato. L'ipotesi è che tale assegnazione sia possibile e che, in caso di rapporti $l_j / l_c > 1$, si scelga un'assegnazione che renda poi possibili le assegnazioni per gli altri pattern della sequenza. Dopo questa fase di assegnazione, si liberano i compartimenti, azzerando cioè i relativi valori $y^k_{p,q}$, che erano associati a gruppi di pezzi ormai tutti estratti.

In generale, dato un pattern, e supponendo di partire da una matrice nulla, quel pattern è detto *tecnologicamente fattibile* se è possibile trovare una assegnazione dei compartimenti per tutti i gruppi di pezzi coinvolti in quel pattern. Si sottolinea che se tutti i rapporti l_j / l_c sono non maggiori di uno, allora la fattibilità tecnologica equivale alla condizione secondo cui il pattern è associato a non più di *c* gruppi diversi. In caso contrario, la verifica della fattibilità tecnologica rappresenta un problema di bin packing, in cui ogni via rappresenta un bin e ad ogni gruppo corrisponde un impegno di spazio pari al relativo valore di $\lceil l_j / l_c \rceil$. E' quindi necessario inserire nei bin un pezzo per ciascun dei gruppi coinvolti, per cui non si tratta di un problema di ottimizzazione ma, appunto, di fattibilità.

Per una sequenza di *w* pattern, identificata dai pattern e dalla funzione *f* che identifica la posizione di ciascun pattern, si parla di fattibilità tecnologica quando esiste una sequenza di matrici Y^1, …, Y^k, …, Y^w, ottenibile con le suddette regole. Nel caso semplificato in cui i rapporti l_j / l_c siano tutti non maggiori di uno, ciò equivale a dire che il massimo numero di stack aperti non deve superare *c*, come in un normale problema MOSP.

Il modello proposto da Armbruster si divide in due parti. La prima parte consiste in un modello lineare intero classico per il Cutting Stock. Si supponga, come già in precedenza, di denotare con l_j la lunghezza dei pezzi del *j*-mo gruppo. Sia inoltre L_h la lunghezza dell'*h*-mo stock, supposto con disponibilità massima pari al valore S_h.

Ogni possibile pattern è identificato dall'insieme di pezzi associati nonché dallo stock da cui si supponga di estrarli. Un generico pattern è quindi rappresentato da un vettore di $n+s$ elementi, se *n* è il numero di gruppi di pezzi, ciascuno con una domanda associata denotata con d_j, ed *s* è il numero di differenti stock disponibili.

Il vettore che identifica il *k*-mo pattern è $p_k = (p_{1k}, \ldots, p_{nk}, 0, \ldots, 1, 0, \ldots 0)$, dove il generico elemento p_{kj}, con $j = 1, \ldots, n$, rappresenta il numero di pezzi del *j*-mo gruppo, mentre uno ed uno soltanto degli altri *s* elementi è pari ad 1 ed identifica il tipo di stock associato, cioè di lunghezza $L_{h(k)}$, dove $h(k)$ è appunto l'unico l'indice, di valore compreso fra 1 ed *s*, tale che valga l'uguaglianza $p_{[n+h(k)]k} = 1$.

Lo scarto associato al pattern rappresentato dal vettore p_k è denotato con r_k ed è ottenuto con l'espressione $r_k = L_{h(k)} - \sum_{j=1}^{n} l_j p_{jk}$.

Si supponga ora di denotare con *w* il numero di tutti i possibili cutting pattern. Il modello utilizzato è dunque il seguente.

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{w} r_k x_k + \sum_{j=1}^{n} l_j \left(\sum_{k=1}^{w} p_{jk} x_k - d_j \right) & & \\
s.t. \quad & \sum_{k=1}^{w} p_{jk} x_k \geq d_j & j = 1, \ldots, n & \quad \text{(a)} \\
& \sum_{k=1}^{w} p_{(n+h)k} x_k \leq S_h & h = 1, \ldots, s & \quad \text{(b)} \\
& x_k \in \mathbb{N}_0 & k = 1, \ldots, w & \quad \text{(c)}
\end{aligned}
\tag{8.1}
$$

Le variabili x_k identificano, per ciascun valore di k, il numero di volte (o frequenza) in cui il k-mo pattern è utilizzato. Affinché si consideri il problema totale, bisogna aggiungere il vincolo secondo cui l'insieme dei pattern utilizzati $\{p_k : x_k > 0\}$ è tecnologicamente fattibile, nel senso che esiste una loro sequenza tecnologicamente fattibile.

L'approccio risolutivo si divide in tre fasi. La prima fase prevede di risolvere il problema (8.1), relativo cioè solo al Cutting Stock, con la tecnica proposta in Gilmore e Gomory (1961, 1963), che per primi hanno proposto un modello lineare equivalente a questo. Differenti modelli sono proposti in Kantorovich (1960), Dyckhoff (1981), Vanderbeck (1999), Valério de Carvalho (1998, 2002). La tecnica di Gilmore e Gomory parte dal presupposto che può essere troppo oneroso generare in partenza tutti i possibili vettori p_k, per cui si preferisce risolvere il problema rilassando l'interezza sulle variabili x_k e generando esplicitamente solo i pattern che di volta in volta corrispondono alle variabili x_k da portare in base nel metodo del simplesso. Oltretutto, con il metodo del simplesso rivisitato, è possibile sviluppare i calcoli conoscendo solo i coefficienti relativi a tali pattern. Ne consegue però che, ad ogni passo del simplesso, la selezione del pattern da far entrare in base avviene risolvendo un apposito problema dello zaino mono-dimensionale.

Per la soluzione di tale problema dello zaino, si potrebbero utilizzare vari metodi, come quelli di Gimore e Gomory (1963), di Martello e Toth (1990), Martello (1994a), Pisinger (2000) e Soma e Toth (2002).

Tale problema viene però risolto, nel lavoro di Armbruster, prima in modo approssimato con una tecnica greedy. Se tale euristica non genera un cutting pattern che può entrare in base nel metodo del simplesso, allora si utilizza una apposita tecnica di programmazione dinamica.

Una volta ottenuta la soluzione ottima del problema rilassato, è necessario costruire una soluzione del problema originario. A tal fine, viene utilizzata una variante dell'euristica proposta da Roodman (1986), che rientra nella classe dei cosiddetti Approcci Compositi per questa trasformazione, secondo la tassonomia data in Wäscher e Gau (1996). La differenza rispetto a tale euristica risiede nell'esclusione dei pattern non tecnologicamente fattibili. Per la verifica di fattibilità tecnologica dei singoli pattern, data l'equivalenza con un problema di bin packing, con bins tutti uguali, come evidenziato in precedenza, viene utilizzata una euristica simile alla First-Fit Decreasing, di cui si è parlato nel primo Capitolo.

Un semplice metodo di passaggio ad una soluzione intera potrebbe essere quello proposto da Gilmore e Gilmore (1961), che consiste semplicemente nell'approssimare per eccesso tutti i valori delle variabili x_k. Altre tecniche sono proposte e studiate in Haessler (1980), Stadtler (1990), Mukhacheva e Zalgaller (1993), Vance, Barnhart, Johnson e Nemhauser (1994), Wäscher e Gau (1996), Scheithauer e Terno (1996, 1997), Nitsche, Scheithauer e Terno (1998), Rietz e Scheithauer (2002) e Poldi ed Arenales (2005), mentre tecniche che prevedono un differente modo di risolvere il problema (8.1) si trovano in Nitsche, Scheithauer e Terno (1999), Scheithauer, Terno, Müller e Belov (2001) e Belov e Scheithauer (2002), relativamente ad approcci esatti, ed in Hinxman (1980), Farley (1983b),

Dyckhoff e Finke (1992), Holthaus (2002) e Morabito e Belluzzo (2007), relativamente ad approcci approssimati. Inoltre, nel settimo paragrafo del terzo Capitolo è stata presentata una modalità simile a quella di Hinxman, ma con alterazione dei profitti associati ai pezzi per meglio pilotare la generazione di buoni insiemi di soluzioni.

Una volta ottenuto un insieme di pattern, la terza parte della procedura consiste nel verificare l'esistenza di una sequenza di tali pattern che sia tecnologicamente fattibile. L'idea su cui si basa tale terza parte, che rappresenta il cuore della combinazione fra il problema di Cutting ed il MOSP, consiste nell'eventuale presenza di backtracking, per la sostituzione di pattern con altri nuovi pattern. Quali siano eventualmente, in caso di esito negativo, i pattern da sostituire, viene scelto in base ai punti critici verificatisi durante la ricerca di una sequenza tecnologicamente fattibile.

In particolare il problema di trovare una sequenza buona è interpretato attraverso un grafo equivalente al MOSP graph introdotto nel precedente Capitolo, con i nodi associati ai gruppi ed archi non orientati fra ogni coppia di nodi tali che almeno un pattern coinvolge pezzi dei due gruppi associati.

Un elemento aggiuntivo, funzionale per l'algoritmo, consiste nell'associazione ad ogni arco di un costo che rappresenta lo scarto aggiuntivo che si otterrebbe se l'arco venisse proibito, cioè se si alterassero i pattern sulla base della proibizione di avere pattern che coinvolgano i gruppi associati ai due nodi in cui incide l'arco. In particolare, lo scarto aggiuntivo viene valutato eliminando i pattern che coinvolgono i pezzi dei due gruppi, reinserendo tutti i pezzi così liberati, ove possibile, in pattern non eliminati che abbiano sufficiente spazio per almeno uno di tali pezzi, e generando nuovi pattern che evitino di condividere pezzi di quei due gruppi. I dettagli delle modalità con cui questa rigenerazione avviene, nonché di altri elementi implementativi dell'euristica, sono forniti in Ambruster (2000). Qui si accenna al fatto che ci si basa sul modello (8.1), utilizzando solo la domanda scaturita dalla suddetta liberazione dei pezzi ed introducendo appositi vincoli che evitino di avere pezzi dei due gruppi su uno stesso pattern.

Tale procedura è ripetuta per ogni possibile arco, determinando così l'associazione di un costo, detto di *repacking*. Se per qualche arco vi dovesse essere un costo negativo, associato ad un miglioramento dello scarto totale, l'eliminazione dell'arco cui è associato il massimo miglioramento, viene realizzata, determinando così l'alterazione dell'insieme di pattern, la rigenerazione del grafo ed il nuovo calcolo dei costi di *repacking*, eventualmente iterando questo procedimento, finché non ci si trovi con un grafo avente tutti archi con costo di *repacking* non negativo. Armbruster sottolinea che tale elemento algoritmico è originale poiché Roodmands non lo utilizzava.

A questo punto, la ricerca di una sequenza tecnologicamente fattibile avviene con una procedura ad albero. Viene però ricercata una sequenza di nodi, cioè una sequenza di apertura degli stack, in modo analogo alle sequenze di chiusura ricercate nel lavoro di Yanasse (1997b) descritto nel precedente Capitolo. Si suppone in prima battuta che ogni gruppo necessiti di un solo compartimento, cioè che tutti i rapporti l_j / l_c sono non maggiori di uno. Questo significa che è ininfluente quale sia il compartimento cui si sceglie di associare un gruppo di pezzi.

Man mano che si aggiungono nodi non ancora aggiunti nei precedenti livelli dell'albero, si considerano come inseriti nella sequenza di pattern quei pattern per i quali tutti i nodi relativi ai gruppi coinvolti nel pattern risultano inseriti. Tali pattern vengono considerati come processati e se, per un gruppo, risultano processati tutti i pattern da cui si estraggono suoi pezzi, allora il compartimento ad esso assegnato viene liberato e risulta disponibile per l'associazione ad un nuovo gruppo non ancora aperto.

Si noti che è come se il gruppo scelto fosse aperto prima della reale processazione del primo pattern che lo coinvolge, ovvero da cui si estraggono suoi pezzi. Si tratta in pratica di

una sorta di pre-occupazione del relativo compartimento, in modo complementare rispetto a quanto accade nel suddetto algoritmo di Yanasse.

Se si arriva ad una foglia dell'albero, in cui cioè tutti i nodi siano stati aperti senza mai trovare tutti i compartimenti occupati, allora ciò vuol dire che è stata trovata una sequenza tecnologicamente fattibile. Quando invece si arriva all'occupazione di tutti i compartimenti senza che nessuno venga liberato per l'eventuale processazione di qualche pattern, allora ci si trova nell'impossibilità di andare avanti, poiché non è possibile aggiungere altri nodi, che sicuramente ci sono. Infatti, in caso contrario, tutti i pattern ancora sarebbero stati processati, con conseguente liberazione di tutti i compartimenti.

Tuttavia si registrano informazioni e le si associano all'ultimo nodo aggiunto che ha determinato il blocco. In particolare si registra quali fossero i nodi non ancora aggiunti ed inoltre direttamente collegati, tramite un arco, a quello che ha determinato il blocco. Si regista inoltre quale fosse il numero di nodi già aggiunti, compreso quello che ha determinato il blocco. Tale secondo parametro è una stima di quanto vicini si fosse alla costruzione di una sequenza tecnologicamente fattibile.

Se, a valle di tutta la ricerca ad albero, non è stata trovata alcuna sequenza buona, allora si sceglie il gruppo associato al nodo che ha procurato un blocco ed a cui corrisponde il minimo valore della somma dei costi di repacking associati agli archi in esso incidenti e che lo collegano con nodi che non erano stati ancora aperti al momento del blocco. Tali nodi vengono, come detto prima, registrati, appunto per permettere di realizzare questa selezione. Il valore di tale somma è una stima di quanto scarto si aggiungerebbe nel caso in cui si proibissero tutti quegli archi, nel senso spiegato sopra.

Se per due nodi dovesse esserci uno stesso valore, si sceglie il nodo cui corrisponde il numero minimo di tali archi. In caso di ulteriore parità si sceglie il nodo associato alla massima sequenza parziale, ovvero al livello di profondità maggiore nell'albero al momento del blocco.

Vengono dunque proibiti i suddetti archi relativi al nodo selezionato e vengono eliminati i pattern che condividono pezzi del nodo eliminato e di almeno uno dei nodi non ancora presi al momento del relativo blocco, ed in cui incideva uno degli archi ora proibiti.

Tali pattern vengono sostituiti con nuovi pattern che evitino le condivisioni di pezzi di coppie di gruppi associate agli archi ora proibiti. Si sottolinea che, a differenza di quanto avviene nella prima parte dell'algoritmo in caso di costi di repacking negativi, nessun pezzo viene inserito, anche se fosse possibile, in uno dei pattern non eliminati, dato che in questa fase il problema è trovare una sequenza tecnologicamente fattibile e tale operazione potrebbe complicare le cose a causa dell'aggiunta di nuovi archi.

Come critica a tale lavoro si sottolinea che, tutto sommato, si potrebbe fare un'eccezione a tale regola, appunto quando un tale inserimento non determina l'aggiunta di alcun nuovo arco al grafo. Oppure, a valle della generazione dei nuovi pattern, si potrebbe verificare se si possono estrarre pezzi dal pattern con scarto maggiore ed inserirli in un qualunque altro pattern non eliminato prima della rigenerazione, a patto di non aggiungere archi non già aggiunti a causa della rigenerazione.

L'unico punto lasciato in sospeso riguarda il caso in cui qualche rapporto l_j / l_c sia maggiore di uno. In tal caso l'autore indica che, in caso di backtracking, bisogna provare, dove possibile, l'associazione di un gruppo a tutte le possibili k-ple di compartimenti consecutivi, se k è il corrispondente valore di $\lceil l_j / l_c \rceil$. Altra critica risulta nel fatto che Armbruster non identifica alcuna strategia di anti-simmetria relativamente a tale aspetto, in relazione all'associato modello di bin packing.

8.1.2 Gli approcci euristici di Pileggi

I lavori di Pileggi (2002) e di Pileggi, Morabito e Arenales (2006) presentano tre euristiche per l'approccio combinato ai problemi di Cutting e Sequencing, considerando, come già detto sopra, l'obiettivo di minimizzare lo scarto ed un vincolo sul massimo numero di stack aperti. Si denota nel seguito con t il massimo numero di stack aperti permesso.

Il punto di partenza è ancora il modello di Gilmore e Gomory (1961), già descritto nel precedente paragrafo, con la semplificazione che l'interesse è rivolto al caso in cui non vi siano limiti al numero di stock, per cui i vincoli (8.1b) vengono omessi.

Inoltre, non si ingloba in r_k il costo totale associato a ciascun pattern. Si noti anche che la funzione obiettivo del modello (8.1) può essere riscritta sfruttando le uguaglianze

$$\sum_{k=1}^{w} r_k x_k + \sum_{j=1}^{n} l_j \left(\sum_{k=1}^{w} p_{jk} x_k - d_j \right) = \sum_{k=1}^{w} \left(r_k + \sum_{j=1}^{n} l_j p_{jk} \right) x_k - \sum_{j=1}^{n} l_j d_j = \sum_{k=1}^{w} L_{h(k)} x_k - \sum_{j=1}^{n} l_j d_j \ .$$

Dunque, a meno della costante $\sum_{j=1}^{n} l_j d_j$, si può minimizzare la somma delle lunghezze degli stock utilizzati $\sum_{k=1}^{w} L_{h(k)} x_k$. Se, inoltre, gli stock sono tutti uguali, ciò equivale a minimizzare il numero di stock utilizzati $\sum_{k=1}^{w} x_k$. Nel modello di Pileggi, i coefficienti r_k inglobano tutto il costo associato a ciascuno stock, che più in generale può non essere lo scarto ed avere quindi un significato diverso. Se ne ricava il seguente modello.

$$\begin{aligned} \min \quad & \sum_{k=1}^{w} r_k x_k & & \\ s.t. \quad & \sum_{k=1}^{w} p_{jk} x_k \geq d_j \quad j = 1, \ ..., \ n & \text{(a)} & \qquad (8.2) \\ & x_k \in \mathbb{N}_0 \qquad k = 1, \ ..., \ w & \text{(b)} & \end{aligned}$$

Vengono però presi in considerazione sia il caso mono-dimensionale che quello bidimensionale, considerando anche il caso specifico di cutting pattern bidimensionale con due soli stage, secondo quanto spiegato nel primo Capitolo.

Il passaggio alla bidimensionalità ha effetti sulle modalità di generazione dei cutting pattern e sull'espressione del costo r_k, secondo la relativa applicazione.

