to Giulia and dad's mind

Università degli Studi di Napoli "Federico II"

Research Doctorate

in Computational Biology and Bioinformatics

Doctoral Thesys

Biclustering of gene expression data: hybridization of GRASP with other heuristic/metaheuristic approaches

Tutor

Prof.ssa Paola Festa

Co-Tutor

Dott.ssa Anna Marabotti

Candidate

Dott. Francesco Musacchia

Academic Year 2011-2012

Abstract

Researchers who work on large amount of data have to face various problems such as data mining and information retrieval: this is the case of gene expression. The general scope of these experiments is to find co-regulated genes, in order to understand the biologic pathways underlying a particular phenomenon. A clustering concept can be used to find out if co-regulated genes can be active only over some conditions. Recently, some biclustering approaches have been used to find groups of co-regulated genes into a data matrix. Among them, several heuristic algorithms have been developed to find good solutions in a reasonable running time.

In the current Ph.D. thesis, a GRASP-like (Greedy Randomized Adaptive Search Procedure) approach was developed to perform biclustering of microarray data. A new local search has been developed composed of three simple steps based on a concept inspired by the social aggregation of groups. It is very fast and allows to obtain results similar to those achieved using some of the best known biclustering algorithms. Other new algorithms have also been proposed using novel combinations of iterated local search and MST clustering.

The different biclustering algorithms were then tested on four different datasets of gene expression data. Results are encouraging because they are similar or even better to those obtained with the

former GRASP-like algorithm. Possible future improvements could be obtained by implementing further combinations of heuristics and testing them onto different datasets in order to evaluate their general application to different kinds of data.

Index

List of Figures

List of Tables

List of abbreviations

$\mathcal{A}$ - A matrix that stores a dataset.
$\mathcal{B}$ - A generic bicluster.
$\mathcal{B}(I, J)$ - A generic bicluster. I and J are the subset of rows and columns respectively.
BP - GO abbreviation for: biological process.
CC - Cheng and Church algorithm for biclustering [10].
CC - GO abbreviation for: cellular components.
DNGr - GRASP-like algorithm from Dahran and Nair [51].
E - A set of edges.
$\mathcal{F}$ - A function used in the iterated local search to evaluate biclusters using the $\mathcal{S}$ value.
$g(s)$ - The cost function to evaluate the solution s.
$G = (V, E)$ - A graph, composed by a set of vertices V and a set of edges E.
H - Formula of the MSR score by Cheng and Church [10].
ILS - Iterated local search.
ISA - Iterative signature algorithm for biclustering [26].
ILSAy - Iterated local search by Ayadi [56].
kM - kMeans clustering.
MF - GO abbreviation for: molecular function.
MSR - The minimum residual score by Cheng and Church [10].
MST - Minimum spanning tree clustering.

MyGR - GRASP-like algorithm developed in this work.
$\mathcal{O}$ - It is used for the computation complexity of the algorithms.
$p(n)$ - A generic function with input n.
$R(x)$ - The residual. It is part of the MSR score of Cheng and Church [10].
RCL - The restricted candidate list.
s - A generic solution.
S - A generic set of solutions.
$\mathcal{S}$ - A function used in the iterated local search to evaluate bicluster.
Yeast TS - abbreviation for Yeast time series dataset.
V - A set of vertices.
$\rho(x, y)$ - A generic distance between two vectors x and y.
Δ - The set of costs associated with an insertion in GRASP algorithm.

Chapter 1

Omics data analysis

The fish you find depends upon the sea you sail upon.

1.1 The origins

The human genome is composed of approximately 20.000 genes [1]. A very long sequence of DNA encodes them. What gives the difference in terms of shape and functions to the cells of our organisms is the activation of genes. For this reason, researchers have been investigating for long time the best way to analyze the DNA and discover what are the genes activated in each kind of cell or in different cell conditions.

The first studies on gene expression data appeared in 1977, when the Northern blot technique was developed [2].

In the 80s, Roger Ekins (Department of Molecular Endocrinology, University College London Medical School) produced and patented ligand-binding assays in a microarray format [3]. The secret of this technique is that, when the spot is small enough, the signal is clear and independent from the chemical compounds used to bind [4].

Since the 90s till today, a growing number of commercial entities and academic groups has contributed to the advancement in microarray technology.

With the evolution of "omics"-based technologies, importance of algorithms to store and manage data increased with the amount of data generated.