Anche Pileggi segue la strada tracciata da Gilmore e Gomory, basata su un metodo di *column generation*, cioè di generazione dei pattern solo se e quando servono, come già descritto nel precedente paragrafo, nell'ambito della risoluzione del problema rilassato, cioè senza i vincoli (8.2b) di interezza sulle variabili x_k. Tale metodo viene sfruttato sia per la prima che per la terza euristica, mentre la seconda euristica risolve direttamente il problema con un metodo che tiene conto del vincolo di Sequencing.

A differenza che nel lavoro di Armbruster, la generazione dei pattern da far entrare in base, nel caso mono-dimensionale, è realizzata con l'algoritmo lessicografico presentato in Gilmore e Gomory (1963), mentre il passaggio da una soluzione del problema rilassato ad una soluzione del problema non rilassato avviene attraverso una approssimazione per eccesso dei valori delle variabili x_k. Inoltre, nel seguito si denota con $\underline{r}$ il costo minimo ottenibile risolvendo il problema (8.2) rilassato.

Nel caso di problema bidimensionale, con vincolo di un massimo numero di stage pari a due, i pattern da far entrare in base vengono generati con la procedura di Gilmore e Gomory (1965). Infine, nel caso di problema bidimensionale con numero illimitato di stage, viene utilizzato l'algoritmo AND/OR-graph di Morabito ed Arenales (1996).

Per tutte e tre le euristiche, nella seconda fase si cerca, attraverso l'euristica Yuen3 già descritta nel precedente Capitolo, una opportuna sequenza dell'insieme di pattern generati nella prima fase, in cui si risolve il problema (8.2).

Nel seguito si descrivono gli step della prima euristica. Essa cicla fra la risoluzione del problema (8.2) e la ricerca di una sequenza opportuna.

Come già detto, viene inizialmente risolto il problema (8.2), prima risolvendo la sua versione rilassata e poi trasformando la soluzione così ottenuta in una soluzione con valori x_k interi. Si sottolinea che i metodi di generazione dei pattern da far entrare in base sono leggermente alterati per evitare a priori di avere pattern con un numero di gruppi associati superiore al limite t prima stabilito per il numero di stack aperti, cioè secondo la stessa logica del lower bound LB_1 del lavoro di Yanasse (1997b) descritto nel precedente Capitolo. In pratica, se un pattern ha più di t gruppi associati, è certo che si apriranno contemporaneamente più di t pattern in una qualunque sequenza che contenga quel pattern.

Una volta ottenuto un insieme di pattern, soluzione non necessariamente ottima del problema (8.2) non rilassato, si cerca una sequenza che permetta di non superare il limite t stabilito. Come già detto in precedenza, la ricerca si basa sull'euristica Yuen3 proposta da Yuen (1995) e descritta nel precedente Capitolo.

Se la sequenza trovata è buona, allora l'algoritmo verifica se il costo totale r associato all'insieme di pattern è inferiore a quello relativo alla migliore soluzione corrente, cioè all'insieme per il quale dovesse essere stata già verificata l'esistenza di una sequenza buona e con il migliore (minimo) valore corrente di costo totale. In caso affermativo, la migliore soluzione corrente viene aggiornata.

Se inoltre r è uguale al lower bound $\underline{r}$, o se vi è sufficientemente vicino secondo parametri stabiliti a priori, l'algoritmo termina. In caso contrario, si procede, per un massimo di *max_iter* iterazioni. Prima della iterazione successiva, si aggiunge la proibizione di almeno un pattern. Inoltre, se la sequenza trovata non era buona, si alterano i pattern fino a rendere buona la sequenza, attraverso il seguente procedimento. Per ciascun pattern p_k si calcola il costo aggiuntivo che si genera se esso viene sostituito da uno o più pattern $po_1, \ldots, po_{g(k)}$, ciascuno contenente un solo tipo di pezzi e per questo detto omogeneo (Herz, 1972), cioè un pattern per ciascuno dei $g(k)$ tipi di pezzi coinvolti nel pattern sostituito. I pattern p_k della sequenza vengono quindi ordinati per valori non crescenti dell'aumento di costo associato e ciascuno di essi, in tale ordine e finché non si ottiene una sequenza buona, viene sostituito dai relativi pattern omogenei $po_1, \ldots, po_{g(k)}$.

Per la sequenza buona così ottenuta si calcola il valore r e si verifica la relazione fra r ed il valore associato alla migliore soluzione corrente, come nel caso in cui la sequenza trovata dall'euristica fosse stata già buona. Comunque, in tal caso, tutti i pattern sostituiti vengono proibiti prima della prossima iterazione.

Si noti dunque che, mentre Armbruster seleziona i pattern a partire dai nodi-gruppo, Pileggi lo fa uno per uno. In effetti, il metodo di Armbruster sembra migliore, dato che parte dai nodi che causano impedimenti nella costruzione di una sequenza buona, mentre Pileggi rischia di proibire pattern che, sebbene hanno un costo inferiore, in termini di costo aggiunto, potrebbero non essere per niente correlati all'impossibilità di ottenere una sequenza buona.

La seconda euristica proposta da Pileggi, come anticipato precedentemente, non parte dalla soluzione del modello (8.2), ma si basa su una procedura costruttiva greedy per la generazione dell'insieme di pattern, poi da sequenziare. La tecnica è simile a quella di Hinxmann (1980), che prevede di ottimizzare uno stock per volta, ripeterlo il massimo

numero di volte possibile, in relazione alle domande residui dei pezzi, e poi iterare il procedimento con i pezzi rimasti.

Tale metodo euristico costruttivo viene però alterato per tener subito conto del limite *t* sul massimo numero di stack aperti, come si spiega nel seguito. Per la generazione del successivo pattern, si sfrutta, a seconda dei casi, mono-dimensionale, bidimensionale 2-staged e bidimensionale con stage illimitati, lo stesso algoritmo del caso della prima euristica complessiva, cioè uno degli algoritmi fra quelli di Gilmore e Gomory (1963), di Gilmore e Gomory (1965) e di Morabito e Arenales (1996). Essi sono opportunamente alterati in modo che, nella ricerca del pattern, si permettano solo i pattern con un numero di gruppi non maggiore di *t*, e con l'obiettivo di ottimizzare il massimo numero di stack che si otterrebbero permutando al meglio l'insieme di pattern che contiene il pattern correntemente verificato e tutti gli altri pattern fissati nelle precedenti iterazioni dell'algoritmo di Hinxman (1980). La permutazione migliore viene trovata con l'euristica Yuen3.

Una ulteriore modifica consiste nell'eseguire più volte la tecnica di Hinxman, ogni volta con la modifica spiegata sopra. Alla fine di ogni esecuzione, nell'ipotesi semplificativa, ma che non determina perdita di generalità, che ogni gruppo di pezzi contenga un sol tipo di pezzi, si alterano i profitti associati ai pezzi, che la prima volta sono pari alle lunghezze, o alle aree nel caso bidimensionale, dei pezzi stessi. Se per il *j*-mo tipo (o gruppo) di pezzi il profitto è denotato con π_j, tale profitto viene alterato e portato al valore $(1 + \beta_j)\cdot \pi_j$, dove β_j è un valore appositamente ottenuto per migliorare il profitto dei pezzi che si trovano su pattern che hanno un costo maggiore e peggiorare quello dei pezzi che si trovano su pattern che hanno un costo inferiore, con un'idea dunque simile a quanto proposto in Perboli, Crainic e Tadei (2006).

In particolare, i coefficienti β_j sono ottenuti con l'intento di ridurre al minimo le differenze fra ciascuna somma $\sum_{j=1}^{n} \beta_j p_{jk}$, associata al pattern p_k, ed il relativo costo r_k, dove *n* è il numero di tipi di pezzi. Si considera a tal fine che l'approccio di Hinxman è tale da esaurire la domanda di almeno un tipo di pezzo per ciascun pattern, che viene appunto replicato il massimo numero di volte fino ad esaurire la domanda di almeno un tipo di pezzi, anche se eventualmente essa viene sforata con l'ultima ripetizione. Il numero *q* di differenti pattern è allora al più pari ad *n*.

Si può dunque considerare il sistema lineare costituito dalle uguaglianze $\sum_{j=1}^{n} \beta_j p_{jk} = r_k$, al variare di *k* da 1 a *q*. Ogni volta che si esaurisce la domanda di un tipo di pezzi, diciamo di indice *j*, per l'aggiunta delle ripetizioni del pattern p_i, allora i pattern successivi, cioè con $k > i$, sono tali da avere il *j*-mo elemento p_{jk} pari a zero. Di conseguenza il rango della matrice del sistema è pari a *q*. Dunque, se $q = n$, allora il sistema ha una sola soluzione. Se invece $q < n$, allora il sistema ha infinite soluzioni. Se ne ricava che si riesce sempre ad ottenere una differenza minima fra le somme $\sum_{j=1}^{n} \beta_j p_{jk}$ ed i relativi valori r_k.

Si indica infine che il procedimento di ripetizione dell'approccio costruttivo di Hinxman viene terminato nel caso in cui i valori dei profitti π_j risultano sufficientemente simili fra una iterazione e l'altra, anche se non si è raggiunto il massimo numero di iterazioni consentite.

Infine, la terza euristica proposta da Pileggi combina le idee delle prime due euristiche. In particolare, lavora come la prima euristica, ma introducendo le proibizioni nella fase di generazione dei pattern, come fatto nella seconda euristica. In pratica non permette di passare in vertici del simplesso cui corrisponda una soluzione intera (ottenuta con l'approssimazione per eccesso), cioè un insieme di pattern, per cui non esista una sequenza

che non sfori il limite *t* sul massimo numero di stack aperti. Ancora una volta la verifica dell'esistenza di una tale sequenza è realizzata con l'euristica Yuen3. Si noti anche che, oltre a modificare l'algoritmo di generazione del pattern che deve entrare in base, come fatto nella prima euristica, si modifica lo stesso algoritmo del simplesso, appunto per evitare di passare in vertici "non buoni". Questo implica che, anche laddove, al posto dell'euristica Yuen3 si utilizzasse una tecnica di ricerca esatta della sequenza ottima, l'algoritmo resterebbe euristico. E' infatti probabile che, per giungere al vertice ottimo fra quelli associati a sequenze buone, si debba passare per vertici con nessuna sequenza buona associata.

Infine, Pileggi suggerisce che le sue euristiche possono essere utilizzate per ottenere curve di trade-off fra il costo totale ed il massimo numero di stack aperti, facendo variare di volta in volta il limite *t*, con passo 1, dato che deve avere valore intero. Ovviamente questa idea è applicabile a qualunque metodo che usi un limite sul massimo numero di stack aperti.

8.1.3 Gli approcci di Belov

I due approcci di Belov (2003) sono stati proposti con riferimento al caso mono-dimensionale, ma l'idea è applicabile anche al caso bidimensionale, se si sfrutta un diverso metodo di generazione dei pattern, come del resto fatto nel lavoro di Pileggi.

In un primo approccio, Belov parte dalla procedura di Mukhacheva e Zalgaller (1993), proposta per il problema di Cutting multi-stock, e che rappresenta una evoluzione rispetto al metodo di Hinxman (1980), in quanto, man mano che costruisce pattern, utilizza informazioni relative ai pattern già generati, così da guidare la costruzione del pattern successivo, attraverso l'alterazione dei profitti dei pezzi, cioè con lo stesso principio usato da Pileggi e pure nell'adattamento dell'euristica descritta nel terzo Capitolo a casi multi-stock, come spiegato nel paragrafo 3.7.

La procedura di Mukhacheva e Zalgaller è chiamata SVC (*Sequential Value Correction*). Belov introduce alcuni accorgimenti per migliorarla, qui non descritti per brevità, in riferimento al solo problema di Cutting. Si sottolinea solo che, a differenza di quanto fatto nelle euristiche di Pileggi (2002), ciascun pattern viene replicato per un numero di volte massimo tale da non sforare la domanda di alcun tipo di pezzi.

Relativamente alla combinazione con il MOSP, Belov affronta sia il caso in cui non vi sia un vincolo al massimo numero di stack aperti, sia il caso in cui vi sia.

Quando tale vincolo non c'è, l'obiettivo di minimizzare il massimo numero di stack aperti è considerato come secondario rispetto alla minimizzazione dello scarto totale, e se ne tiene conto alterando i profitti dei pezzi moltiplicando, a valle dell'alterazione presente nella SVC modificata, il generico profitto π_j per il fattore $\beta_0 \cdot \rho^h$, dove β_0 e ρ sono prefissati parametri maggiori di uno ed *h* è l'indice dell'iterazione della procedura SVC. Tale alterazione è in realtà eseguita solo sui pezzi dei gruppi per cui il relativo stack sia già aperto.

Nel caso, invece, di limite al massimo numero di stack aperti, il nuovo pattern da aggiungere in ciascuna iterazione della procedura viene selezionato solo fra quelli che non determinano uno sforamento del limite. Belov indica che la generazione del pattern avviene con una procedura branch and bound, che quindi viene modificata al pari di quanto fatto da Pileggi rispetto, ad esempio, al metodo di Gilmore e Gomory (1963).

Si sottolinea che il lavoro di Belov, inoltre, affronta sia il problema della minimizzazione del numero di differenti pattern, sia il problema in cui siano combinati gli obiettivi di minimizzazione del numero di differenti pattern e di minimizzazione del massimo numero di stack aperti.

Un secondo approccio è basato su un modello lineare, descritto nel seguito, che combina la generazione dei pattern ed il loro sequenziamento. Le variabili x_k rappresentano la frequenza del *k*-mo pattern e le variabili p_{jk} rappresentano il numero di pezzi di tipo *j* presenti

nel *k*-mo pattern. Si sottolinea che si suppone l'equivalenza fra i tipi ed i gruppi di pezzi, secondo la stessa semplificazione utilizzata nella terza euristica di Pileggi. Il valore x^* rappresenta il numero di stock ottenuto con una procedura che minimizzi lo scarto, cioè relativa al solo problema di Cutting. Gli stock sono qui supposti tutti uguali, per cui la minimizzazione dello scarto equivale alla minimizzazione del numero di stock. Il principio di fondo è lo stesso di quello della prima euristica di Pileggi, in cui si itera la procedura tentando di avvicinarsi allo scarto minimo, lì denotato con $\underline{r}$. Belov, infatti, impone lo scarto minimo e minimizza il massimo numero di stack aperti, potendo contare sulla semplificazione di avere stock identici.

Si suppone inoltre di conoscere un upper bound $\overline{K}$ sul numero di pattern differenti che sarà necessario utilizzare. Migliore è l'upper bound e meno variabili avrà il modello.

Vengono inoltre introdotte le seguenti variabili per tener conto degli obiettivi di Sequencing. La variabile binaria s_{jk} vale 0 se il *j*-mo tipo di pezzi non è stato estratto in nessuno dei primi *k* pattern della sequenza ed 1 altrimenti. La variabile binaria e_{jk} vale 0 se per il *j*-mo tipo di pezzi non è stata terminata l'estrazione prima della processazione del *k*-mo pattern. Se d_j rappresenta la domanda associata al *j*-mo tipo di pezzi, valgono dunque le due seguenti relazioni.

$$s_{jk} = \begin{cases} 0 \quad if \sum_{h=1}^{k} p_{jh} x_h = 0 \\ 1 \quad if \sum_{h=1}^{k} p_{jh} x_h > 0 \end{cases} \qquad e_{jk} = \begin{cases} 0 \quad if \sum_{h=1}^{k-1} p_{jk} x_h < d_j \\ 1 \quad if \sum_{h=1}^{k-1} p_{jk} x_h \geq d_j \end{cases}$$

Vengono anche definite le variabili binarie $\underline{e}_{ij}$ nel seguente modo.

$$\underline{e}_{jk} = \begin{cases} 1 \quad if \sum_{h=k}^{\overline{K}} p_{jh} x_h > 0 \\ 0 \quad if \sum_{h=k}^{\overline{K}} p_{jh} x_h = 0 \end{cases}$$

Vale la complementarietà fra le variabili e_{jk} e le variabili $\underline{e}_{jk}$, secondo la relazione $e_{jk} = 1 - \underline{e}_{jk}$. Sulla base delle variabili descritte, si può definire la funzione binaria *open*(*j*, *k*), che vale 1 se il gruppo *j* risulta aperto al momento della processazione del *k*-mo pattern. Vale infatti, secondo tale definizione, la relazione $open(j, k) = s_{jk} - e_{jk}$.

Se dunque l'obiettivo di Sequencing è minimizzare l'order spread medio, va minimizzata la somma $\sum_{j=1}^{n} \sum_{k=1}^{\overline{K}} open(j,k) \cdot x_k$. In realtà Belov propone la minimizzazione dell'order spread massimo, introducendo una variabile *MaxSpread*, da minimizzare, per la quale valgono i vincoli $MaxSpread \geq \sum_{k=1}^{\overline{K}} open(j,k) \cdot x_k$, al variare di *j* da 1 ad *n*. Nel caso in cui, invece, si voglia minimizzare il massimo numero di stack aperti, allora va minimizzata la massima somma $\sum_{j=1}^{n} open(j,k)$, al variare di *k*, dove *n* è il numero di gruppi. Questo è ottenuto definendo una apposita variabile *MaxOs* da minimizzare, per la quale valgono i vincoli $MaxOs \geq \sum_{j=1}^{n} open(j,k)$, al variare di *k*, con il seguente modello.

$$
\begin{aligned}
&\min && MaxOS && && \\
&s.t. && MaxOS \geq \sum_{j=1}^{n} \left(s_{jk} + \underline{e}_{jk} - 1 \right) && k = 1, \ ..., \ \overline{K} && \text{(a)} \\
& && \sum_{h=1}^{k} p_{jh} x_h \leq d_j s_{jk} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ \overline{K} && \text{(b)} \\
& && \sum_{h=k}^{\overline{K}} p_{jh} x_h \leq d_j \underline{e}_{jk} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ \overline{K} && \text{(c)} \\
& && \sum_{k=1}^{\overline{K}} p_{jk} x_k \geq d_j && j = 1, \ ..., \ n && \text{(d)} \\
& && \sum_{j=1}^{n} p_{jk} l_j \leq L && k = 1, \ ..., \ \overline{K} && \text{(e)} \\
& && \sum_{k=1}^{\overline{K}} x_k \leq x^* && && \text{(f)} \\
& && p_{jk} \in \{0, 1, ..., d_j\} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ \overline{K} && \text{(g)} \\
& && x_k \in \{0, 1, ..., x^*\} && k = 1, \ ..., \ \overline{K} && \text{(h)} \\
& && s_{jk}, \underline{e}_{jk} \in \{0, \ 1\} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ \overline{K} && \text{(i)}
\end{aligned}
\tag{8.3}
$$

Tale modello, in realtà, non è lineare, per i vincoli (8.3b), (8.3c) ed (8.3d), che presentano prodotti fra le variabili p_{jk} e le variabili x_k.

Per trasformarlo in un modello lineare, Belov propone di utilizzare la stessa strategia usata da Kantorovich (1960), attraverso opportune variabili y_k. Nel nuovo modello, si considerano i pattern uno per uno, senza dunque utilizzare variabili di frequenza, ed il vincolo (8.3f) sul numero di pattern si trasforma nel dare al valore x^* il ruolo prima assunto da $\overline{K}$. Il modello diventa quindi il seguente.

$$
\begin{aligned}
&\min && MaxOS && && \\
&s.t. && MaxOS \geq \sum_{j=1}^{n} \left(s_{jk} + \underline{e}_{jk} - 1 \right) && k = 1, \ ..., \ x^* && \\
& && \sum_{h=1}^{k} p_{jh} \leq d_j s_{jk} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ x^* && \\
& && \sum_{h=k}^{x^*} p_{jh} \leq d_j \underline{e}_{jk} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ x^* && \\
& && \sum_{k=1}^{x^*} p_{jk} \geq d_j && j = 1, \ ..., \ n && \\
& && \sum_{j=1}^{n} p_{jk} l_j \leq L \cdot y_k && k = 1, \ ..., \ x^* && \\
& && p_{jk} \in \{0, 1, ..., d_j\} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ x^* && \\
& && y_k \in \{0, \ 1\} && k = 1, \ ..., \ x^* && \\
& && s_{jk}, \underline{e}_{jk} \in \{0, \ 1\} && j = 1, \ ..., \ n \ ; \ k = 1, \ ..., \ x^* &&
\end{aligned}
\tag{8.4}
$$

Belov propone di approcciare la risoluzione del problema (8.4) attraverso un metodo di column generation come nel caso del modello di Gilmore e Gomory, come discusso nei precedenti paragrafi. Sottolinea inoltre che il modello potrebbe essere alterato con l'utilizzo di variabili di *subpattern*, sfruttando cioè le stesse idee che propone nel suo lavoro per passare dal modello di Gilmore e Gomory ad un modello diverso, ma questo argomento esula dagli scopi del corrente lavoro. La cosa interessante ai fini della combinazione fra Cutting e Sequencing è che Belov indica che il rilassamento continuo del modello (8.4) rischia di mirare alla riduzione del numero di tipi diversi di pezzi per ciascun pattern piuttosto che alla riduzione del numero di stack aperti e propone perciò una ulteriore variazione, ridefinendo la funzione *open*(*j*, *k*) sulla base di nuove variabili ds_{jk} e de_{jk}, secondo le seguenti relazioni

$$de_{jk} = \max\left\{de_{j,k+1}, s_{jk} - e_{jk}\right\}$$
$$ds_{jk} = \max\left\{ds_{j,k-1}, s_{jk} - e_{jk}\right\}$$
$$open(j,k) = ds_{jk} + de_{jk} - 1$$

oppure con le ulteriori variabili md_{jk}, secondo le relazioni

$$md_{jk} = \max\left\{ds_{jk}, de_{jk}\right\}$$
$$open(j,k) = ds_{jk} + de_{jk} - md_{jk}$$

L'interpretazione di tali variabili è la seguente. Il valore de_{jk} vale 1 se l'ultimo pattern su cui risulta aperto lo stack relativo al gruppo *j* ha indice maggiore o uguale a *k*, e 0 altrimenti. Il valore ds_{jk} vale 1 se il primo pattern su cui risulta aperto lo stack relativo al gruppo *j* ha indice maggiore o uguale a *k*, e 0 altrimenti.

8.1.4 L'approccio di Yanasse e Pinto Lamosa

Il lavoro di Yanasse e Pinto Lamosa (2007) approccia la combinazione fra Cutting e MOSP attraverso un modello derivato dall'analogia con il problema MTSP, già sfruttato nel lavoro di Yanasse (1997b) e da cui Yanasse aveva appunto derivato un modello lineare per il MOSP, come spiegato nel Capitolo precedente.

Quel modello prevedeva però di conoscere a priori i pattern da utilizzare, mentre nell'approccio combinato non può essere così e le quantità di pezzi di ciascun gruppo per ciascun pattern passano da costanti a variabili, come già negli approcci descritti nei paragrafi precedenti.

Anche in questo caso, si fa l'ipotesi semplificativa secondo cui ciascun gruppo di pezzi sia costituito da un solo tipo di pezzi. Questo non comporta perdita di generalità, poiché l'effetto consisterebbe nella definizione di due differenti parametri, uno pari al numero di tipi di pezzi ed uno pari al numero di gruppi relativi agli stack, per cui nei vincoli di soddisfacimento della domanda andrebbero utilizzati variabili e parametri associati ai singoli tipi di pezzi, mentre nei vincoli relativi agli stack andrebbero usati variabili e parametri "aggregati", che tengano cioè conto, per ciascun gruppo, di tutti i tipi di pezzi di quel gruppo.

L'approccio parte dai seguenti due modelli. Il primo è quello per il Cutting proposto da Gilmore e Gomory (1961).

$$\min \quad \sum_{i=1}^{m} c_i x_i \tag{8.5}$$

$$s.t. \quad \sum_{i=1}^{m} a_{ij} x_j \geq d_j \quad j = 1, \ ..., n \tag{8.6}$$

$$x_i \in \mathbb{N}_0 \qquad i = 1, \ ..., m \tag{8.7}$$

dove i valori a_{ij} sono gli analoghi dei valori p_{ij} nei modelli da (8.1) ad (8.4), ma gli indici assumono un ruolo invertito rispetto a quanto avveniva in quei modelli. Ciascun valore a_{ij} rappresenta infatti il numero di pezzi di tipo j presenti sul pattern i. I valori c_i rappresentano un costo associato a ciascun pattern i, ed assumono lo stesso ruolo generalizzato dei valori r_i del modello (8.2). Il valore n è il numero di tipi di pezzi, mentre m è il numero di tutti i possibili pattern. Le variabili x_j rappresentano le frequenze dei pattern, così come nei modelli da (8.1) ad (8.3).

Il modello di MOSP, derivato da quello di Tang e Dendardo (1988) per l'MTSP, è qui nuovamente riportato.

$$\min \quad T \tag{8.8}$$

$$s.t. \quad \sum_{k=1}^{m} \sum_{j=1}^{n} P_{kj} \geq m - T \tag{8.9}$$

$$P_{kj} \geq W_{k+1,j} - W_{kj} \quad k = 1, \ ..., m-1 \ ; \ j = 1, \ ..., n \tag{8.10}$$

$$P_{kj} \geq 0 \qquad k = 1, \ ..., m \ ; \ j = 1, \ ..., n \tag{8.11}$$

$$\sum_{j=1}^{n} W_{kj} \leq T \qquad k = 1, \ ..., m \tag{8.12}$$

$$y_{ik} a_{ij} \leq W_{kj} \qquad i, k = 1, \ ..., m \ ; \ j = 1, \ ..., n \tag{8.13}$$

$$\sum_{i=1}^{m} y_{ik} = 1 \qquad k = 1, \ ..., m \tag{8.14}$$

$$\sum_{k=1}^{m} y_{ik} = 1 \qquad i = 1, \ ..., m \tag{8.15}$$

$$y_{ik} \in \{0, \ 1\} \qquad i, k = 1, \ ..., m \tag{8.16}$$

$$W_{kj} \in \{0, \ 1\} \qquad k = 1, \ ..., m \ ; \ j = 1, \ ..., n \tag{8.17}$$

Il valore T rappresenta il massimo numero di stack aperti. Il valore P_{kj} è 1 se il j-mo stack, associato cioè al j-mo tipo di pezzi, si apre nel passaggio dal pattern fra la k-ma e la (k+1)-ma posizione, e 0 altrimenti. Il valore binario W_{kj} è pari ad 1 se lo stock relativo al j-mo tipo di pezzi è aperto prima della terminazione del pattern in k-ma posizione nella sequenza, e 0 altrimenti.

I valori binari y_{ik} sono pari ad 1 se l'i-mo pattern è posto in k-ma posizione nella sequenza. I parametri a_{ij} sono quelli del modello (8.5)-(8.7).

Per la comprensione del modello, si rimanda al paragrafo 7.2.2, dove è riportato un semplice teorema di Yanasse (1997b) su cui si basa la validità del modello.

Nel lavoro di Yanasse e Pinto Lamosa, come già in quelli di Armbruster e di Pileggi, nel combinare i problemi di Cutting ed il problema MOSP, il valore di T è assunto costante, e

pari ad un limite superiore per il massimo numero di stack aperti, mantenendo l'obiettivo della minimizzazione dello scarto.

Per poter combinare i due modelli, Yanasse e Pinto Lamosa hanno introdotto le variabili binarie v_i, con $i = 1, \ldots, m$. Ciascuna variabile v_i vale uno se l'i-mo pattern è utilizzato, cioè se vale la relazione $x_i > 0$, e zero altrimenti.

Le disequazioni che legano le nuove variabili v_i alle variabili x_i sono dunque le seguenti.

$$v_i \leq x_i \tag{8.18}$$

$$K \cdot v_i \geq x_i \qquad i = 1, \ldots, m \tag{8.19}$$

$$v_i \in \{0, 1\} \qquad i = 1, \ldots, m \tag{8.20}$$

Infatti, la (8.18) implica l'azzeramento di v_i quando $x_i = 0$, mentre la (8.19) implica l'azzeramento di x_i quando $v_i = 0$. Il valore della costante K è sufficientemente grande da evitare limiti per x_i quando $v_i = 1$. In pratica dovrebbe essere almeno pari al massimo valore che x_i può assumere, secondo la domanda dei pezzi associati all'i-mo pattern. Dovendo però valere per tutti i pattern, deve essere almeno pari al massimo fra i singoli valori minimi associabili a ciascun pattern.

Per poter però fondere i modelli, bisogna considerare, in relazione ai vincoli relativi al MOSP, che non tutti i pattern vengono in realtà scelti, per cui i pattern non selezionati possono finire in posizioni qualsiasi nella sequenza, senza che però ci debba essere influenza sui valori W_{kj} e P_{kj}. Poiché le posizioni sono identificate dai valori x_{ik}, questo si ottiene alterando il vincolo (8.13), che viene trasformato nel seguente vincolo

$$y_{ik} a_{ij} + (v_i - 1) a_{ij} \leq W_{kj} \quad i, k = 1, \ldots, m \; ; \; j = 1, \ldots, n \tag{8.21}$$

in modo che, quando $v_i = 0$, con $y_{ik} = 1$, nessun valore W_{kj} sia vincolato a dover essere maggiore del corrispondente valore a_{ij}.

Il modello combinato ha dunque l'obiettivo (8.5) ed i vincoli (8.6), (8.7), da (8.9) ad (8.12) e da (8.14) ad (8.21).

Una volta definito così il modello, Yanasse e Pinto Lamosa sottolineano che il numero di variabili cresce troppo velocemente con i parametri m ed n, e questo rende difficile il problema, dato che a sua volta il valore m, che rappresenta il numero di tutti i possibili pattern, cresce con n. Ci si trova cioè dinanzi ad una amplificazione delle difficoltà che hanno spinto Gilmore e Gomory a proporre un approccio di column generation già sul solo problema (8.5)-(8.7).

Per ovviare alle suddette difficoltà, il problema combinato è stato allora approcciato, nel lavoro che si sta descrivendo, attraverso un rilassamento Lagrangiano, che permette la separazione del problema combinato in due sottoproblemi, spezzando il legame che era stato appositamente introdotto attraverso i vincoli (8.18) ed (8.19).

Per poter giungere a tale obiettivo, Yanasse e Pinto Lamosa sostituiscono i vincoli (8.18) ed (8.19) con i seguenti due vincoli

$$K_{1i} v_i \leq x_i \qquad i = 1, \ldots, m \tag{8.22}$$

$$-K_{2i} \cdot v_i \leq -x_i \qquad i = 1, \ldots, m \tag{8.23}$$

che non alterano l'essenza del modello se ciascun K_{1i} non è maggiore del corrispondente valore x_i e se ciascun K_{2i} non è minore dello stesso x_i. Non si esplicitano però i corrispondenti

vincoli che legano K_{1i} e K_{1i} ad x_i, in quanto essi sono usati come perno fra i due sottoproblemi che emergono dalla separazione spiegata nel seguito.

Per risolvere il problema, qui denotato con CSP, con i vincoli (8.22) ed (8.23) al posto del vincoli (8.18) ed (8.19), si sfrutta un rilassamento Lagrangiano proprio sui vincoli (8.22) ed (8.23), trasformando così la funzione obiettivo nella seguente

$$\sum_{i=1}^{m}\left[c_i x_i + \lambda_{1i}\left(K_{1i}v_i - x_i\right) + \lambda_{2i}\left(x_i - K_{2i}v_i\right)\right]$$

I valori λ_{1i} e λ_{2i} rappresentano i moltiplicatori Lagrangiani associati ai vincoli (8.22) ed (8.23), che vengono dunque trascurati avendoli tradotti in componenti della funzione obiettivo.

Il problema così costruito è denotato con LAGP(λ) da Yanasse e Pinto Lamosa. Appare chiaro che a questo punto il problema è separabile in due, e cioè nei due seguenti sottoproblemi.

$$\text{LAGP1}(\lambda)\begin{cases}\min & \sum_{i=1}^{m}\left(c_i x_i - \lambda_{1i}x_i + \lambda_{2i}x_i\right)\\ s.t. & (8.6),\ (8.7)\end{cases}$$

$$\text{LAGP2}(\lambda)\begin{cases}\min & \sum_{i=1}^{m}\left(\lambda_{1i}K_{1i} - \lambda_{2i}K_{2i}\right)v_i\\ s.t. & (8.9)\text{-}(8.12),\ (8.14)\text{-}(8.17),\ (8.20),\ (8.21)\end{cases}$$

Se, per fissati valori dei moltiplicatori λ_{1i} e λ_{2i}, si risolve il problema LAGP1(λ), allora si possono fissare i valori delle variabili frequenza x_i, trattate dal problema LAGP2(λ) come parametri. Inoltre, tornando sui valori K_{1i} e K_{2i} , basta porre $K_{1i} = K_{1i} = x_i$ per ogni i, secondo quanto stabilito circa i vincoli (8.22) ed (8.23). Ne deriva che la funzione obiettivo del problema LAGP2(λ) diventa $\sum_{i=1}^{m} x_i\left(\lambda_{1i} - \lambda_{2i}\right)v_i$.

Per risolvere ottimamente il problema originario, si segue la procedura (Nemhauser e Wolsey, 1988; Wolsey, 1998) di risolvere prima il duale Lagrangiano per poi trasformare la sua soluzione ottima in una soluzione fattibile per CSP, qualora non lo fosse.

Il duale Lagrangiano è il problema in cui i moltiplicatori assumono il ruolo di variabili e l'obiettivo è trovare i valori dei moltiplicatori che, in questo caso, massimizzino l'ottimo del problema LAGP(λ), i cui parametri sono funzione dei valori dei moltiplicatori, e che, come visto, è risolvibile sfruttando la separazione nei due sottoproblemi LAGP1(λ) e LAGP2(λ).

Infatti, qualunque siano i valori dei moltiplicatori, purché non negativi, l'ottimo del problema LAGP(λ) è, in questo caso, un lower bound per l'ottimo di CSP, per cui il massimo fra gli ottimi dei problemi LAGP(λ) al variare dei valori non negativi dei moltiplicatori, rappresenta il miglior lower bound ottenibile dai rilassamenti Lagrangiani associati al rilassamento dei vincoli (8.22) ed (8.23). In generale, quando si considerano solo valori non negativi dei moltiplicatori, se la soluzione ottima del duale Lagrangiano, ristretta ai soli valori delle variabili originarie e cioè trascurando i valori dei moltiplicatori, è fattibile per il problema originario, allora essa è anche ottima per il problema originario. Tuttavia Yanasse e Pinto Lamosa permettono valori negativi per i moltiplicatori, ricavandone un vantaggio come si descrive nel seguito, ma perdendo tale proprietà.

Si noti che il duale Lagrangiano, come sempre del resto, non è un problema lineare. Per la sua risoluzione, Yanasse e Pinto Lamosa utilizzano il metodo del sottogradiente, sottolineando che in tal caso si tratta di un metodo modificato per il fatto che in principio, l'uguaglianza $K_{1i} = K_{1i} = x_i$ rende ciascun problema LAGP(λ) non lineare, ma la separazione in due sottoproblemi di ciascun LAGP(λ) premette di affrontare problemi lineari.

Il suddetto metodo prevede di realizzare un ciclo iterativo, risolvendo ogni volta il problema LAGP(λ) associato ai valori fissati per i moltiplicatori nel ciclo corrente. Fra un ciclo e l'altro, i valori dei moltiplicatori vengono aggiornati.

L'aggiornamento di tali valori avviene con la seguente regola generale

$$\lambda_q^{h+1} = \max \left\{ \lambda_q^h + \mu_h \left(B^q \cdot \eta(\lambda^h) - b_q \right),\ 0 \right\}$$

dove λ_q^h identifica la q-ma generica componente del vettore λ alla h-ma iterazione. Si ricorda che, nel caso in esame, il vettore λ contiene $2m$ componenti, cioè una per ogni vincolo tradotto in una componente della funzione obiettivo, e cioè m per la (8.22) ed m per la (8.23). La matrice B ed il vettore b sono quelli tali che la forma $B \cdot \eta \leq b$ rappresenti in modo compatto i vincoli (8.22) ed (8.23). Dunque B^q denota la q-ma riga della matrice B, mentre $\eta(\lambda^h)$ rappresenta il vettore di $2m$ elementi con i valori che le variabili v_i ed x_i, coinvolte nei suddetti due coinvolti, assumono nella soluzione ottima del problema LAGP(λ) alla h-ma iterazione.