1.2 Microarrays

A microarray is a slide (chip) on which short DNA molecules called "probes", which correspond unambiguously to a gene, are fixed. This DNA may be of genomic origin, e.g. in case it originates from prokaryotic organisms, or can be obtained by extraction of the RNA molecules from the cells of the organism of interest, and reverse transcription into cDNA using an enzyme called *reverse transcriptase*, in the case it derives from eukaryotes. Finally it can be obtained through synthesis. In the most recent microarrays, the probes are deposited on the slide by various techniques, including photolithography [5]. In this technique, as schematized in Figure 1.1, the surface of a substrate (a silicon wafer), rich of hydroxyalkyl groups, is covered by photosensitive molecules that are able to mask the reactive groups. The synthesis of the oligonucleotides is carried out directly on the surface of the slide, through a series of cycles of deprotection and coupling. A photolithographic mask allows the transmission of light radiation only in some selected points of the wafer. Therefore, when the photolithographic mask is aligned with the wafer, the light allows the deprotection, and the subsequent activation of selected reactive groups. By passing solutions of nucleotides, they will couple to the deprotected reactive groups. The multiple steps of deprotection and coupling allow the synthesis of the entire set of the desired product.

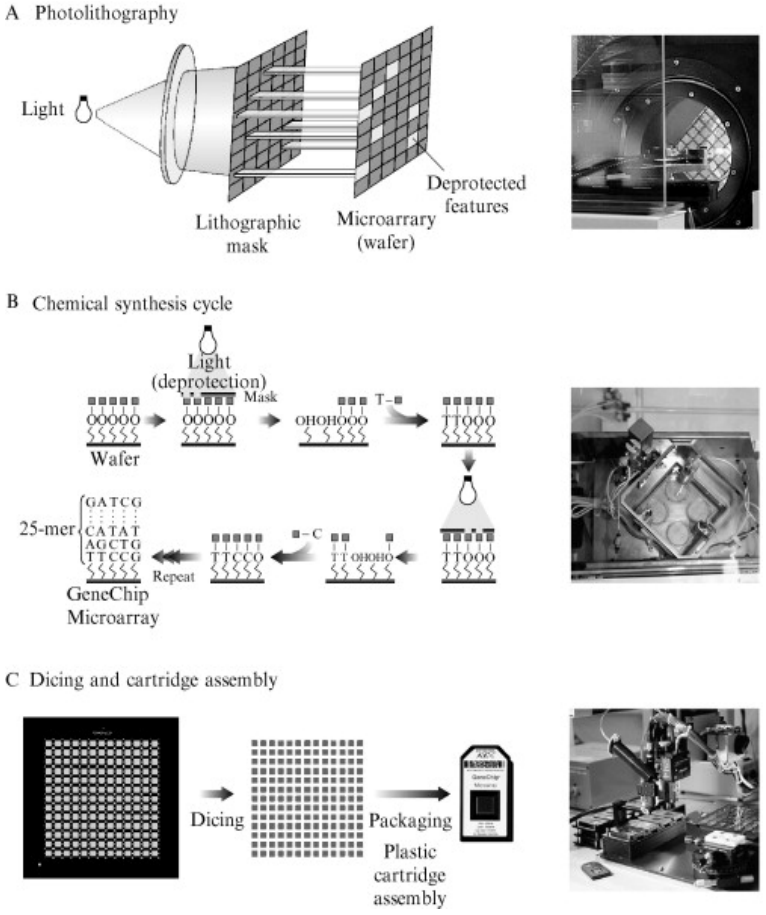

Figure 1.1: Manufacture of a gene chip. (A) On the left, an ultraviolet light pass through a mask containing open windows. The position of each open window identifies the surface on the wafer which will be activated for the chemical synthesis. On the right, a picture of the photolithographic process; (B) On the left is represented the cycle through which the nucleic acids are synthesized: the light removes the protections (square) in the areas of the array. A single nucleotide is coupled in the area without protection. Through successive steps, any oligonucleotide sequence can be mounted on each feature of the array. On the right, an image of the chemical synthesis station where nucleotides binds; (C) On the left the complete synthesis in the wafer results in many microarrays in a single wafer that will be placed in plastic cartridges. On the right, the machine that takes care of incorporating the microarrays in the cartridge. Image taken from [5].

An array of 1.28 cm^2 may contain about 1.4 million locations for the probe, and each of them can contain millions of identical

molecules of DNA. For example, an Affymetrix microarray can contain a total of about 6.5 million probes on the same array. Microarrays containing entire genomes of many animal species, including humans are commercially available, but researchers may also ask for the creation of custom microarrays for particular research interests.

Since microarrays allow the study of large amounts of genes at a time and are very fast, they have been developed as an important tool to monitor the gene expression leading to the development of a new "omics" technique called *Transcriptomic*.

A possible application is the analysis of a pathological tissue against the same tissue in the healthy state (Figure 1.2).

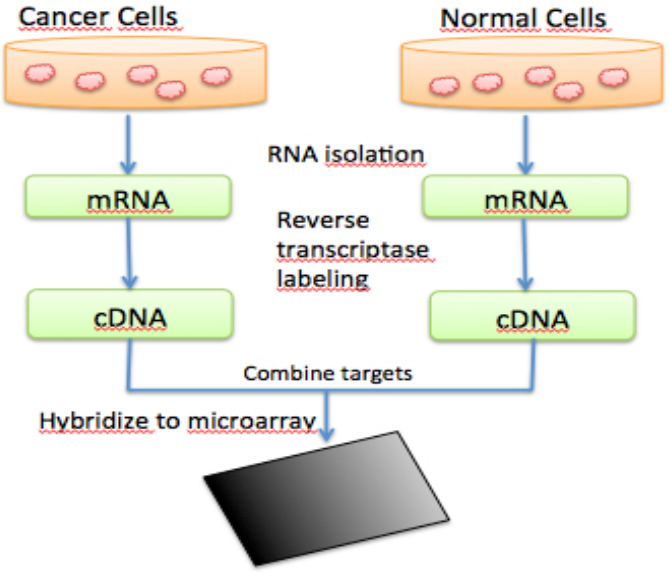

Figure 1.2: Workflow of a possible application of microarray. In this example a cancer cell and an healthy one are taken, mRNA is extracted and cDNA created. Then the cDNA is hybridized on the microarray.

1.2.1 From mRNA to images

A microarray experiment starts by taking samples of tissues or cells in different conditions.

The mRNA molecules are extracted from the cells and are retro-transcribed into cDNA (complementary DNA) using an enzyme called reverse transcriptase. This step is necessary because the mRNA molecules are easily degraded, while the DNA is more stable. The cDNAs are then labeled in order to distinguish the different conditions analysed, using fluorescent probes which, when energized, emit light at different wavelengths, corresponding to different colors (the most common being red and green). Then, the cDNA from the two samples are mixed and put on the slide, where they bind to the spot where their complementary sequence is present, with a process called hybridization. After hybridization and subsequent washing to remove samples that did not react, the chip is excited by a laser at an appropriate wavelength to allow the excitation of fluorescent probes and the emission of the color of the corresponding fluorescence, indicating that the cDNA hybridization to a probe of the chip is done (Figure 1.3).

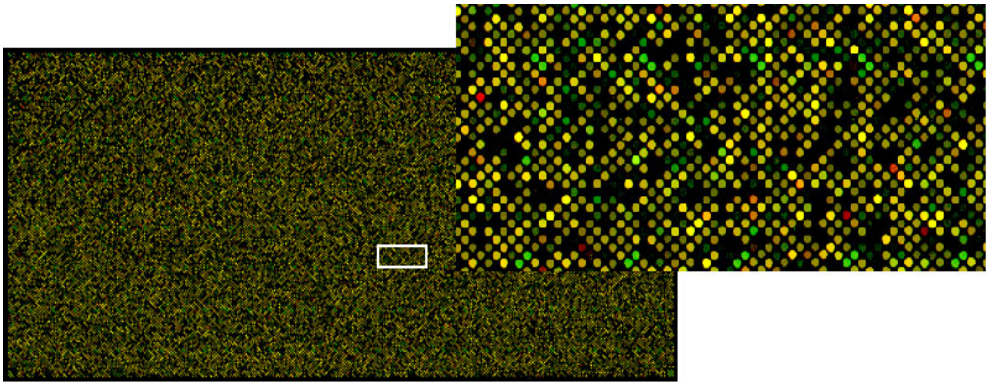

Figure 1.3: An example of microarray. Green, red, yellow, black represent the fact that genes in the samples are expressed or not. (Image of Agilent microarray)

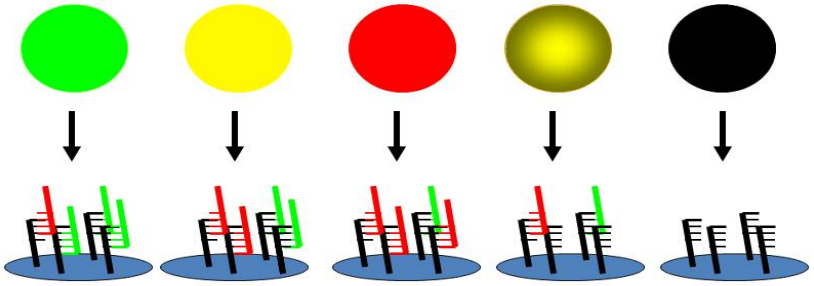

Figure 1.4: Formation of colors on a microarray spot. The colors are: red or green for exclusive expression of genes in one tissue, yellow for expression of gene in both tissues, dark yellow for a low expression and black for no expression.