Nel caso in esame il vettore b, di $2m$ elementi, ha valori tutti nulli, mentre la matrice B, quadrata, con dimensioni $2m \times 2m$, ha molti elementi nulli, secondo il seguente schema.

$$\begin{pmatrix}
K_{11} & 0 & \dots & & 0 & -1 & 0 & \dots & & 0 \\
0 & K_{12} & 0 & \dots & 0 & 0 & -1 & 0 & \dots & 0 \\
\dots & & & & & \dots & & & & \dots \\
0 & \dots & & 0 & K_{1n} & 0 & \dots & & 0 & -1 \\
-K_{21} & 0 & \dots & & 0 & 1 & 0 & \dots & & 0 \\
0 & -K_{22} & 0 & \dots & 0 & 0 & 1 & 0 & & 0 \\
\dots & & & & \dots & \dots & & & & \dots \\
0 & \dots & & 0 & -K_{2n} & 0 & & & 0 & 1
\end{pmatrix}
\begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}
\leq
\begin{pmatrix} 0 \\ \dots \\ \\ \\ \\ \\ \\ 0 \end{pmatrix}$$

La matrice B è cioè costituita da quattro blocchi quadrati, di dimensioni $n \times n$, con elementi non nulli solo sulle relative diagonali.

Il parametro μ_h è un coefficiente che stabilisce l'entità della perturbazione nella variazione dei valori dei moltiplicatori. La tecnica utilizzata (Wolsey, 1998) prevede di porre tale valore pari, in principio, al risultato dell'espressione

$$\phi \cdot \frac{z^* - LB}{\left\| B \cdot \eta - b \right\|^2} \tag{8.24}$$

dove z^* è l'ottimo del problema iniziale CSP, cioè senza rilassamento Lagrangiano, LB è un lower bound del problema iniziale, e ϕ è un parametro costante compreso fra 0 e 2. In realtà la tecnica standard prevedrebbe di utilizzare la differenza fra un lower bound LB_{DL} del duale Lagrangiano ed il valore $z(\lambda^h)$ ottimo del problema LAGP(λ^h), con l'assicurazione che il valore $z(\lambda^h)$, al divergere di h, tende al valore LB_{DL} oppure diventa, dopo un numero finito di

passi, maggiore o uguale di LB_{DL} e minore o uguale dell'ottimo del duale Lagrangiano. Tuttavia, in generale, la difficoltà risiede nel conoscere un valore che possa assumere il ruolo di LB_{DL}. D'altro canto lo stesso valore z^* non è noto, essendo l'obiettivo del problema originario. Più avanti si dettaglia cosa sia stato usato da Yanasse e Pinto Lamosa al posto di z^* e cosa nel ruolo di LB.

Il sottogradiente $B\cdot\eta(\lambda^h)$ (b infatti è nullo) da utilizzare alla h-ma iterazione è un vettore di $2n$ elementi, che ha due blocchi B_1 e B_2, ciascuno di n elementi. Il primo ha i-mo elemento B_{1i} pari a $K_{1i}\cdot v^h_i - x^h_i$, mentre il secondo ha i-mo elemento B_{2i} pari a $x^h_i - K_{2i}\cdot v^h_i$, dove tali espressioni rispecchiano appunto quelle presenti nei vincoli (8.22) ed (8.23), i valori x^h_i sono ottenuti come soluzione ottima del problema LAGP1(λ^h) ed i valori v^h_i come soluzione ottima del problema LAGP2(λ^h) in cui $K_{1i} = K_{2i} = x^h_i$.

Dunque valgono le relazioni $B_{1i} = x^h_i\cdot(v^h_i - 1)$ e $B_{2i} = x^h_i\cdot(1 - v^h_i) = -B_{1i}$. Yanasse e Pinto Lamosa sottolineano che, nei casi in cui $x^h_i = 0$ oppure $v^h_i = 1$, il valore della relativa componente del sottogradiente è nullo.

Ciò vuol dire che le uniche componenti del vettore λ, che vengono perturbate in ciascuna iterazione, sono quelle λ_{1i} e λ_{2i} associate agli indici i per i quali $x_i > 0$ e $v_i = 0$. Tale insieme è denotato con I^h. Per tali indici, si ottiene $B_{1i} = -x^h_i$ e $B_{2i} = x^h_i$ e vengono utilizzate le seguenti relazioni

$$\lambda^{h+1}_{1i} = \lambda^h_{1i} - \theta\cdot x^h_i$$
$$\lambda^{h+1}_{2i} = \lambda^h_{2i} + \theta\cdot x^h_i$$

mentre, per tutti gli indici i non appartenenti ad I^h, cioè per i quali $x^h_i = 0$ oppure $v^h_i = 1$, si ha $\lambda^{h+1}_{1i} = \lambda^h_{1i}$ e $\lambda^{h+1}_{2i} = \lambda^h_{2i}$. Dunque, Yanasse e Pinto Lamosa permettono valori negativi per i moltiplicatori, con le conseguenze prima sottolineate.

Si noti che, dunque, per $i \in I^h$, il coefficiente di costo $c_i - \lambda_{1i} + \lambda_{2i}$, nella funzione obiettivo di LAGP1(λ), aumenta di una quantità $2\cdot\theta\cdot x^h_i$, sicuramente positiva, per cui, essendo LAGP1(λ) a minimizzare, sarà meno probabile avere una frequenza x_i positiva alla successiva iterazione.

Si può interpretare questo comportamento notando che tale frequenza è associata ad un pattern scartato nella soluzione di LAGP2(λ), nel senso che il relativo valore v_i è stato scelto pari a zero. In presenza di un coefficiente per v_i non positivo nella funzione obiettivo di LAGP2(λ), ciò vuol dire che non è stato possibile avere $v_i = 1$, senza con ciò comportare uno sforamento del limite T sul massimo numero di stack aperti, o almeno non lo è stato senza con ciò comportare un azzeramento di altre variabili v_k con un miglioramento della funzione obiettivo.

Si noti inoltre che, per i valori $x^h_i = 0$, risulta nullo il coefficiente per la relativa variabile v_i nella soluzione del problema LAGP2(λ^h), e dunque esiste una soluzione ottima per il problema LAGP2(λ^h) con tali v_i pari a zero, dato che il valore di ciascuno di tali v_i non ha effetti sulla funzione obiettivo di LAGP2(λ^h), mentre il valore 0 elimina la restrizione dovuta al vincolo (8.21).

Ancora, se $x^h_i > 0$, il coefficiente di costo della variabile v_i nel problema LAGP2(λ^{h+1}), pari ad $x^{h+1}_i\cdot\left(\lambda^{h+1}_{1i} - \lambda^{h+1}_{2i}\right) = x^{h+1}_i\cdot\left(\lambda^h_{1i} - \lambda^h_{2i} - 2\cdot\theta\cdot x^h_i\right)$, risulta ridotto, rispetto a quanto avveniva per il problema LAGP2(λ^h) all'iterazione precedente, a parità di x_i, cioè considerando il caso $x^{h+1}_i = x^h_i$. Ciò vuol dire che, con maggior probabilità, il pattern i verrà utilizzato, cioè si avrà $v^{h+1}_i = 1$, mentre al precedente step era valso $v^h_i = 0$. L'interpretazione di questo comportamento risiede nel fatto che, se un pattern, pur scartato nel problema

LAGP2(λ^h), viene riconfermato nel problema LAGP1(λ^{h+1}), allora è probabilmente opportuno spingere la ricerca verso sequenze che contengono l'*i*-mo pattern, a discapito di altri.

Inoltre, una importante proprietà consiste nel fatto che se, in qualche iterazione, il coefficiente $x_i \cdot (\lambda_{1i} - \lambda_{2i})$ della variabile v_i è non positivo, allora esso non può tornare positivo in nessuna delle iterazioni successive, dato che, come visto, le differenze $\lambda_{1i} - \lambda_{2i}$ possono solo decrescere.

Se dunque tutti i coefficienti sono inizialmente non positivi, ponendo ad esempio valori iniziali tutti nulli per i moltiplicatori di Lagrange, allora essi resteranno non positivi. Questo viene sfruttato nella soluzione del problema LAGP2(λ). Infatti, in generale, quando si ottimizza il problema LAGP2(λ), è sufficiente restringere l'attenzione ai soli indici *i* per i quali si abbia $x_i > 0$ nella soluzione del problema LAGP1(λ) associato alla corrente iterazione. L'insieme di tali indici è qui denotato con V^h. Infatti, esiste certamente una soluzione ottima di LAGP2(λ), con valori nulli di v_i in corrispondenza degli altri indici, analogamente a quanto affermato circa il vincolo (8.21). Si ottiene dunque un problema ridotto.

Avendo però tutti coefficienti non positivi, si può tentare di risolvere il problema LAGP2(λ) ridotto, verificando se è possibile avere ad uno tutti i valori v_i interessanti, cioè associati agli indici *i* per cui $x_i > 0$. Questo può essere fatto ottimizzando il problema MOSP che si ottiene a partire dal LAGP2(λ) ridotto, ponendo nel vincolo (8.21) valori $v_i = 1$. In realtà è sufficiente provare ad ottenere una soluzione con al più *T* stack aperti, sfruttando così le conseguenze di cui si è già discusso nel paragrafo 7.2.4 sulle semplificazioni delle tecniche di soluzione per il MOSP.

Yanasse e Pinto Lamosa indicano che, se una soluzione buona per il MOSP viene trovata, allora si è di fatto trovata una soluzione fattibile al problema originario CSP, dato che tutti i pattern selezionati tramite LAGP1(λ), cioè per i quali valga $x_i > 0$, sono stati sequenziati in LAGP2(λ). Se i corrispondenti valori degli elementi di λ fossero tutti non negativi, la soluzione sarebbe anche ottima per CSP.

Se, al contrario, non si trova alcuna soluzione buona per il MOSP, vuol dire che l'ottimo per LAGP2(λ) non genera una soluzione fattibile per CSP, dato che i pattern con $x_i > 0$, ma $v_i = 0$, vengono scartati dalla sequenza e le domande dei tipi di pezzi non sono più soddisfatte.

Yanasse e Pinto Lamosa procedono allora, come spiegato nel seguito, nella ricerca della soluzione ottima di LAGP2(λ) ridotto, per poi poter iterare il procedimento di aggiornamento dei moltiplicatori. Supponendo di ordinare tutte le possibili combinazioni dei valori binari delle variabili v_i, per valore non crescente della funzione obiettivo, si prova a risolvere il relativo MOSP, ottenuto dal LAGP2(λ) ridotto, imponendo i corrispondenti valori v_i nel vincolo (8.21). Appena si trova una combinazione di valori binari tali che il corrispondente MOSP abbia un ottimo non superiore a *T*, quella combinazione di valori binari rappresenta l'ottimo del problema LAGP2(λ) ridotto, dato che per tutte le combinazioni di valori cui corrispondeva un valore di funzione obiettivo migliore, la ricerca di una sequenza opportuna ha dato esito negativo. L'approccio alla soluzione del problema LAGP2(λ) ridotto può quindi essere vista come un metodo di enumerazione implicita standard per problemi interi 0-1, con la soluzione di un problema MOSP in ogni nodo del relativo albero di ricerca. Tale tecnica combinata è utilizzata per il fatto che gli scarsi risultati legati al modello MTSP si riflettono in altrettanti scarsi risultati sul problema LAGP2(λ), anche utilizzando risolutori standard quali ILOG CPLEX.

Yanasse e Pinto Lamosa sottolineano inoltre che la soluzione del problema LAGP1(λ), di fatto equivalente al cutting stock standard, può essere troppo onerosa, soprattutto in relazione al fatto che bisogna risolverne uno in ogni iterazione del metodo del sottogradiente. Propongono dunque di utilizzare un rilassamento continuo di LAGP1(λ), denotato con

RLAGP1(λ), esattamente come proposto da Gilmore e Gomory (1961), applicando lo schema di column generation di Gilmore e Gomory di cui si è più volte discusso in precedenza.

Essi fanno inoltre notare che l'unione dei problemi RLAGP1(λ) e LAGP2(λ) corrisponde al rilassamento Lagrangiano del problema RCSP, ottenuto a partire da CSP trascurando l'interezza delle variabili x_i. Dunque il metodo del sottogradiente, applicato alla coppia di problemi RLAGP1(λ) e LAGP2(λ) anziché alla coppia LAGP1(λ) e LAGP2(λ), permette in principio di risolvere il duale Lagrangiano di RCSP.

Tutto ciò implica però che, anche nel caso in cui, in qualche step del procedimento, si dovesse trovare una soluzione del LAGP2(λ) ridotto, con tutti valori $v_i = 1$, non si sarebbe trovata una soluzione di CSP, bensì una soluzione di RCSP. Per passare ad una soluzione di CSP, Yanasse e Pinto Lamosa operano nel seguente modo. Fissano i pattern con $x_i > 0$ nella soluzione ottenuta, nel senso che utilizzeranno solo quelli, sfruttando la sequenza già ottenuta attraverso i valori y_{ik} nella soluzione dell'ultimo LAGP2(λ) ridotto, ovvero del MOSP associato. Ciascuno di tali valori frazionari x_i viene approssimato per difetto, comportando con ciò la non copertura delle domande di uno o più tipi di pezzo, se almeno uno di tali valori non era intero. Viene poi risolto un ultimo problema di Cutting Stock puro, in cui si utilizzano solo i pattern fissati, in modo che si possa sfruttare la stessa sequenza di pattern, con aumento delle frequenze e, quindi, senza alterare il massimo numero di stack aperti. Ovviamente, se si sfrutta la tecnica standard di Gilmore e Gomory, con un rilassamento continuo e poi l'approssimazione per eccesso, di fatto la soluzione ottenuta con rilassamento continuo è quella corrispondente alle parti frazionarie dei valori x_i ottenuti nella soluzione del problema RCSP, per cui in tal caso basta realizzare direttamente l'approssimazione a partire dalla soluzione dell'RCSP. Infatti, utilizzando solo i suddetti pattern, la soluzione ottima per il Cutting Stock ridotto con rilassamento continuo, aggiunta alla soluzione che ha i valori x_i approssimati per difetto, cioè aggiungendo a ciascun x_i il corrispondente valore frazionario ottenuto per il Cutting Stock ridottosi determina una soluzione per l'RCSP. Dunque, all'ottimo del Cutting Stock ridotto e rilassato, corrisponde l'ottimo per l'RCPS.

Nel Box 8.1 si riporta lo schema algoritmico descritto, così come nel lavoro di Yanasse e Pinto Lamosa.

$h = 0$; $\lambda_{1i}^0 = \lambda_{2i}^0 = 0$ per $i = 1, \ldots, n$
While Not Stop **Do**
Begin
Risolvi il problema RLAGP1(λ^h) e determina l'insieme V^h di indici i tali che $x^h_i > 0$
Risolvi il problema RLAGP2(λ^h) imponendo $v_i = 0$ per $i \notin V^h$
If $v_i = 1$ $\forall i \in V^h$ **Then**
E' stata trovata una soluzione di RCSP, da trasformare in una soluzione di CSP
Stop
If Criterio di Stop verificato **Then**
Stop
Determina l'insieme I^h di indici i per i quali $x^h_i > 0$ e $v^h_i = 0$
For all $i \in I^h$ **Do**
Begin
$$\lambda_{1i}^{h+1} = \lambda_{1i}^{h} - \theta^h \cdot x_i^h$$
$$\lambda_{2i}^{h+1} = \lambda_{2i}^{h} + \theta^h \cdot x_i^h$$
End
End

Box 8.1 Schema algoritmico della soluzione del problema combinato

Tornando sull'espressione (8.24) di θ, Yanasse e Pinto Lamosa indicano di utilizzare in ogni iterazione, al posto di *LB*, il valore ottenuto per RLAGP(λ) all'iterazione precedente, approssimato per eccesso. Si sottolinea nuovamente che, se qualche moltiplicatore ha valore negativo, allora questo non è un lower bound per z^*, ma si torna su tale argomento alla fine del paragrafo. Per la prima iterazione, si può utilizzare LAGP1(λ^0), dove λ^0 è il vettore iniziale di moltiplicatori, posti tutti pari a zero, come detto prima. Invece, al posto di z^* viene utilizzato un upper bound, ottenuto con una euristica spiegata nel seguito. Non è chiaro, non essendo indicato esplicitamente, se tale euristica sia sfruttata, con qualche apposita variazione, anche nel caso in cui l'algoritmo del sottogradiente venga fermato secondo uno o più criteri di stop, che possono essere in relazione al massimo tempo computazione, al massimo numero di iterazioni, o alla minima distanza fra l'upper bound ed il lower bound, o ancora alla minima variazione dei moltiplicatori. Infatti, in tali casi, la soluzione migliore trovata per il duale Lagrangiano non è fattibile per il problema CSP.

Per la valutazione dell'upper bound, da utilizzare nell'espressione di θ, la suddetta euristica parte dalla soluzione intera trovata per LAGP1(λ^0), cui corrisponde l'insieme di pattern *i* per i quali $x_i > 0$.

Si costruisce dunque il MOSP graph associato ai pattern selezionati tramite LAGP1(λ^0) e si verifica se esista una sequenza fattibile, cioè con massimo numero di stack aperti non maggiore di *T*. Se esiste, la soluzione trovata è la soluzione ottima di LAGP2(λ^0), fattibile per CSP, quindi ottima per il duale Lagrangiano e per il problema originario CSP, come già spiegato in precedenza, dato che λ^0 ha componenti tutte nulle e quindi non negative.

Nel caso in cui non esista una sequenza fattibile dei pattern selezionati tramite LAGP1(λ^0), si verifica quale sia il massimo clique presente nel MOSP graph, cioè il massimo sottoinsieme di nodi collegati l'un l'altro da un arco. Si noti che a ciascun pattern, che coinvolga almeno due gruppi di pezzi, corrisponde un clique relativo ai nodi associati ai gruppi coinvolti da quel pattern. Del resto esiste almeno un pattern con più di un gruppo di pezzi, altrimenti esisterebbe una sequenza con massimo numero di stack aperti pari ad uno.

La ricerca del clique massimo K avviene con una euristica che verifica il minimo grado dei nodi del grafo complementare. All'interno del clique massimo di associa un costo a ciascun arco, pari allo scarto sull'insieme di pattern ottenuto eliminando dall'insieme iniziale di pattern quelli che contengono pezzi dei due gruppi associati ai nodi su cui incide l'arco.

Questa idea è simile a quella della prima euristica di Pileggi, che però aggiunge lo scarto dei pattern omogenei necessari a coprire la domanda dei pezzi liberati. L'arco cui corrisponde lo scarto massimo viene selezionato e tutti i relativi pattern vengono eliminati, inserendoli in una apposita *TabuList*. Per l'insieme residuo di pattern, che dunque non copre completamente le domande dei pezzi, potrebbe esistere una sequenza che determina un massimo numero di stack aperti non maggiore di *T*. Comunque, prima di verificare l'esistenza di una tale sequenza, si procede nella ricerca dei clique massimi sul grafo residuo, ottenuto cioè dall'eliminazione dell'arco prima selezionato. Finché esiste un clique massimo con numero di nodi superiore a *T*, si seleziona un arco al suo interno, con lo stesso criterio di prima, per poi eliminarlo, eliminando anche i pattern ad esso associati ed inserendoli nella *TabuList*. Si noti che l'esistenza di un clique con un numero di nodi qualsiasi *q* non implica necessariamente che vi sia un pattern che coinvolge tutti i *q* gruppi di pezzi associati ai *q* nodi. La riduzione del clique massimo ha però lo scopo di aumentare le probabilità di trovare una sequenza fattibile per l'insieme residuo di pattern.