Using red and green dyes, the possible emitted colors are, as in Figure 1.4:

- yellow, when the gene is expressed in both tissues in a comparable quantity;
- red, when only the gene of the tissue in the first condition is expressed;
- green, when only the gene of the tissue in the second condition is expressed;
- dark yellow, when the genes are both expressed in low quantity;
- black, when gene is not expressed in both cases. In this case, in fact, no cDNA is bound to the probe.

1.2.2 From images to data

The ratio of the emission intensity relative to each fluorophore is used to identify the genes that are over-expressed or under-expressed in one of two samples. This is performed by a segmentation algorithm that, in general, is based on the color and edge detection. It can be affected by errors in scanning or by the presence of poor-quality and low-intensity features, thus the result has to be filtered. Then the background is subtracted to keep in evidence only the spots and the detection of their intensity is performed. This is a crucial step for the following analysis.

Once the image is produced, a set of statistics are computed: each pixel is taken into account and mean, median and intensity values are reported. The signal intensity is considered significant if it is at least two standard deviation above the background [6]. Finally, a table of fluorescent intensities for each gene in the array is exported in some *.dat* files (RAW images that represent the microscope rough signal). Since the amounts of mRNA for each sample can be different, these values in raw form have to be normalized.

The data generated by the microarrays has been subject of several discussion on how to standardize: common ways to name the genes have been decided. A standardization occurs also for the chip technology, for the biological processes involved in its realization and, finally, for the display of results.

Another big problem is data storage. An huge amount of data needs great memory space. NCBI (National Center of Biotechnology Information) spawned a system called the *GEO* (Gene Expression Omnibus) which contains microarray results from laboratories all around the world stored in a standard format widely recognized by the scientific community. Many types of gene expression data are accepted and archived as public dataset and this permits to use them also to researcher that cannot produce their own microarrays [7].

1.2.3 From data to biological significance

The final result of a microarray experiment is a large table of real or integer numbers whose values represent the gene expression. Sometimes values are discretized between 0 and 1 (where 1 means that gene is expressed above a certain threshold and 0 below).

A gene can be expressed as a row vector $g_i = \{a_{i1}, \ldots, a_{in}\}$ and datasets of gene expression data are usually represented in form of a matrix $\mathcal{A}$ where each value $a_{i,j}$ corresponds to the expression value of the gene i in the j condition. The following is an example:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

where m and n correspond to the cardinality of, respectively, gene and conditions sets.

To derive a biological significance from these data, various analyses can be made.

The ultimate goal of a microarray experiment is to identify groups of co-regulated genes, because this allows to assign possible functions to genes not previously characterized, or identify common regulatory motifs and cascades of regulation deriving the reasons for the physiological reactions to a stimulus.

There are several approaches that allow to extract this kind of information. Among these, the most widespread and popular are the clustering algorithms, which allow to group those genes that are expressed in a similar manner under certain conditions without making *a priori* assumptions about the possible categories to be assigned to the data.

1.3 Clustering

Different types of clustering algorithms have been studied and implemented in the past and recent years. They belong to the class of *pattern recognition* algorithms, where the *pattern* is a pair {*observation*, *meaning*} and the term *recognition* is the act of giving a meaning to the observation. The goal of pattern recognition is to establish a mapping from the space of the observation to the one of meanings. The steps are: data pre-processing, feature extraction, classification. Each step can be optimized in its computational complexity to obtain more powerful algorithms and database structures [8].

Pattern recognition is a subclass of the broader *machine learning* class of algorithms and it is subdivided in two different types of learning: supervised and unsupervised. The first approach allows to recognize the class of an observation on the basis of a priori data. Some frameworks like *neural networks*, *self organized maps* (SOM), *support vector machines* (SVM) belong to this first class where the system can have in input a large set of pairs {*observation*, *meaning*} and they can "learn" from a *training set* and later test by using a *test set* (used also to assess the robustness of the network). Instead unsupervised techniques start without any *a priori* information and try to classify objects using only the information contained in the observations given as input [9].

A gene subjected to a number of conditions can be represented as a vector whose values (data conditions) are its characteristics (features). The characteristics of two different vectors can be compared to express the similarity of features according to their spatial proximity.

As stated, in the case of genes, features will be given by their expression levels $g_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$. Using measures of similarity (*cosine distance*, *euclidean distance*, *Pearson coefficient*, *Manhattan*,

etc) one can write an algorithm that will compute distances between the vectors and put them in the correct class (Figure 1.5).

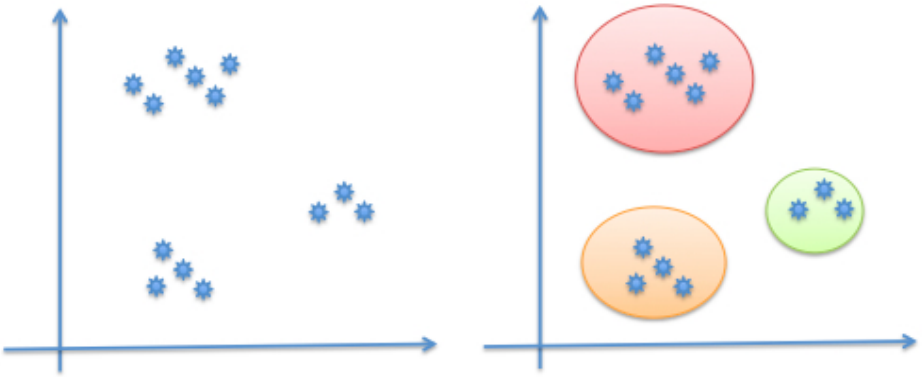

Figure 1.5: An example of clustering. On the left, the dataset represented by feature vectors that are point in the n-dimensional space. On the right, vectors have been grouped in clusters of elements.

Many techniques exist to address the problem of clustering. For the clustering of microarray data one of the most widely used strategy is the **hierarchical clustering** that builds clusters gradually, while they are growing up. It can be agglomerative, when it starts from each separated vector and put them in the most similar cluster, or divisive, when, starting from big clusters, it splits them in smaller ones with similar properties. Partitioning methods (like kMeans) create clusters shifting elements from a cluster to another one during their execution [9].

1.3.1 Clustering *vs* biclustering

Clustering algorihtms can be very useful for the purpose of gene expression analysis and classification. However, with matrices of gene expression data, sometimes one does not want to group the genes on the basis of the overall conditions. A way to solve this problem

is to disclose the involvement of a gene in more than one cluster and saving the most significant one. This cannot be done with clustering, where genes are collected based on the overall similarities, but it can be done using a technique called *Biclustering* introduced by Cheng and Church in 2000 [10].

Biclustering of gene expression data permits also to discover what are the similarities between genes subjected to a subset of conditions. This is intuitively good when one thinks that a cellular process is active only in some conditions or that a gene may participate in pathways that may or may not be active together under all conditions.

Different types of biclusters have been described with mathematical rules. An overview on these rules will be given in the next chapter. The ones relevant here are those in which a gene/condition belongs to more than one cluster and is grouped using only a subset of genes/conditions.

Chapter 2

Biclustering and its evaluation

Do not make a decision when you do not have to take one.

2.1 Problem formulation

The term *Biclustering* was first introduced by Mirkin [11] but a similar idea was already present in a work of Hartigan in 1972, where he was working on the possibility to explore a table of electoral votes [12]. It was used then by Cheng and Church for the analysis of gene expression data [10].

Biclustering is the process of identifying n sub-matrices $\mathcal{B} = \mathcal{B}_1, \ldots, \mathcal{B}_n$ from a matrix of values $\mathcal{A}$ so that elements of $\mathcal{B}_i$ follow a desired behavior. Each $\mathcal{B}_i$ may share columns or rows of the matrix $\mathcal{A}$ with other sub-matrices. The order is not important: each row (column) of a bicluster $\mathcal{B}_i$ can be followed by any subsequent or previous row (column) [10]. It is shown in Figure 2.1 that the order is not important and elements can belong to different subsets.