Non appena si sia ottenuto un grafo ridotto con clique massimo con più di *T* nodi, si verifica l'esistenza di una sequenza fattibile. Se non viene trovata, si verifica nuovamente quale sia il clique massimo e si seleziona nuovamente un arco secondo lo stesso criterio di prima, eliminando sia l'arco che i relativi pattern, inseriti nella *TabuList*. Questo

procedimento viene iterato, indipendentemente dal numero di nodi del clique massimo, finché non si ottiene una sequenza fattibile.

Una volta ottenuta una sequenza fattibile, la rimozione dei pattern avrà determinato la non copertura delle domande dei pezzi. Viene dunque risolto nuovamente un problema di Cutting Stock, considerando l'intera domanda di pezzi, proibendo tutti i pattern eliminati, ed utilizzando il procedimento di column generation di Gilmore e Gomory sul rilassamento continuo. La base del simplesso è inizializzata con le variabili associate ai pattern residui. In realtà non è indicato esplicitamente come venga completata la base nel caso in cui, per qualche gruppo di pezzi, nessuno dei pattern rimasti ne contenga. Probabilmente, in tale caso particolare, vengono aggiunti pattern omogenei, sfruttati come visto prima anche nella prima euristica di Pileggi (2002), dichiarata tra l'altro, da Yanasse e Pinto Lamosa, simile a questa. Si noti che i pattern omogenei non possono essere proibiti, dato che la loro aggiunta in una qualsivoglia presenza non può far aumentare in nessun caso il massimo numero di stack aperti. Del resto, i pattern proibiti contengono pezzi di almeno due diversi gruppi.

La risoluzione del problema di Cutting Stock determina la selezione di un nuovo insieme di pattern. Per tale nuovo insieme di pattern, si costruisce ex-novo un MOSP graph, come fatto inizialmente per l'insieme di pattern selezionato dalla risoluzione di RLAGP1(λ^0) che rappresenta, di fatto, il problema di Cutting Stock sull'intera domanda di pezzi e senza alcuna proibizione.

Si replica sul nuovo insieme di pattern lo stesso procedimento già realizzato sull'insieme iniziale selezionato da LAGP1(λ^0), iterando la generazione di nuovi insiemi di pattern finché non se ne trovi uno per il quale esista una sequenza fattibile. Tutti i pattern che entrano nella *TabuList* in una qualsiasi delle iterazioni, vi permangono per tutto il resto dell'esecuzione dell'euristica.

Non appena si trova una sequenza fattibile, le iterazioni terminano e si realizza una procedura di ricerca locale, che verifica se il reinserimento dei pattern della *TabuList* può determinare un miglioramento dello scarto, a patto di continuare ad avere una sequenza fattibile. Il Box 8.2 contiene lo schema algoritmico dell'euristica descritta, come riportato da Yanasse e Pinto Lamosa. L'insieme *S* è l'insieme di pattern correntemente selezionati, mentre *V* rappresenta il valore della soluzione corrente.

TabuList = ∅ ; *V* = +∞ ; *S* = insieme associato all'ottimo di LAGP1(λ^0)
While Not Stop **Do Begin**
    If Esiste una sequenza fattibile per *S* **Then**
        Aggiorna *V* ; Esegui ***LocalSearch***(*TabuList*) ; Stop
    Costruisci il MOSP graph G e cerca il massimo clique K
    Repeat
        Seleziona l'arco di K associato allo scarto minimo
        Elimina l'arco da G ed i corrispondenti pattern da *S*, inserendoli nella *TabuList*
        Cerca il nuovo massimo clique K
    Until |K| ≤ *T*
    While Not Esiste una sequenza fattibile per *S* **Do**
    Begin
        Cerca il massimo clique K di G
        Seleziona l'arco di K associato allo scarto minimo
        Elimina l'arco da G ed i corrispondenti pattern da *S*, inserendoli nella *TabuList*
    End
    Risolvi il Cutting Stock proibendo i pattern di *TabuList* e partendo da una base corrispondente all'insieme *S*. La soluzione ottenuta identifica il nuovo insieme *S*
End

Box 8.2 Schema algoritmico dell'euristica

La procedura di ricerca locale prova a liberare un pattern per volta, risolvendo nuovamente il problema del Cutting Stock con la *TabuList* così ridotta, cioè senza il pattern liberato. Se per i pattern così selezionati, che determinano uno scarto certamente non maggiore del precedente, esiste una sequenza fattibile, e se lo scarto risulta ridotto, allora si lancia ricorsivamente la procedura di ricerca locale per poi terminare. In caso contrario, il pattern è nuovamente aggiunto alla *TabuList*, e si procede con la verifica per il pattern successivo, procedendo allo stesso modo finché non si ottenga un miglioramento con sequenza fattibile o non terminino i pattern da verificare. Si noti che se un pattern i_1 viene reinserito nella *TabuList*, ma per un pattern successivo i_2 si ottiene un miglioramento, allora la chiamata ricorsiva della procedura di ricerca locale potrebbe determinare la liberazione del pattern i_1, poiché la contemporanea liberazione dei pattern i_1 ed i_2 potrebbe determinare un miglioramento dello scarto e con relativa sequenza fattibile.

Il Box 8.3 riporta lo schema algoritmico della procedura di ricerca locale.

Procedure *LocalSearch*(*TabuList*)
$k = 1$
While ($k \leq |TabuList|$) **and Not** Stop **Then**
Begin
Libera il k-mo pattern di *TabuList*
Risolvi il Cutting Stock proibendo i pattern di *TabuList* e partendo da una base corrispondente all'insieme *S*. La soluzione ottenuta è identificata con l'insieme di pattern S' ed il suo valore è denotato con V'.
If $V' < V$ ed esiste una sequenza fattibile per S' **Then**
Aggiorna $S = S$; Aggiorna $V = V$; Esegui ***LocalSearch***(*TabuList*) ; Stop
Else
Reinserisci il pattern nella *TabuList* in k-ma posizione
$k = k + 1$
End

Box 8.3 Procedura di ricerca locale

Come nota, relativa alle prove sperimentali di Yanasse e Pinto Lamosa, essi utilizzano inizialmente, come *LB* nell'espressione (8.24), il valore $\lceil \text{RLAGP1}(\lambda^0) \rceil$. Calcolano poi UB attraverso l'euristica, come spiegato sopra, e nella (8.24) utilizzano, al posto di z^*, un valore z_{target}, inizialmente posto pari al minimo fra *UB* ed $1.01 \cdot LB$, ed aggiornato ogni qual volta un nuovo valore di *LB*, ottenuto come $\lceil \text{RLAGP1}(\lambda^h) \rceil$, risulti superiore di z_{target}, ponendo nuovamente z_{target} pari al minimo fra *UB* ed $1.01 \cdot LB$. Come sottolineato prima, se qualche componente di λ^h è negativa, allora RLAGP1(λ^h) non è un lower bound per RCSP, ma si ovvia a questo imponendo z_{target} superiore ad *LB*, tanto è vero che viene detto esplicitamente, nel lavoro di Yanasse e Pinto Lamosa, che il nuovo valore di *LB* può superare *UB*.

Il valore della costante ϕ è invece inizializzato a due, per poi essere ridotto ogni qual volta che la soluzione non cambia per $1.5 \cdot n$ iterazioni. La riduzione è realizzata dimezzando ϕ se superiore a 0.5, o elevandone il valore al quadrato, in caso contrario, e seguendo dunque la sequenza 2, 1, 0.5, 0.25, 0.0625, e così via. Come criterio di stop nel procedimento di risoluzione del Lagrangiano duale, si utilizza un massimo numero di iterazioni pari a mille.

8.2 Approcci originali

Questo paragrafo descrive due approcci originali alla combinazione di Cutting e Sequencing. Il primo di essi si basa sull'adattamento della Boundary Disposition Heuristic proposta nel terzo Capitolo. Il secondo sviluppa invece un'idea metodologica originale, secondo cui una buona euristica può consistere nel pre-generare più cutting pattern di quanto strettamente necessario, per poi combinare la selezione di quelli da utilizzare con la ricerca di una sequenza fattibile in termini di massimo numero di stack aperti.

8.2.1 Un approccio basato sulla Boundary Disposition Heuristic

Il primo approccio originale consiste nell'adattamento della procedura euristica basata sulle Boundary Disposition, di seguito denotata con l'acronimo BDH. Nel contesto del terzo Capitolo, in relazione alle prove sperimentali, si è già discusso del suo adattamento al caso multi-stock, con il solo obiettivo di minimizzazione dello scarto.

Quel primo adattamento è stato basato su una alterazione dei profitti associati ai tipi di pezzi, aumentando il profitto specifico (cioè il rapporto tra profitto ed area) dei pezzi più grandi, e riducendo quelli dei pezzi più piccoli, in relazione al numero ideale di stock necessari per l'estrazione dei pezzi, ottenuto come rapporto frazionario fra la somma delle aree dei pezzi da estrarre e l'area di ogni singolo stock, supponendo cioè di trattare il caso di stock identici.

In particolare, al pari di metodi costruttivi greedy come quello di Hinxman (1980), l'adattamento prevede, come già spiegato nel terzo Capitolo, la costruzione di uno schema di taglio per volta. Prima della ricerca dello schema di taglio, si valuta il parametro Γ, attraverso l'espressione $\left(\sum_{j=1}^{n} l_j \cdot w_j \cdot d_j \right) \Big/ \left(L_S \cdot W_S \right)$, che cresce al crescere dell'area totale dei pezzi da estrarre, dato che l_j , w_j e d_j, sono, rispettivamente, le dimensioni e la domanda del j-mo degli n tipi di pezzi, mentre L_S e W_S sono le dimensioni degli stock.

Il profitto π_j per ciascuno dei tipi di pezzi viene quindi alterato elevandolo ad una potenza pari a $(2 - 1 / \Gamma)$. Nel terzo Capitolo, questo è stato fatto per il caso di problema non pesato, cioè utilizzando l'espressione $\pi_j = \left(l_j \cdot w_j \right)^{2-1/\Gamma}$ (approssimata al centesimo), dato che $l_j \cdot w_j$ rappresenta il profitto associato in caso di problema non pesato single-stock.

L'obiettivo che ci si prefigge con il nuovo adattamento è ottenere un insieme C di cutting pattern cui sarà probabilmente associata una sequenza con un basso numero di stack aperti, che si suppone debba essere inferiore ad un massimo. In pratica, tale parametro corrisponde al valore di T nell'approccio di Yanasse e Pinto Lamosa. Laddove, alla sequenza dei pattern di C che minimizzi il massimo numero di stack aperti, corrisponda un massimo superiore al limite T, l'insieme di cutting pattern trovato può essere utilizzato come punto di partenza per le fasi di feed-back di uno degli approcci di letteratura sopra descritti. Eventualmente si può continuare ad utilizzare la BDH per la generazione di nuovi insiemi.

Infatti, a seconda dell'approccio utilizzato fra quelli prima descritti, è necessario vietare alcune combinazioni di gruppi di pezzi su uno stesso cutting pattern. Questo può essere ottenuto, evitando, durante l'esecuzione della BDH single-stock, di aggiungere pezzi di un determinato gruppo, se per tutti gli altri gruppi presenti in un insieme proibito vi è almeno un pezzo già aggiunto. Laddove la tecnica di letteratura utilizzata preveda di proibire un pattern, si può tradurre ciò nella proibizione dell'insieme di gruppi di pezzi coinvolti. Del resto, anche nei metodi di letteratura, sarebbe bene evitare pattern equivalenti da questo punto di vista, che magari contengono meno pezzi di un gruppo e più di altri, senza cambiare la

sostanza in relazione alla difficoltà di essere utilizzati in sequenze fattibili per il massimo numero di stack aperti.

Ricordando i passi principali della BDH, tale limitazione è ottenibile basandosi sul fatto che la costruzione delle soluzioni avviene aggiungendo un pezzo alla volta nello sviluppo delle boundary disposition. D'altro canto, il calcolo di lower bounds per ogni nodo dell'albero di ricerca sviluppato nella BDH, avviene costruendo soluzioni fattibili, attraverso l'inserimento di strisce (strip) di pezzi, sviluppate all'inizio dell'algoritmo. Tali strip, in ciascun nodo, vengono eventualmente private di pezzi, secondo la riduzione corrente della domanda non ancora soddisfatta per il tipo (di pezzi) associato.

Si può dunque procedere con la seguente modalità. Se c'è un solo insieme proibito G di gruppi, si verifica per quali dei suoi gruppi non siano presenti pezzi nelle boundary disposition già costruite. Fra questi, si seleziona il gruppo con la minor somma dei rapporti fra profitto ed area dei suoi pezzi. Più in dettaglio, se J_g è l'insieme di indici che identifica i tipi di pezzi del gruppo g, si seleziona il gruppo per cui sia minimo il valore $\delta_0(g)$ ottenuto con l'espressione $\sum_{j \in J_g} \left(\frac{\pi_j}{l_j \cdot w_j} \cdot d_j \right)$.

Nel caso in cui vi siano più insiemi proibiti G, è necessario in generale selezionare più gruppi di pezzi, in modo da selezionarne almeno uno per ciascuno degli insiemi proibiti, profilandosi dunque un problema di *set covering*. Più precisamente, per ogni insieme G, si considera, come prima, il sottoinsieme G^* di gruppi per i quali nessun pezzo sia già stato utilizzato nella costruzione delle boundary disposition correntemente presenti sullo stock. Si devono allora "coprire" i sottoinsiemi G^* e lo si fa nel seguente modo greedy.

Se $\psi_1(g)$ è il numero di insiemi in cui è coinvolto un gruppo g, allora si associa a ciascun gruppo il valore $\delta_1(g)$ ottenuto come risultato dell'espressione generalizzata $\frac{1}{\psi_1(g)} \sum_{j \in J_g} \left(\frac{\pi_j}{l_j \cdot w_j} \cdot d_j \right)$. Si seleziona allora il gruppo g_1 cui sia associato il minimo valore di $\delta_1(g)$. Questo determina la "copertura" di $\psi_1(g_1)$ insiemi di gruppi proibiti. Se restano altri insiemi proibiti, si valuta il parametro $\psi_2(g)$ per tutti i gruppi identificati dall'unione di tali insiemi, trascurando la copertura dei $\psi_1(g_1)$ insiemi già coperti. Dunque, se almeno uno degli insiemi prima potenzialmente coperti da un generico gruppo g è stato già coperto dalla selezione di g_1, allora vale la relazione $\psi_2(g) < \psi_1(g)$. Ai nuovi valori $\psi_2(g)$ corrisponde l'introduzione dei relativi valori $\delta_2(g)$, definiti analogamente ai valori $\delta_1(g)$. Viene quindi selezionato il gruppo g_2 cui sia associato il minimo valore di $\delta_2(g)$. Si procede allo stesso modo, finché vi siano insiemi non contenenti alcuno dei gruppi selezionati nei precedenti step.

Una volta realizzata la selezione di q gruppi $\{g_1, g_2, \ldots, g_q\}$, che determinano la copertura di tutti i sottoinsiemi G^*, si procede alla selezione di altri gruppi, per evitare di costruire una soluzione con più di T gruppi. A tal fine, si verificano quali siano i K gruppi con pezzi già correntemente utilizzati nelle boundary disposition. Se, considerando i soli gruppi non ancora utilizzati, e trascurando i q gruppi prima selezionati, in totale vi sono N gruppi, con $N > T - K$, è allora necessario selezionare altri $N - (T - K)$ gruppi da proibire, e questo viene fatto selezionando un gruppo per volta secondo valori $\delta_0(g)$ non decrescenti.

Per tutti i gruppi così selezionati, i loro pezzi vengono eliminati dalle strip e viene quindi realizzato il calcolo del lower bound. E' anche possibile, per ciascuna strip, rimpiazzare momentaneamente i "buchi" così causati, attraverso l'inserimento di pezzi di gruppi non proibiti.

A ciascun lower bound ottenuto con questa tecnica viene fatto corrispondere, nel caso in cui vi sia stato almeno un gruppo proibito, il calcolo di un secondo lower bound. Si ricorda

infatti che in ogni nodo dell'albero di ricerca della BDH, la soluzione fattibile, il cui valore è utilizzato come lower bound, è costruita in modo greedy, riempiendo con delle strip le aree rettangolari in cui l'area libera di ciascun rettangolo residuo viene divisa, attraverso una coppia di disposition cuts. Inoltre, il riempimento di ciascuna di tali aree rettangolari, avviene con la tecnica strip heuristic proposta da Hifi e Zissimopoulos (1997), che pure è di natura greedy. Il secondo lower bound è allora costruito secondo la seguente tecnica, che si basa sul fatto che si aggiunge una strip per volta. Prima di scegliere la successiva strip da aggiungere, si verifica se eliminare pezzi da ciascuna possibile strip. L'eliminazione avviene solo se, considerando i pezzi già presenti nelle boundary disposition ed i pezzi delle strip già inserite, l'aggiunta della strip determina la presenza di almeno un pezzo per ciascuno dei gruppi di un qualunque insieme proibito, oppure determina la presenza di pezzi di almeno T gruppi. In caso di eliminazione, per ciascuna strip si opera la selezione dei gruppi come spiegato prima, considerando però solo gli insiemi G^* per cui la strip determinerebbe il non soddisfacimento della proibizione.

Si noti che, per realizzare l'adattamento spiegato del calcolo dei lower bound, si può sfruttare una tecnica incrementale, dato che non è necessario riverificare le eliminazioni ogni volta da capo, ma solo quando si aggiungano pezzi di gruppi ancora non presenti e solo per le strip che ne contengono pezzi.

Relativamente agli upper bound, essi vengono calcolati come se non vi fosse alcun limite in relazione agli obiettivi di Sequencing. Questo infatti rafforza la loro validità, in quanto aggiunge un ulteriore gap fra le condizioni reali e le condizioni ideali cui è associato il loro calcolo.

Un'altra caratteristica dell'adattamento della BDH risiede nella alterazione dinamica dei profitti dei pezzi in relazione, per semplicità, a due soli elementi dell'algoritmo. Come detto in precedenza, è già presente una alterazione fra ciascun cutting pattern ed il successivo. Qui si aggiunge l'alterazione dei profitti da utilizzare sia come prima componente del bound $UB_c(R, i)$, per l'ordinamento dei corner type, sia nella procedura di Suliman (2001), spiegata nel paragrafo 3.3.2, per l'ordinamento dei tipi di pezzi.

In particolare, relativamente a questo secondo aspetto, si utilizza, per ciascun tipo di pezzi, il profitto alterato $\pi^*_j = \pi^{*(1+\eta)}_j$, dove η è la porzione, in termini di somma di aree, dei pezzi già utilizzati del gruppo cui appartiene il tipo. Questo vuol dire che per gruppi i cui pezzi determinano una somma di aree piccola, l'aggiunta di uno o pochi pezzi determina una crescita veloce del parametro η. Del resto, in presenza di gruppi con tanti pezzi, o con pezzi grandi, è molto probabile che essi siano distribuiti su molti stock, anche con differenti schemi di taglio, per cui può servire a poco accrescerne la probabilità di utilizzo, se non si vuole rischiare di far crescere troppo lo scarto totale, nell'ottica multi-stock. Ed infatti, per tale seconda classe estrema di gruppi, il parametro η cresce lentamente.

8.2.2 Un approccio euristico basato su sovragenerazione di cutting pattern

Il secondo approccio euristico proposto trae spunto dalla seguente considerazione. Ad eccezione del metodo proposto da Belov (2003) e basato sul modello lineare già esposto, tutti gli altri metodi di letteratura descritti procedono secondo una delle due seguenti metodologie. La prima prevede la scissione del problema combinato di Cutting e Sequencing, iterando con dei feed-back fra l'uno e l'altro e risolvendo ogni volta il problema di Cutting Stock in cui siano proibiti dei pattern o combinazioni di gruppi di pezzi su ciascun pattern.

La seconda metodologia, al pari dell'euristica proposta nel paragrafo precedente, prevede di costruire in modo greedy un insieme di cutting pattern per poi sequenziarli, eventualmente guidando la procedura costruttiva, come nei casi della SVC modificata di Belov e della seconda euristica di Pileggi (2002).

Negli approcci in cui si utilizza un ciclo iterativo, in ogni iterazione si costruisce un nuovo insieme di cutting pattern, imponendo delle proibizioni, per tentare di superare le difficoltà trovate all'iterazione precedente nel trovare una sequenza che soddisfi il vincolo sul massimo numero di stack aperti. Lo svantaggio consiste nel fatto che non è detto che le proibizioni fatte siano quelle migliori, Infatti, Armbruster (2002), come spiegato in precedenza, utilizza una tecnica di selezione appositamente mirata a fare buone scelte, mentre, ad esempio, Yanasse e Pinto Lamosa (2007) utilizzano una procedura di ricerca locale per provare a reinserire, uno alla volta, i pattern proibiti.

Qui si propone di utilizzare una idea alternativa, che consiste in una sovragenerazione di cutting pattern. In pratica non si costruisce un insieme minimo di pattern, bensì un insieme aumentato, per cui diversi sottoinsiemi di tale insieme possono coprire la domanda di pezzi, ciascuno utilizzando le opportune frequenze per ogni suo pattern.

A partire dall'insieme di pattern generato, va poi effettuata una ricerca del sottoinsieme con scarto ottimo e con il vincolo di avere una sequenza associata che soddisfi il massimo numero di stack aperti. La selezione ed il sequenziamento devono dunque avvenire contemporaneamente, poiché è ovvio che, se si utilizzassero tecniche con feed-back e proibizioni, si ricadrebbe nelle metodologie già esistenti, con l'aggravante che non tutti i pattern potrebbero essere utilizzati, ma solo quelli dell'insieme inizialmente generato.

Uno dei modi in cui può essere utilizzata l'idea della sovragenerazione consiste nell'utilizzare un modello lineare combinato di Cutting e di Sequencing, senza però ricorrere al rilassamento continuo classicamente utilizzato per i vincoli di Cutting. In genere, tale rilassamento è utilizzato a causa dell'enorme numero di tutti i pattern possibili, come più volte evidenziato nei paragrafi precedenti. Nel caso della metodologia proposta, invece, questo fattore viene meno.

Nel seguito si descrive l'euristica proposta, che ingloba la suddetta idea innovativa. In principio, essa dovrebbe svilupparsi su due sole fasi, una di generazione dei pattern ed una di selezione e sequenziamento degli stessi.

E' però evidente che la fase di costruzione dell'insieme assume in tale ottica un ruolo cruciale. Dunque, piuttosto che sviluppare un siffatto procedimento euristico, si preferisce realizzare un ciclo iterativo secondo lo schema spiegato nel seguito, in cui, inoltre, si utilizza nuovamente la Boundary Disposition Heuristic.

Inizialmente si risolve il problema del Cutting Stock, con la tecnica greedy che genera un cutting pattern per volta, con la procedura BDH multi-stock e secondo le modalità spiegate nella prima parte del precedente paragrafo.

Sull'insieme di cutting pattern così generato, si applica la ricerca di una sequenza che minimizzi il massimo numero di stack aperti. Si noti che, in questa fase, non c'è ancora stata sovragenerazione, per cui si possono utilizzare le tecniche esatte descritte nel precedente Capitolo. In particolare si utilizza una combinazione degli algoritmi di Becceneri, Yanasse e Soma (2004) e di Yanasse e Limeira (2004), sfruttando le riduzioni del problema proposte nel primo lavoro e le decomposizioni del problema proposte nel secondo.

Se esiste una sequenza che rispetta il massimo numero T di stack aperti, allora l'algoritmo termina. In caso contrario, si considera l'insieme di stack aperti a valle di ciascuno dei pattern della sequenza ottima. Almeno uno di essi ha cardinalità superiore a T. Tutti quelli di cardinalità superiore a T identificano un insieme di gruppi di pezzi. Si riesegue allora la BDH multi-stock, proibendo tutti tali insiemi di gruppi, secondo le modalità spiegate nella seconda parte del paragrafo precedente. Si noti che, poiché la proibizione è applicata secondo questo semplice meccanismo di soglia, la ricerca di una sequenza buona può avvalersi delle semplificazioni più volte sottolineate nel precedente Capitolo, circa il fatto che, laddove si trovi una sottosequenza, nelle opportune condizioni, con massimo numero di stack aperti inferiore a T, allora ci si può accontentare senza spingere la ricerca verso ottimi parziali. Ad

esempio, questo avviene se per una delle componenti connesse del MOSP graph si trova una sottosequenza di stack con al più $T - 1$ stack aperti.

Si evidenzia il fatto che la proibizione applicata è di natura diversa rispetto a quelle dei metodi di letteratura, nel senso che si tratta di proibizioni temporanee, il cui obiettivo è determinare la generazione di pattern che, in termini di Sequencing, sono probabilmente in grado di determinare sequenze diverse rispetto a quella trovata alla prima iterazione, ed almeno altrettanto buone in termini di massimo numero di stack aperti. In ogni caso, la BDH multi-stock genera un secondo insieme di cutting pattern, che si aggiunge a quello generato alla prima iterazione.

Da questo momento in poi, qualora siano necessarie altre iterazioni, l'insieme di cutting pattern viene fatto crescere senza mai eliminare alcuno dei pattern già generati. Inoltre, dopo ogni generazione di un nuovo insieme di pattern, si applica una fase di selezione e sequenziamento, mentre dopo la prima iterazione viene, come già detto, eseguita la sola ricerca di una sequenza, cioè senza selezione, dato che l'insieme di pattern a disposizione è "minimo".

Per ottenere selezione e sequenziamento in modo combinato, si sfrutta il modello (8.3) proposto da Belov (2003) e precedentemente descritto, apportando però opportune modifiche per tener conto sia dei presupposti differenti, sia della generalizzazione secondo cui un gruppo di pezzi può essere costituito da diversi tipi di pezzi.

In particolare, si utilizzano le costanti n ed m, per rappresentare, rispettivamente, il numero di tipi di pezzi ed il numero dei gruppi di pezzi. Per ciascun gruppo di indice generico l, da 1 ad m, si denota con J_l l'insieme degli indici dei tipi di pezzi associati, mentre, per ciascun tipo di indice generico j, si denota con $g(j)$ l'indice del gruppo associato. La costante C denota invece il numero di pattern selezionati fino alla corrente iterazione.

Il modello (8.3) di Belov, tramite i vincoli (8.3b), (8.3c) ed (8.3d), impone il soddisfacimento esatto delle domande di pezzi. Questo implica che, utilizzando i soli pattern correntemente generati, è improbabile riuscire a combinare pattern generati in iterazioni differenti. Per ovviare a questo difetto, e traendone anche un vantaggio in termini di numero di pattern virtualmente selezionabili, si può permettere di utilizzare pattern ridotti. Se ad esempio p_{ij} rappresenta il numero di pezzi di tipo j presenti sull'i-mo pattern (cioè con ruolo degli indici invertito rispetto al modello (8.3)), si può pensare di utilizzare una corrispondente variabile intera q_{ij}, da imporre non maggiore di p_{ij}, e da utilizzare al posto di p_{ij} nei suddetti tre vincoli di domanda. In tal modo si possono utilizzare tutti i pattern, ottenuto da uno di quelli selezionati, eliminando da esso dei pezzi. Ciò determina però la non linearità del modello, come del resto era il modello (8.3) in cui già le p_{ij} erano variabili e non costanti. L'analogo del passaggio al modello (8.4), proposto da Belov proprio per passare ad un modello lineare, in questo caso richiede l'uso di un elevato numero di variabili, a causa del fatto che vanno introdotte delle variabili binarie che rappresentano la selezione dei pattern ed il loro posizionamento all'interno della sequenza. Nel modello di Belov, la genericità dei pattern utilizzabili non faceva emergere questo impedimento, e del resto si aveva a che fare in partenza con un numero di variabili elevato, soprattutto in caso di problemi bidimensionali. Nel seguito, comunque, si propongono due modelli lineari, da scegliere a seconda del numero di variabili coinvolte.

Si sottolinea pure che la possibilità di eliminare pezzi da un pattern fa aumentare le probabilità di trovare una sequenza buona, in quanto permette di spezzare dipendenze fra gruppi di pezzi. Dunque fa aumentare le probabilità di chiudere, e con successo, l'algoritmo.

Nel primo modello proposto, si utilizzano la costante LB_S che rappresenta un lower bound sul numero di stock necessari ed UB_{OS} che rappresenta un upper bound sul massimo numero di stack aperti. Il primo può essere valutato all'inizio dell'algoritmo. Il secondo può essere posto, per la prima iterazione, pari al numero m di differenti gruppi di pezzi e poi

essere aggiornato come il minimo ottenuto al variare delle sequenze ricavate per le iterazioni precedenti.

Si utilizza inoltre la costante *S*, che denota il massimo numero di stock da utilizzare. Esso è posto pari al minimo numero di stock che si sa essere necessario, secondo le informazioni ricavate fino alla iterazione corrente. In particolare, è possibile che il suo valore sia aggiornato ogni volta che si riesegue la BDH multi-stock oppure ogni volta che, trovata la selezione e sequenza ottima dei pattern correntemente generati, si verifica la possibilità di soddisfare la domanda con un numero di stock inferiore a quanto precedentemente stabilito.

Se ne ricava dunque il seguente modello, in cui l'espressione del valore S_{new}, nuovo potenziale nuovo valore di *S* per l'iterazione successiva, si ottiene con la somma $\sum_{i=1}^{C}\sum_{k=1}^{S} Q_{ik}$.

$$
\begin{aligned}
&\min && MaxOS && && \\
&s.t. && MaxOS \geq OS_k = \sum_{l=1}^{m}\left(s_{lk} + \underline{e}_{lk} - 1\right) && k = 1,\ ...,\ S && \text{(a)} \\
& && \sum_{i=1}^{C}\sum_{h=1}^{k} q_{ijh} \leq d_j s_{g(j)k} && j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ S && \text{(b)} \\
& && \sum_{i=1}^{C}\sum_{h=k}^{S} q_{ijh} \leq d_j \underline{e}_{g(j)k} && j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ S && \text{(c)} \\
& && \sum_{i=1}^{C}\sum_{k=1}^{S} q_{ijk} \geq d_j && j = 1,\ ...,\ n && \text{(d)} \\
& && \sum_{j=1}^{n} q_{ijk} \geq Q_{ik} \quad ; \quad q_{ijk} \leq p_{ij} \cdot Q_{ik} && i = 1,\ ...,\ C\ ;\ j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ S && \text{(e)} \\
& && \sum_{i=1}^{C} Q_{ik} = 1 && k = 1,\ ...,\ LB_S && \text{(f)} \\
& && \sum_{i=1}^{C} Q_{ik} \leq 1 && k = LB_S + 1,\ ...,\ S && \text{(g)} \\
& && OS_{k+1} \leq UB_{OS} \cdot OS_k && k = LB_S,\ ...,\ S-1 && \text{(h)} \\
& && s_{lk}, \underline{e}_{lk} \in \{0,\ 1\} && k = 1,\ ...,\ S\ ;\ l = 1,\ ...,\ m && \text{(i)} \\
& && q_{ijk} \in \{0, 1, ..., p_{jk}\} && i = 1,\ ...,\ C\ ;\ j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ S && \text{(j)} \\
& && Q_{ik} \in \{0,\ 1\} && i = 1,\ ...,\ C\ ;\ k = 1,\ ...,\ S && \text{(k)}
\end{aligned}
\qquad (8.25)
$$

La generica variabile q_{ijk} rappresenta il numero di pezzi di tipo *j* utilizzati nel pattern posto come *k*-mo nella sequenza, supponendo di partire dall'*i*-mo pattern. Questa moltiplicazione di variabili, con ben tre indici, è stata necessaria per inglobare la rappresentazione della sequenza di pattern e mantenere il modello lineare. Ad ogni insieme di variabili q_{ijk} , al variare di *j*, corrisponde la variabile binaria Q_{ik} che vale 1 se si utilizza una replica dell'*i*-mo pattern nella *k*-ma posizione della sequenza.

Si noti pure che, nel vincolo (8.25g) si presuppone la possibilità di utilizzare non più di LB_S differenti stock. Per imporre però di avere posizioni "vuote" solo se consecutive ed in coda, si è aggiunto il vincolo (8.25h), in cui si sfruttano le apposite variabili OS_k, che identificano il numero di stack aperti alla processazione del *k*-mo step della sequenza di pattern.

Si sottolinea che, a differenza che nel caso di Belov, il processo che ha portato a questo modello è stato appositamente pensato per il problema bidimensionale, per cui era necessario utilizzare, per le variabili q_{ijk} , dei limiti (rappresentati dalle costanti p_{ij}) tali che i

cutting pattern selezionati fossero certamente fattibili. Nel modello (8.4) di Belov, la fattibilità per il caso mono-dimensionale, era semplicemente espressa dall'analogo del vincolo (8.3e).

La difficoltà nell'affrontare la soluzione del modello (8.25) consiste nel fatto che il numero di variabili q_{ijk} cresce con il prodotto $C \cdot n \cdot S$. Qualora tale numero di variabili sia eccessivo, si utilizza una logica che medi fra le diverse esigenze emerse. Dal punto di vista del soddisfacimento esatto della domanda, si permette di avere pochi pattern ridotti e tutti in coda alla sequenza. D'altro canto, il più delle volte questo è sufficiente allo scopo.

Invece, per permettere di semplificare le dipendenze fra i pezzi, e rafforzare la capacità di soddisfare esattamente la domanda utilizzando gruppi generati in iterazioni differenti, si aggiungono i seguenti pattern. Per ciascun pattern correntemente generato, di indice generico i, che coinvolge un insieme di gruppi denotato con G_i, si utilizza un pattern per ciascuno degli elementi l di G_i, ottenuto eliminando tutti i pezzi del gruppo di indice l. Questo determina, in principio, un numero di pattern pari a $C \cdot (m + 1)$. Il numero di effettivi pattern è denotato con C_m. Ad essi si aggiungono i potenziali pattern ridotti.

Nei vincoli di domanda si utilizzano variabili di frequenza intere generali, per ciascuna delle quali si dichiara una corrispondente variabile intera binaria, utilizzata nei vincoli che stabiliscono la fattibilità della sequenza in quanto tale.

Sia R il numero totale di pattern ridotti permessi e sia D il massimo numero di cutting pattern fra quelli con frequenza libera. Ovviamente, è inutile che il valore di D superi il minimo fra i due valori C_m ed S.

Il secondo modello proposto è quindi il seguente (8.26). La generica variabile intera x_{ik} identifica la frequenza dell'i-mo pattern, se utilizzato in k-ma posizione. Inoltre, la generica variabile intera y_{ilk} identifica la frequenza del pattern, se utilizzato in k-ma posizione, ottenuto dall'i-mo pattern per eliminazione dei pezzi del gruppo di indice l. A ciascuna di tali variabili corrisponde una variabile intera binaria X_{ik} o Y_{ilk}, con legami simili a quelli fra le variabili q_{ijk} e Q_{ik} del modello (8.25), ma tenendo conto della differente natura.

Le variabili q_{ijk} e Q_{ik} sono presenti anche in questo secondo modello, ma con indici k che vanno da 1 ad R. In pratica ad una soluzione fattibile di questo modello corrispondono due sottosequenze. Una sottosequenza di al più D cutting pattern, ciascuno con una propria frequenza, ed una sottosequenza di al più R cutting pattern ridotti, con frequenza unitaria. E' possibile avere posizioni "vuote" in ciascuna delle due sottosequenze, in modo indipendente, ma solo in coda a ciascuna di esse, secondo i vincoli (8.26l) ed (8.26n). Esse vengono considerate una in coda all'altra, con in testa la sequenza di al più D pattern ed in coda la sequenza di al più R pattern. Questo lo si può evincere dai vincoli (8.26c) ed (8.26d).

La sequenza reale di pattern che corrisponde ad una soluzione del modello (8.26) si ottiene accodando le due sottosequenze l'una all'altra nell'ordine spiegato, per poi rimuovere le posizioni vuote.