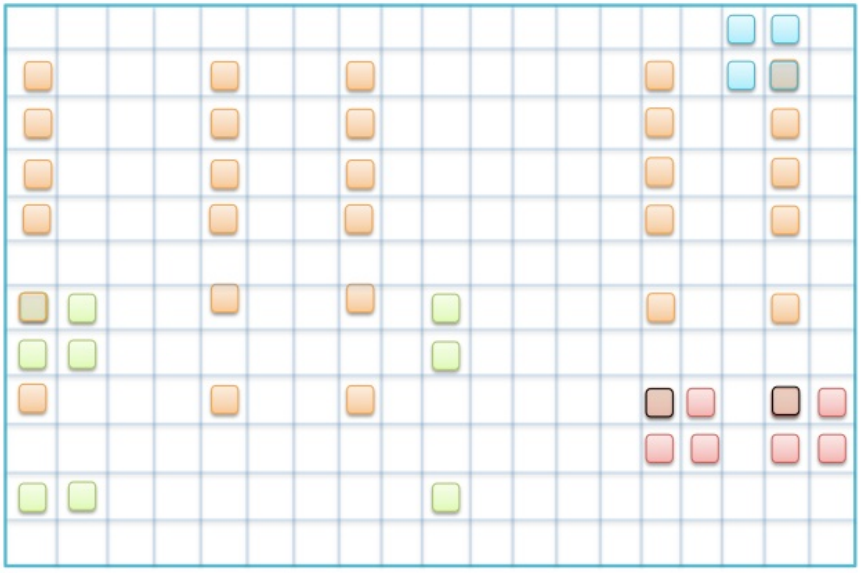

Figure 2.1: An example of biclustering output. In the image are present four different biclusters distinguished by four different colors. Biclusters can be overlapped and, in this case, some elements from a bicluster can belong also to another one. This is evidenced in the image as a mixed color, i.e. the elements of the bicluster with darker edges.

Biclustering is applied also to other problems and not only to gene expression data. For example it has been applied for electoral vote and truth tables [13] and for the detection of important terms in the advertising [14].

A generic mathematical formalization of the problem can be written: given a matrix $\mathcal{A} = (X, Y)$, where $X = \{x_1, \ldots, x_m\}$ is the set of rows, $Y = \{y_1, \ldots, y_n\}$ the set of columns and a_{ij} represents the value of object i subject to the condition j (the expression value of gene i at the j-th condition), then one can define a bicluster as a subset $\mathcal{B} = (I, J)$ where $I = \{i_1, \ldots, i_k\} \subseteq X$ and $J = \{j_1, \ldots, j_s\} \subseteq Y$. Each bicluster has to respect some rule of homogeneity that can differ with the addressing problem.

Madeira and Oliveira [15] stated that the problem of bicluster-

ing is NP-complete because, representing the matrix as a complete weighted bipartite graph, one has to find the maximum edge biclique [16].

Different types of biclusters can be found and in the following paragraph a brief illustration of them is given.

2.1.1 Types of biclusters

According to Madeira and Oliveira [15] four types of biclusters can be defined:

1. **constant values** bicluster, when all the values are identical;
2. **constant rows** bicluster, when all the rows have the same values;
3. **constant columns**, when all the columns have the same values;
4. **coherent evolution** in additive or multiplicative way. These are the more interesting biclusters: they are composed of elements that are sequences of numbers that increase or decrease in a regular way.

All these types of biclusters are shown in Figure 2.2.

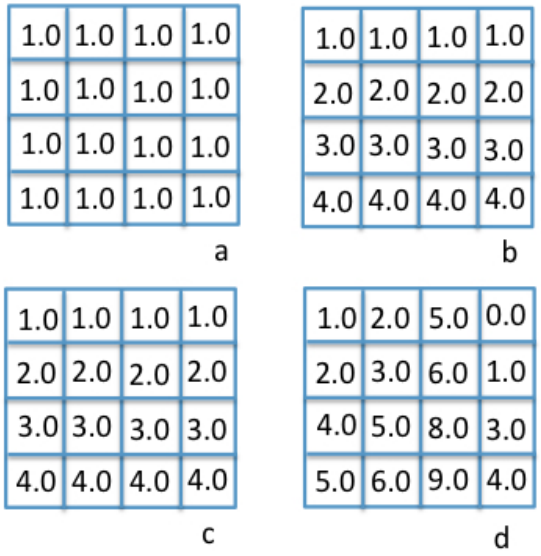

Figure 2.2: Bicluster types. a) bicluster with constant values; b) constant rows bicluster; c) constant columns; d) coherent evolution in additive or multiplicative way.

These types are obviously related to specific problems tha require specific algorithms to be solved. Usually they start from a bicluster seed and improve it, but also two-way clustering methods exist that search first for columns and rows separately and then combine these results to obtain a final result.

In some algorithms the search is restricted only to one representative bicluster that can have different localization inside the matrix $\mathcal{A}$. When more than one bicluster is present, the possible structures are the following [15]:

1. non-overlapping biclusters;
2. completely separated non-overlapping;
3. overlapping biclusters with hierarchical structure;
4. overlapping biclusters with arbitrary positions.