I vincoli (8.26i) ed (8.26m) servono a selezionare, in k-ma posizione, al più un solo pattern, mentre i vincoli (8.26j) ed (8.26k) aiutano la soluzione del modello, poiché impongono di utilizzare ciascun possibile pattern con frequenza libera in una sola posizione. Infatti, ai fini della minimizzazione del numero di stack aperti, è inutile usare uno stesso di tali pattern in due posizioni diverse. Nel caso di posizioni consecutive, basterebbe sommare le due frequenze associate, mentre nel caso di posizioni non consecutive, si può solo ottenere un peggioramento del massimo numero di stack aperti. Si può inoltre notare che, in realtà, dato il vincolo (8.26l), è necessario usare il vincolo (8.26i) solo con l'indice $k = 1$.

Infine, il valore di S_{new} è ottenibile con l'espressione $\sum_{i=1}^{C}\left[\sum_{k=1}^{D} x_{ik} + \sum_{k=1}^{D}\sum_{l \in G_i} y_{ilk} + \sum_{k=1}^{R} Q_{ik}\right]$.

$$\min \quad MaxOS \tag{a}$$

$$s.t. \quad MaxOS \geq OS_k = \sum_{l=1}^{m} \left(s_{lk} + \underline{e}_{lk} - 1 \right) \qquad k = 1,\ ...,\ D+R \tag{b}$$

$$\sum_{i=1}^{C} \left[\sum_{h=1}^{\min\{k,D\}} \left(p_{ij} x_{ih} + \sum_{\substack{l \in G_i \\ l \neq g(j)}} p_{ij} y_{ilh} \right) + \sum_{h=1}^{k-D} q_{ijh} \right] \leq d_j s_{g(j)k} \qquad j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ D+R \tag{c}$$

$$\sum_{i=1}^{C} \left[\sum_{h=\min\{k,D\}}^{D} \left(p_{ij} x_{ih} + \sum_{\substack{l \in G_i \\ l \neq g(j)}} p_{ij} y_{ilh} \right) + \sum_{h=\max\{k-D,1\}}^{R} q_{ijh} \right] \leq d_j \underline{e}_{g(j)k} \qquad j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ D+R \tag{d}$$

$$\sum_{i=1}^{C} \left[\sum_{k=1}^{D} \left(p_{ij} x_{ik} + \sum_{\substack{l \in G_i \\ l \neq g(j)}} p_{ij} y_{ilk} \right) + \sum_{k=1}^{R} q_{ijk} \right] \geq d_j \qquad j = 1,\ ...,\ n \tag{e}$$

$$x_{ik} \geq X_{ik} \quad ; \quad x_{ik} \leq S \cdot X_{ik} \qquad i = 1,\ ...,\ C\ ;\ k = 1,\ ...,\ D \tag{f}$$

$$y_{ilk} \geq Y_{ilk} \quad ; \quad y_{ilk} \leq S \cdot Y_{ilk} \qquad i = 1,\ ...,\ C\ ;\ l \in G_i\ ;\ k = 1,\ ...,\ D \tag{g}$$

$$\sum_{j=1}^{n} q_{ijk} \geq Q_{ik} \quad ; \quad q_{ijk} \leq p_{ij} \cdot Q_{ik} \qquad i = 1,\ ...,\ C\ ;\ j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ R \tag{h}$$

(8.26)

$$\sum_{i=1}^{C} X_{ik} + \sum_{i=1}^{C} \sum_{l \in G_i} Y_{ilk} \leq 1 \qquad k = 1,\ ...,\ D \tag{i}$$

$$\sum_{k=1}^{D} X_{ik} \leq 1 \qquad i = 1,\ ...,\ C \tag{j}$$

$$\sum_{k=1}^{D} Y_{ilk} \leq 1 \qquad i = 1,\ ...,\ C\ ;\ l \in G_i \tag{k}$$

$$\sum_{i=1}^{C} X_{i,k+1} + \sum_{i=1}^{C} \sum_{l \in G_i} Y_{il,k+1} \leq \sum_{i=1}^{C} X_{ik} + \sum_{i=1}^{C} \sum_{l \in G_i} Y_{ilk} \qquad k = 1,\ ...,\ D-1 \tag{l}$$

$$\sum_{i=1}^{C} Q_{ik} \leq 1 \qquad k = 1,\ ...,\ R \tag{m}$$

$$OS_{k+1} \leq UB_{OS} \cdot OS_k \qquad k = 1,\ ...,\ R-1 \tag{n}$$

$$s_{lk}, \underline{e}_{lk} \in \{0,\ 1\} \qquad k = 1,\ ...,\ D+R\ ;\ l = 1,\ ...,\ m \tag{o}$$

$$X_{ik} \in \{0,\ 1\}\ ;\ x_{ik} \in \mathbb{N}_0 \qquad i = 1,\ ...,\ C\ ;\ k = 1,\ ...,\ D$$

$$Y_{ilk} \in \{0,\ 1\}\ ;\ y_{ilk} \in \mathbb{N}_0 \qquad i = 1,\ ...,\ C\ ;\ l \in G_i\ ;\ k = 1,\ ...,\ D \tag{p}$$

$$q_{ijk} \in \{0,1,...,p_{jk}\} \qquad i = 1,\ ...,\ C\ ;\ j = 1,\ ...,\ n\ ;\ k = 1,\ ...,\ R \tag{q}$$

$$Q_{ik} \in \{0,\ 1\} \qquad i = 1,\ ...,\ C\ ;\ k = 1,\ ...,\ R \tag{r}$$

Sebbene tale modello sia complesso, va sottolineato che nelle tecniche di letteratura la soluzione veloce di problemi MOSP avviene con elevata frequenza, mentre in tal caso sono generalmente sufficienti non più di cinque iterazioni per ottenere una buona soluzione.

Un elemento di debolezza del modello consiste nella presenza di soluzioni simmetriche, in quanto i pattern con frequenza libera nelle prime *D* posizioni sono replicabili attraverso gli ultimi *R* pattern ridotti.

Relativamente al ciclo delle iterazioni, può in principio accadere che l'esecuzione della BDH multi-stock di una iterazione, possa non produrre alcun cutting pattern non già generato in iterazioni precedenti. Se questo avvenisse in due iterazioni consecutivi, il procedimento andrebbe in loop. Per evitarlo ed ottenere così una diversificazione, si riesegue,

alla seconda di tali esecuzioni, la BDH multi-stock, proibendo tutti i gruppi di pezzi associati a ciascuno dei cutting pattern già generati, eccezion fatta per i pattern che contengono un solo tipo di pezzi. Questo, infatti, comporta la sicura generazione di nuovi pattern, salvo il caso in cui tutti i nuovi pattern generati contengano ciascuno pezzi di un sol gruppo e siano tutti già stati generati. In questa particolare evenienza, tra l'altro mai riscontrata nella pratica, si propone di rieseguire la BDH multi-stock associando per una sola esecuzione dei profitti randomatici ai tipi di pezzi.

Infine si sottolinea che tale evenienza potrebbe accadere a causa dell'elevato numero di iterazioni già realizzate, probabilmente a causa dell'impossibilità di utilizzare il minimo numero di possibile di stock e soddisfare contemporaneamente il vincolo sul massimo numero di stack aperti. Per ovviare a tale ulteriore evenienza, è opportuno limitare il massimo numero di iterazioni proporzionalmente ad m, dopo di che è opportuno utilizzare un approccio esatto per verificare l'eventuale impossibilità di raggiungere l'obiettivo.

In chiusura, si indica che l'idea innovativa di sovragenerazione di pattern può in principio essere utilizzata in vari altri modi, anche molto diversi da quello proposto, e può essere integrato nelle tecniche già esistenti in letteratura, per migliorarne gli step di natura euristica.

BIBLIOGRAFIA

1. Aarts E, Korst JK (1989) Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley & Sons.
2. Aarts E, Lenstra JK (1997) Local Search in Combinatorial Optimization, John Wiley & Sons, Chichester.
3. Alizadeh F, Karp RM, Newberg LA, Weisser DK (1995) Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13, 52-76.
4. Alvarez-Valdés R, Parajon A, Tamarit JM (2002) A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers & Operations Research*, 29, 925-947.
5. Andreatta G, Basso A, Caumo A, Desserti L (1989) Un problema min cutwidth generalizzato e sue applicazioni ad un FMS. *Atti delle giornate di lavoro AIRO*, 1-17.
6. Applegate DL, Bixby RE, Chvàtal V e Cook WJ (2007) The Travelling Salesman Problem: A Computational Study, Princeton University Press.
7. Arenales MN, Cherri A, Yanasse HH (2007) The unidimensional cutting stock problem with usable leftovers – A heuristic approach. In: The 22nd EURO Conference (European Conference on Operational Research), 44, Prague, Czech Republic, July, 8-11.
8. Arenales MN, Morabito R, Yanasse HH (Eds.) (1999) Cutting and packing problems. *Pesquisa Operacional*, 19.
9. Armbruster M (2000) Minimal verschnitt beim zuschnitt von staeben. Ph.D. Thesis, Technische Universitaet Chemnitz.
10. Armbruster M (2002) A solution procedure for a pattern sequencing problem as part of a one-dimensional cutting stock problem in the steel industry. *European Journal of Operational Research*, 141, 328-340.

11. Baker BS, Schwarz JS (1983) Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12, 508-525.
12. Beasley JE (1985) Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36, 297-306.
13. Becceneri JC (1999) The pattern sequencing problem to minimize the maximum number of open stacks in industrial cutting environments. Dissertations, Aeronautic Institute of Technology, Brazil.
14. Becceneri JC, Yanasse HH, Soma NY (2004) A method for solving the minimization of the maximum number of open stacks problem within a cutting process. *Computers & Operations Research*, 31, 2315-2332.
15. Belov G (2003) Problems, models and algorithms in one- and two-dimensional cutting. Ph.D. Thesis, University of Dresda, Germany.
16. Belov G, Scheithauer G (2002) A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141, 274-294.
17. Belov G, Scheithauer G (2006) A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research*, 171, 85-106.
18. Belov G, Scheithauer G (2007) Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS Journal on Computing*, 19, 27-35.
19. Berkley JO, Wang PY (1987) Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38, 423-429.
20. Bischoff EE, Wäscher G (Eds.) (1995) Cutting and packing. *European Journal of Operational Research*, 84, 503-505.

21. Boschetti MA, Montaletti L (2007) Heuristic and exact methods for the strip packing problem. In: The 38th AIRO Annual Conference (Conference of the Italian Operations Research Society), 27, Genova, Italy, September, 5-8.
22. Brooks RL, Smith AB, Stone H, Tutte WT (1940) The dissection of rectangles into squares. *Duke Mathematical Journal*, 7, 312-340.
23. Chazelle B (1983) The bottom-left bin packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, 32, 697-707.
24. Cheng TCE, Diamond J, Lin BMT (1993) Optimal scheduling in film production to minimize talent hold cost. *Journal of Optimization Theory and Applications*, 79, 197-206.
25. Christofides N, Whitlock C (1977) An algorithm for two-dimensional cutting problems. *Operations Research*, 25, 30-44.
26. Chung FKR, Garey MR, Johnson DS (1982) On packing two-dimensional bins. *SIAM Journal of Algebraic and Discrete Methods*, 3, 66-76.
27. Coffman EG, Garey MR, Johnson DS, Tarjan RE (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9, 801-826.
28. CPLEX (http://www.ilog.com/products/cplex/).
29. Csirik J, Woeginger GJ (1997) Shelf algorithms for online strip packing. *Information Processing Letters*, 63, 171-175.
30. Cui Y, Zhang X (2007) Two-stage general block patterns for the two-dimensional cutting problem. *Computers & Operations Research*, 34, 2282-2893.
31. Cung VD, Hifi M, Le Cun B (2000) Constrained two-dimensional cutting stock problems. A best-first branch-and-bound algorithm. *International Transactions In Operational Research*, 7, 185-200.

32. De Giovanni L, Pezzella F, Pfetsch M, Rinaldi G, Ventura P (2007). The open stack problem. In: The 38th AIRO Annual Conference (Conference of the Italian Operations Research Society), 24, Genova, Italy, September, 5-8.
33. De la Banda MG, Stuckey PJ (2007) Dynamic programming to minimize the maximum number of open stacks. *INFORMS Journal on Computing*, 19, 607-617.
34. Degraeve Z, Gochet W, Jans R (2002) Alternative formulations for a layout problem in the fashion industry. *European Journal of Operational Research*, 143, 80-93.
35. Deo K, Krishnamoorty MS, Langston MA (1987) Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer-Aided Design*, 6, 79-84.
36. Desler JF, Hakimi SL (1969) A graph theoretic approach to a class of integer programming problems. *Operations Research*, 17, 1017-1033.
37. Deutsch SB, Martin JJ (1971) An ordering algorithm for analysis of data items. *Operations Research*, 19, 1350-1362.
38. Dowsland KA (1993) Simulated annealing. In: Reeves C (Ed.) Modern Heuristics Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford, 20-69.
39. Dowsland KA, Dowsland WB (1992) Packing problems. *European Journal of Operational Research*, 56, 2-14.
40. Dowsland KA, Dowsland WB (1995) Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84, 506-521.
41. Dyckhoff H (1981) A new linear programming approach to the cutting stock problem. *Operations Research*, 29, 1092-1104.
42. Dyckhoff H (1990) A typology of cutting and packing problems. *European Journal of Operational Research*, 44, 145-159.

43. Dyckhoff H, Finke U (1992) Cutting and Packing in Production and Distribution – A Typology and Bibliography, Physica Verlag, Heidelberg.
44. Dyckhoff H, Scheithauer G, Terno J (1997) Cutting and packing (C&P). In: Dell'Amico M, Maffioli F, Martello S (Eds.) Annotated Bibliographies in Combinatorial Optimization, John Wiley & Sons, Chichester, 393-412.
45. Dyckhoff H, Wäscher G (Eds.) (1990) Cutting and packing. *European Journal of Operational Research*, 44.
46. Dyson RG, Gregory AS (1974) The cutting stock problem in the flat glass industry. *Operational Research Quarterly*, 25, 41-53.
47. Ellis JA, Sudborough IH, Turner JS (1994) The vertex separation and search number of a graph. *Information and Computation*, 113, 50-79.
48. Faggioli E, Bentivoglio CA (1998) Heuristic and exact methods for the cutting sequencing problems. *European Journal of Operational Research*, 110, 564-575.
49. Farley A (1983a) A note on modifying a two-dimensional trim-loss algorithm to deal with cutting restrictions. *European Journal of Operational Research* 14, 393-395.
50. Farley A (1983b) Practical adaptations of the Gilmore-Gomory approach to cutting stock problems. *OR Spektrum*, 10, 113-123.
51. Farley A (1983c) Trim-loss pattern rearrangement and its relevance to the flat-glass industry. *European Journal of Operational Research*, 14, 386-392.
52. Fasano G (2004) A MIP approach for some practical packing problems: Balancing constraints and tetris-like items. *4OR: A Quarterly Journal of Operations Research*, 2, 161-174.
53. Fasano G (2007) MIP-based heuristic for non-standard 3D-packing problems. *4OR: A Quarterly Journal of Operations Research*, on-line doi: 10.1007/s10288-007-0049-1.

54. Fayard D, Hifi M, Zissimopoulos V (1998) An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society*, 49, 1270-1277.
55. Fayard D, Zissimopoulos V (1995) An approximation algorithm for solving unconstrained two-dimensional knapsack problems. *European Journal of Operational Research*, 84, 618-632.
56. Fekete SP, Schepers J (2004) A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations research*, 60, 311-329.
57. Fekete SP, van der Veen JC (2007) PackLib^2: An integrated library of multi-dimensional packing problems. *European Journal of Operational research*, 183, 1131-1135.
58. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6, 109-133.
59. Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42, 860-878.
60. Fink A, Voβ S (1999) Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, 26, 17-34.
61. Foerster H, Wäscher G (1998) Simulated annealing for order spread minimization in sequencing cutting patterns. *European Journal of Operational Research*, 110, 272-281.
62. Foerster H, Wäscher G (2000) Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research*, 38, 1657-1676.
63. Fortin D, Tsevendorj G (2004) Global optimization and multi knapsack: A percolation algorithm. *European Journal of Operational Research*, 154, 46-56.
64. Frenk JB, Galambos GG (1987) Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem. *Computing*, 39, 201-217.

65. G Y-G, Seong Y-J, Kang MK (2003) A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems. *Operations Letters*, 31, 301-307.
66. Garey MR, Johnson DS (1979) Computers and Intractability: a guide to the theory of NP-completeness, Freeman, San Francisco.
67. Gau T, Wäscher G (1995) CUGEN1: A problem generator for the standard one-dimensional cutting stock problem. *European Journal of Operational Research*, 84, 572-579.
68. Gavril F (1977) Some MP-complete problems on graphs. In: Proceedings of the 11th Conference on Information Sciences and Systems, John Hopkins University, Baltimore, Mariland, 91-95.
69. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problems. *Operations Research*, 9, 849-859.
70. Gilmore PC, Gomory RE (1963) A linear programming approach to the cutting-stock problems – Pat II. *Operations Research*, 11, 863-888.
71. Gilmore PC, Gomory RE (1965) Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13, 94-120.
72. Gilmore PC, Gomory RE (1966) The theory and computation of knapsack functions. *Operations Research*, 14, 1045-1074.
73. Glover F, Laguna M (1993) Tabu search. In: Reeves C (Ed.) Modern Heuristics Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford, 70-141.
74. Glover F, Laguna M (1997) Tabu Search, Kluwer Academid Publishers, Boston.
75. Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

76. Gomes AM, Oliveira JF (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171, 811-829.
77. Gradisar M, Resinovic G, Klijajic M (2002) Evaluation of algorithms for one-dimensional cutting. *Computers & Operations Research*, 29, 1207-1220.
78. Haessler R (1980) A note in computational modifications to the Gilmore-Gomory cutting stock algorithm. *Operations Research*, 28, 1001-1005.
79. Han X, Iwama K, Ye D, Zhang G (2007) Strip packing vs. bin packing. In Algorithmic Aspects in Information and Management, Springer Berlin, Heidelberg, 358-367.
80. Herz JC (1972) A recursive computing procedure for two-dimensional stock cutting. *IBM Journal of Research and Development*, 16, 462-469.
81. Hifi M (1994) Study of some combinatorial optimization problems: cutting stock, packing and set covering problems. Ph.D. Thesis, University of Paris 1 Pantheon-Sorbonne.
82. Hifi M (1997a) The DH/KD algorithm: A hybrid approach for unconstrained two-dimensional cutting problems. *European Journal of Operational Research*, 97, 41-52.
83. Hifi M (1997b) An improvement of Viswanathan and Bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers & Operations Research*, 24, 727-736.
84. Hifi (2001) Exact algorithms for large-scale unconstrained two and three staged cutting problems. *Computational Optimization and Applications*, 18, 63-88.
85. Hifi M (Ed.) (2002) Cutting and packing. *Studia Informatica Universalis*, 2.
86. Hifi M (2004) Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. *Journal of Combinatorial Optimization*, 8, 65-84.
87. Hifi M, M'Hallah R (2002) A best-local position procedure-based heuristic for two-dimensional layout problems. *Studia Informatica Universalis*, 2, 33-56.