They are shown in Figure 2.3

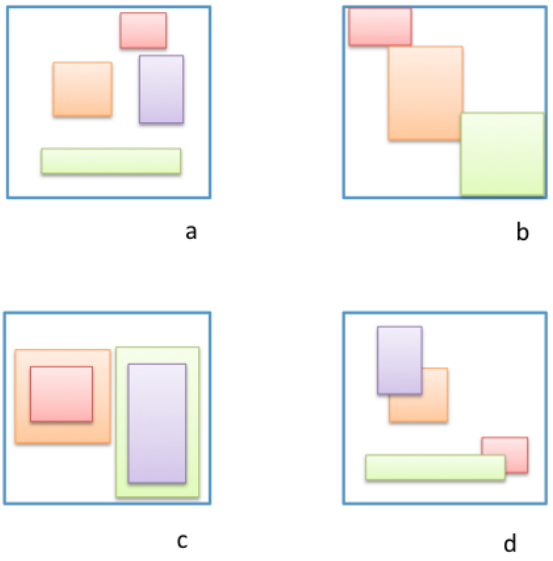

Figure 2.3: Bicluster structures. a) non overlapping biclusters; b) completely separated non-overlapping; c) overlapping biclusters with hierarchical structure; d) overlapping biclusters with arbitrary positions.

Here has been studied only the type of biclusters where values on row and columns increase or decrease in the same way (Figure 2.2, coherent evolution) and those that can have elements in common and be positioned everywhere (Figure 2.3, overlapping bicluster with arbitrary position) have been studied. Biclusters can also be non-exhaustive: there will be rows or columns that will not belong to any bicluster.

H score

Cheng and Church [10] defined a measure of evaluation for a generic bicluster $\mathcal{B}_{IJ}$ as the sum of the squared residues $R(a_{ij})|a_{ij} \in \mathcal{A}$. It is a measure of how well the element fits into the bicluster $\mathcal{B}_{IJ}$ and it is indicated with the letter H.

If the residue is written as:

$$R(a_{ij}) = a_{ij} - a_{Ij} - a_{iJ} + a_{IJ}$$

then H score is defined as:

$$H = \frac{\sum_{i \in I, j \in J} (R(a_{ij}))^2}{|I| \cdot |J|},$$

where

$$a_{Ij} = \frac{\sum_{i \in I} a_{ij}}{|I|},$$

is the column mean of column j,

$$a_{iJ} = \frac{\sum_{j \in J} a_{ij}}{|J|},$$

is the row mean of row i,

$$a_{IJ} = \frac{\sum_{i \in I, j \in J} a_{ij}}{|I| \cdot |J|},$$

is the bicluster mean.

The lower the H value, the higher the coherence of the bicluster. Computational complexity of the H score computation depends from the dimensions of the bicluster: $\mathcal{O}((|I| \cdot |J|)^2)$.

2.1.2 Solution to the problem of biclustering

A **candidate solution** is a member of a set of possible solutions for a given problem because it is in the set of solutions that satisfy all constraints. The space of all candidate solutions is called the **feasible region** or the **solutions space**.

In the world of biclustering, a **feasible solution** is a set of submatrices overlapped (or not) that best describes a "biclustering" of the genes represented. Given the set $\mathcal{B} = \{\mathcal{B}_1 = (I_1, J_1), \ldots, \mathcal{B}_k =$

$(I_k, J_k)\}$ of biclusters resulting from a run of the algorithm, one would to maximize the expression:

$$min \sum_{i=1}^{k} H(\mathcal{B}_i),$$

where i is the i-th bicluster and H is the score of the bicluster.

2.2 State of the art

2.2.1 State-of-the-art algorithms

Several techniques have been designed to address the problem of biclustering. They have been divided from Madeira and Oliveira in five distinct groups [15]. These are actually slightly different and for this reason the fundamental ones are explained. A first type of algorithms does not solve the computational intractability and maybe can work only with low dimensions matrices: it is the *exhaustive* one that try with possible combination of biclusters finding the better ones. The second is the *two-way clustering*, where a clustering is computed first for the rows and then for the columns and results are mixed together to obtain significant biclusters. A third type is that of the *greedy algorithms* where, starting from a bicluster seed, elements are inserted as first improvement. There are also some *distribution parameter identification* algorithms that try to identify the distribution parameter that can generate the data by minimizing a certain criterion. The last group is represented by *divide et impera* algorithms, where one can start from the overall matrix and subdivide it iteratively until the result can be considered a good grouping.

Some of the most famous algorithms for biclustering are the following, grouped according to Madeira and Oliveira:

Two-way clustering

- **CTWC** (Coupled Two-Way Clustering), that combines rows and columns clusters searching for non-overlapping biclusters. These are ordered and at the next step only the best is kept [17];
- **ITWC** (Interrelated Two-Way Clustering), that is similar to CTWC but uses kMeans for clustering [18].