88. Hifi M, M'Hallah R (2005) An exact algorithm for constrained two-dimensional two-staged cutting problems. *Operations Research*, 53, 140-150.
89. Hifi M, M'Hallah R (2006) Strip generation algorithms for constrained two-dimensional two-staged cutting problems. *European Journal of Operational Research*, 172, 515-527.
90. Hifi M, M'Hallah R (2007) A dynamic adaptive local search algorithm for the circular packing problem. *European Journal of Operational Research*, 183, 1280-1294.
91. Hifi M, M'Halla S, Sadfi (2005) An exact algorithm for the knapsack sharing problem. *Computers & Operations Research*, 32, 1311-1324.
92. Hifi M, Ouafi R (1997) Best-first search and dynamic programming methods for cutting problems: The cases of one or more stock plates. *Computers & Industrial Enginering*, 32, 187-205.
93. Hifi M, Paschos VT, Zissimopoulos V (2004) A simulated annealing approach for the circular cutting problem. *European Journal of Operational Research*, 159, 430-448.
94. Hifi M, Roucairol C (2001) Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems. *Journal of Combinatorial Optimization*, 5, 465-494.
95. Hifi M, Zissimopoulos V (1996) A recursive exact algorithm for weighted two-dimensional cutting. *European Journal of Operational Research*, 91, 553-564.
96. Hifi M, Zissimopoulos V (1997) Constrained two-dimensional cutting: An improvement of Christofides and Withlock's exact algorithm. *Journal of the Operational Research Society*, 48, 324-331.
97. Hinxman A (1980) The trim-loss and assortment problems. A survey. *European Journal of Operational Research*, 5, 8-18.

98. Holthaus O (2002) Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths. *European Journal of Operational Research*, 141, 295-312.

99. Huang W, Chen D, Xu R (2007) A new heuristic algorithm for rectangle packing. *Computers & Operations Research*, 34, 3270-3280.

100. Kantorovich LM (1960) Mathematical methods of organizing and planning production. 1939 (Russian). English translation appeared in *Management Science*, 6, 362-422.

101. Karmakar N, Karp RM (1982) An efficient approximation scheme for the one-dimensional bin packing problem. In: Proceedings of the 23nd IEEE Symposium on Foundation of Computer Science (FOCS), 312-320.

102. Kellerer H, Pferschy U, Pisinger D (2004) Knapsack Problems, Springer, New York.

103. Kenyon C, Rémilla E (2000) A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25, 645-656.

104. Lee CC, Lee DT (1985) A simple on-line bin-packing algorithm. *Journal of the Association for Computing Machinery*, 32, 562-572.

105. León C, Miranda G, Rodríguez C, Segura C (2007) 2D Cutting stock problem: a new parallel algorithm and bounds. In: Proceedings of the 13th International Euro-Par Conference. European Conference on Parallel and Distributed Computing, IRISA, Rennes, France, August 28-31, 2007. Springer Verlag.

106. Limeira MS (1998) Development of an exact algorithm for solving a cutting pattern sequencing problem. Master, Brazilian Space Research Insitute, São José dos Campos, Brazil.

107. Lin E (1998) A bibliography survey on some weel-known non-standard knapsack problems. *INFOR*: *Information Systems and Operational Research*, 36, 274-317.

108. Lin S (1965) Computer solutions to the travelling-salesman problem. *Bell System Technical Journal*, 44, 2245-2270.

109. Linhares A, Yanasse HH (2002) Connections between cutting pattern sequencing, VLSI design and flexible machines. *Computers & Operations Research*, 29, 1759-1772.

110. Linhares A, Yanasse HH, Torreão JRA (1999) Linear gate assignment: A fast statistical mechanism approach. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 18, 1750-1758.

111. Lins R, Lins L, Morabito R (2002) An *n*-tet graph approach for non-guillotine packings of *n*-dimensional boxes into an *n*-container. *European Journal of Operational Research*, 141, 421-439.

112. Lins S (1989) Traversing trees and scheduling tasks for duplex corrugator machines. *Pesquisa Operacional*, 9, 40-54.

113. Lirov Y (Ed.) (1992) Cutting stock: Geometric resource allocation. *Mathematical and Computer Modelling*, 16.

114. Little JDC, Murty KG, Sweeney DW, Karel C (1963) An algorithm for the travelling salesman problem. *Operations Research*, 12, 972-989.

115. Lodi A (1999) Algorithms for two-dimensional packing and assignment problems. Ph.D. Thesis, Università di Bologna.

116. Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141, 241-252.

117. Lodi A, Martello S, Vigo D (2002a) Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123, 379-396.

118. Lodi A, Martello S, Vigo D (2002b) Heuristic algorithms for the tree-dimensional bin packing problem. *European Journal of Operational Research*, 141, 410-420.

119. Lodi A, Monaci M (2003) Integer linear programming model for 2-staged two-dimensional knapsack problems. *Mathematical Programming Ser. B*, 94, 257-278.

120. Madsen OBG (1979) Glass cutting in a small firm. *Mathematical Programming*, 17, 85-90.

121. Madsen OBG (1980) A cutting sequencing algorithm. In: Iracki K, Malanowsky K, Walukiewitz S (Eds.) Optimization Methods, Springer Verlag, Hamburg, 388-396.

122. Madsen OBG (1988) An application of travelling salesman routines to solve pattern allocation problems in the glass industry. *Journal of the Operational Research Society*, 39, 249-256.

123. Martello S (Ed.) (1994a) Knapsack, packing and cutting, Part I: One-dimensional knapsack problems. *INFOR*: *Information Systems and Operational Research*, 32.

124. Martello S (Ed.) (1994b) Knapsack, packing and cutting, Part II: Multi-dimensional knapsack and cutting stock problems. *INFOR*: *Information Systems and Operational Research*, 32.

125. Martello S, Monaci M, Vigo D (2003) An exact approach to the strip packing problem. *INFORMS Journal on Computing*, 15, 310-319.

126. Martello S, Pisinger D, Vigo D (2000) The three-dimensional bin packing problem. *Operatinos Research*, 48, 256-267.

127. Martello S, Toth P (1990) Knapsack Problems – Algorithms and Computer Implementations, John Wiley & Sons, Chichester.

128. Martello S, Vigo D (1998) Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44, 388-399.

129. McCormick WT Jr, Schweitzer PJ, White TW (1972) Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20, 993-1009.

130. McDiarmind C (1999) Pattern minimisation in cutting stock problems. *Discrete Applied Mathematics*, 98, 121-130.

131. Möhring RH (1990) Graph problems related to gate matrix layout and PLA folding. *Computing*, 7, 17-51.

132. Morabito R, Arenales MN (1996) Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. *European Journal of Operational Research*, 94, 548-560.

133. Morabito R, Arenales MN, Arcaro VF (1992) An AND/OR-graph approach for two-dimensional cutting problems. *European Journal of Operational Research*, 58, 263-271.

134. Morabito R, Belluzzo L (2007) Optimising the cutting of wood fibre plates in the hardboard industry. *European Journal of Operational Research*, 183, 1405-1420.

135. Morabito R, Garcia L (1998) The cutting stock problem in a hardboard industry: A case study. *Computers & Operations Research*, 25, 469-485.

136. Morse PM (1972) Optimal linear ordering of information items. *Operations Research*, 20, 741-751.

137. Mukhacheva EA (Ed.) (1997) Decision making under conditions of uncertainty: Cutting-packing problems. *The International Scientific Collection*, Ufa, Russia.

138. Mukhacheva EA, Zalgaller VA (1993) Linear programming for cutting problems. *International Journal of Software Engineering and Knowledge Engineering*, 3, 463-477.

139. Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization, Wiley, New York.

140. Nitsche C, Scheithauer G, Terno J (1998) New cases of the cutting stock problem having MIRUP. *Mathematical Methods of Operations research*, 48, 105-115.

141. Nitsche C, Scheithauer G, Terno J (1999) Tighter relaxations for the cutting stock problem. *European Journal of Operational Research*, 112, 654-663.

142. Nordström AL, Tufekci S (1994) A genetic algorithm for the talent scheduling problem. *Computers & Operations Research*, 21, 927-940.

143. Oliveira JF, Ferreira JS (1990) An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, 44, 256-266.

144. Oliveira ACM, Lorena LAN (2002) A constructive genetic algorithm for gate matrix layout problems. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 21, 969-974.

145. Oliveira JF, Wäscher G (Eds.) (2007) Cutting and packing. *European Journal of Operational Research*, 183, 1106-1108.

146. Padberg M (2000) Packing small boxes into a big box. *Mathematical Methods of Operations Research*, 52, 1-21.

147. Pan P, Shi W, Liu CL (1996) Area minimization for hierarchical floorplans. *Algorithmica*, 15, 550-571.

148. Perboli G, Crainic TG, Tadei R (2006) UniPack: A new heuristic framework for multi-dimensional packing problems. In: The 37th AIRO Annual Conference (Conference of the Italian Operations Research Society), 127, Cesena (FC), Italy, September 12-15.

149. Pezzella F, De Giovanni L (2007) A genetic approach for the cutting pattern sequencing problem. In: The 22nd EURO Conference (European Conference on Operational Research), 67, Prague, Czech Republic, July, 8-11.

150. Pileggi GCF (2002) Approaches for the integrated optimization of patterns generation and sequencing problems. Ph.D. Thesis, University of São Paulo, São Carlos, Brazil.

151. Pileggi GCF, Morabito R, Arenales MN (2006) Integrated optimization of the generation and sequencing of cutting patterns in the cutting stock problem. Technical Report.

152. Pisinger D (2000) A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, 12, 75-84.

153. Pisinger D, Toth P (1998) Knapsack problems. In: Du DZ, Pardalos P (Eds.) Handbook of Combinatorial Optimization, Kluwer, 299-428.

154. Poldi KC, Arenales MN (2005) Dealing with small demand in integer cutting stock problems with limited different stock lengths. Technical Report ICMC-85, Universitade de São Paulo, São Carlos, Brazil.

155. Puchinger J (2005) Combining metaheuristics and integer programming for solving cutting and packing problems. Ph.D. Thesis, University of Technology, Vienna.

156. Puchinger J, Raidl GR, Koller G (2004) Solving a real-world glass cutting problem. In: Gottlieb J, Raidl GR (Eds.) Proceedings of the 4th International Conference on Combinatorial Optimization (EvoCOP 2004, Coimbra, Portugal), volume 3004 of Springer-Verlag, Lecture Notes in Computer Science, 162-173.

157. Respicio A, Captivo EM (2005) Bi-objective sequencing and cutting patterns - An application for the paper industry. In: Ibaraki T, Nonobe K, Yagiura M (Eds.) Metaheuristics: Progress as Real Problem Solvers, Springer US, 227-241.

158. Richey MB (1991) Improved bounds for harmonic-based bin packing algorithms. *Information Processing Letters*, 34, 203-227.

159. Riehme J, Scheithauer G, Terno J (1996) The solution of two-stage guillotine cutting stock problems having extremely varying order demands. *European Journal of Operational Research*, 91, 543-552.

160. Rietz J, Scheithauer G (2002) Tighter bounds for the gap and non-IRUP constructions in the one-dimensional cutting stock problem. *Optimization*, 51, 927-963.

161. Rinaldi F, Franz A (2007) A two-dimensional strip cutting problem. *European Journal of Operational Research*, 183, 1371-1384.

162. Roodman GM (1986) Near-optimal solutions to one-dimensional cutting stock problems. *Computers & Operations Research*, 13, 713-719.

163. Salzer, HE (1947) The approximation of numbers as sums of reciprocals. *The American Mathematical Monthly*, 54, 135-142.

164. Scheithauer G (1993) Computation of optimal phi-simple guillotine cutting patterns. *Journal of Information Processing Cybernetics***,** 29, 115-128.

165. Scheithauer G (1995) The solution of packing problems with pieces of variable length and additional allocation constraints. *Optimization*, 34, 81-96.

166. Scheithauer G, Sommerweiβ U (1998) 4-Block heuristic for the rectangle packing problem. *European Journal of Operational Research*, 108, 509-526.

167. Scheithauer G, Terno J (1996) The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, 84, 562-571.

168. Scheithauer G, Terno J (1997) Theoretical investigations on the modified integer round-up property for the one-dimensional cutting stock problem. *Operations Research Letters*, 20, 93-100.

169. Scheithauer G, Terno J, Müller A, Belov G (2001) Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm. *Journal of the Operational Research Society*, 52, 1390-1401.

170. Seiden SS, Woeginger GJ (2005) The two-dimensional cutting stock problem revisited. *Mathematical Programming Ser. A*, 102, 519-530.

171. Soma NY, Toth P (2002) An exact algorithm for the subset sum problem. *European Journal of Operational Research*, 136, 57-66.

172. Stadtler (1990) A one-dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research*, 44, 209-223.

173. Stoyan Y, Scheithauer G, Gil N, Romanova T (2004) Φ-functions for primary 2D-objects. *4OR: A Quarterly Journal of Operations Research*, 2, 69-84.

174. Suliman SMA (2001) Pattern generation procedure for the cutting stock problem. *International Journal of Production Economics*, 74, 293-301.

175. Sweeney PE, Paternoster E (1992) Cutting and packing problems. *Journal of the Operational Research Society*, 43, 691-706.

176. Tang CS, Dendardo EV (1988) Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. *Operations Research*, 36, 767-777.

177. Tschöke A, Holthöfer N (1995) A new parallel approach to the constrained two-dimensional cutting stock problem. In: Ferreira A, Rolim J (Eds.) Parallel Algorithms for Irregularly Structured Problems, Springer Verlag, Berlin, Germany, 285-300.

178. Valério de Carvalho JM (1998) Exact solutions of cutting stock problems using column generation and branch-and-bound. *International Transactions in Operational Research*, 5, 35-44.

179. Valério de Carvalho JM (2002) LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141, 253-273.

180. Valério de Carvalho JM, Guimarães Rodriguez AJ (1995) An LP-based approach to a two-stage cutting stock problem. *European Journal of Operational Research*, 84, 580-589.

181. Vance P, Barnhart C, Johnson E, Nemhauser G (1994) Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3, 111-130.

182. Vanderbeck F (1999) Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming Ser. A*, 86, 565-594.

183. Vanderbeck F (2000) Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem. *Operations Research*, 48, 915-926.

184. Vasko FJ (1989) A computational improvement to Wang's two-dimensional cutting stock algorithm. *Computers & Industrial Engineering*, 16, 109-115.

185. Vasko FJ, Wolf FE, Stott KL (1989) A practical solution to a fuzzy two-dimensional cutting stock problem. *Fuzzy Sets and Systems*, 29, 259-275.

186. Viswanathan KV, Bagchi A (1993) Best-first search methods for constrained two-dimensional cutting stock problems. *Operations Research*, 41, 768-776.

187. Wang H, Huang W, Zhang Q, Xu D (2002) An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141, 440-453.

188. Wang PY (1983) Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, 31, 573-586.

189. Wang PY, Valenzuela CL (2001a) Data set generation for rectangular placement problems. *European Journal of Operational Research*, 134, 378-391.

190. Wang PY, Valenzuela CL (2001b) Data set generation for non-slicing rectangular placement problems. Technical Report (http://citeseer.ist.psu.edu/616245.html).

191. Wang PY, Wäscher G (Eds.) (2002) Cutting and packing. *European Journal of Operational Research* 141, 239-240.

192. Wäscher G, Gau T (1996) Heuristics for the one-dimensional cutting stock problem: A computational study. *OR Specktrum*, 18, 131-144.

193. Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. *European Journal of Operational research*, 183, 1109-1130.

194. Whitwell (2004) Novel heuristic and metaheuristic approaches to cutting and packing. Ph.D. Thesis, University of Nottingham.

195. Wolsey LA (1998) Integer Programming, John Wiley & Sons, New York.

196. Yanasse HH (1996) Minimization of open orders - Polynomial algorithms for some special cases. *Pesquisa Operacional*, 16, 1-26.

197. Yanasse HH (1997a) An exact algorithm for the tree case of minimization of open orders problem. Technical Report LAC-001/97, INPE, São José dos Campos, Brazil.

198. Yanasse HH (1997b) On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational research*, 100, 454-463.

199. Yanasse HH (1997c) A transformation for solving a pattern sequencing problem in the wood cut industry. *Pesquisa Operacional*, 17, 57-70.

200. Yanasse HH (1998) A note on generating solutions of a pattern sequencing problem to minimize the maximum number of open orders. Technical Report LAC-002/98, INPE, São José dos Campos, Brazil.

201. Yanasse HH, Becceneri JC, Soma NY (1998) An exact algorithm for a special case of a pattern sequencing problem. In: The 30th XXX SBPO, Curitiba, PR, Brazil.

202. Yanasse HH, Becceneri JC, Soma NY (1999) Bounds for a problem of sequencing patterns. *Pesquisa Operacional*, 19, 249-277.

203. Yanasse HH, Limeira MS (2004) Refinements on an enumeration scheme for solving a pattern sequencing problem. *International Transactions in Operations Research*, 11, 277-292.

204. Yanasse HH, Limeira MS (2006) A hybrid heuristic to reduce the number of different patterns in cutting stock problems. *Computers & Operations Research*, 33, 2744-2756.

205. Yanasse HH, Pinto Lamosa MJ (2007) An integrated cutting stock and sequencing problem. *European Journal of Operational Research*, 183, 1353-1370.

206. Yanasse HH, Zinober ASI, Harris RG (1991) Two-dimensional cutting stock with multiple stock size. *Journal of the Operational Research Society*, 42, 673-683.

207. Yuen BY (1991) Heuristics for sequencing cutting patterns. *European Journal of Operational Research,* 55, 183-190.

208. Yuen BY (1995) Improved heuristics for sequencing cutting patterns. *European Journal of Operational Research,* 87, 57-64.

209. Yuen BY, Richardson KV (1995) Establishing the optimality of sequencing heuristic for cutting stock problems. *European Journal of Operational Research,* 84, 590-598.

210. Zak E (2002a) Modeling multistage cutting stock problems. *European Journal of Operational Research,* 141, 313-327.

211. Zak E (2002b) Row and column generation technique for a multistage cutting stock problem. *Computers & Operations Research,* 29, 1143-1156.