Divide et impera

- **Block clustering** was the first divide et impera approach based on a top-down strategy that starts with big blocks of data and separate them until the overall variance reaches a certain threshold [12];
- **Bimax** that searches for blocks of 1 in a binarized matrix in which these blocks are over-expressed genes. These algorithms recursively subdivides the matrix in blocks and stops when each block represents a bicluster [19].

Greedy search algorithms

- **CC** (Cheng & Church): this algorithm is based on a greedy heuristic to converge to a locally optimal bicluster with score smaller than a threshold [10]. It is based on a set of iterations: for each of them an initial bicluster is increased with the addition of one more element that gives the best score to the bicluster itself. The algorithm finishes when the bicluster score exceeds the input threshold. In the same work the Authors introduce the MSR formula (here called H score) to be used as a reference measure for the score;
- **FLOC** (FLexible Overlapped biClustering) extends Cheng and Church algorithm by dealing with missing values via a threshold. The algorithm is developed in two phases: the first is the

creation of biclusters based on the use of a parameter that establishes its increase, and in the second phase, bicluster is built by using the selected threshold [20];

- **OPSM** (Order-Preserving SubMatrices) searches sub-matrices inside the starting matrix, preserving the order of columns. This can happen only in good biclusters [21];

- **xMotif**: biclusters are built following a given bicluster with coherent evolution. An xMotif is one with coherent evolutions on rows. The algorithm searches for the set of rows (I) and columns (J) that follow the xMotif [22].

Exhaustive algorithms

- **SAMBA**: it simply searches exhaustively for the best biclusters [23]. The goal of the algorithm is the identification of a maximum weight subgraph of a bipartite graph. It assumes that the weight of the sub-graph corresponds to its statistical significance.

Distribution parameter identification

- **Plaid Models**, that looks at the matrix as an image. This algorithm assigns a value to each of the elements and tries to give an order at the matrix elements in a way that one can do the assumptions that specific blocks are biclusters [24].

Other algorithms

Other biclustering algorithms not grouped in [15] are the following:

- **PSM** (Particle Swarm Optimization), that emulates the movements of groups of swarm birds searching food to group the elements as animals do [25];

- **ISA** (Iterative Signature Algorithm) considers a bicluster as a module with regulated genes associated with regulated conditions. These initial biclusters can be grown according to a threshold and a measure of quality. The algorithm is composed of a set of iterations that run until the modules do not change anymore [26];

- **CCC** (Contiguous Column Coherent) is a method by Madeira and Oliveira that finds patterns in contiguous columns. This method is applied to time series datasets [27];

- **cMonkey** is an algorithm that searches for co-expressed and co-regulated genes for example those sharing the same regulatory control or also those genes whose products form a protein complex. This algorithm implies a pre-processing step to discover or set these similarities, reduce the dimensionality of the data and reduce the noise [28];

- **FABIA** is a multiplicative model that searches for linear dependencies between genes and conditions. It uses the Bayesian models to discover statistical rules from the behaviors of the genes at each experiment [29];

- **QUBIC** (QUalitative BIClustering) is a model by [30] where the initial matrix is transformed in one with values that corresponds wherever the genes are similar. Transforming it in this way, the final step is to search for sub-matrices that have similar elements. This will be a bicluster.

2.2.2 Meta-heuristics

When a problem is computationally intractable, an *heuristic* is a method to find an *approximate* solution in a reasonable computational time. In combinatorial optimization, *meta-heuristics* are

combinations of algorithms thought to solve generic intractable problems. The following are some of the meta-heuristics used to address the problem of biclustering:

- **Genetic algorithm**: it is based on some iterations inspired to principles of selection and evolution proposed by Darwin in "The origin of species". Implementations can be found in [31], [32] and [33];
- **Simulated Annealing**: the idea comes from the annealing process in metallurgy: it regards the controlled heating and cooling of a material to modify its dimensions. If bad solutions are accepted then the temperature of the system is lowered, thus only a limited number of bad solution are accepted [34]. Accepting worse solution is critical in meta-heuristics since this way to proceed can allow to a deep search of the optimal solution. Implementations are found in [35] and [36];
- **GRASP** (Greedy Randomized Adaptive Search Procedure) will be explained in the next chapter. It has been largely used to solve several problems in different fields. Applications can be found in telecommunication [37],[38],[39], electrical transmission problems [40],[41], biology and related fields [42],[43].

GRASP meta-heuristic together with two algorithms for biclustering of gene expression data will be explained thoroughly in the next chapter. The first is the algorithm from which this work has started. Some drawbacks have been overcome by modifying it. The other is an algorithm used for the same purpose introducing an *iterated local search*.

2.3 Measures of distances

To evaluate distances between two vectors in the clustering procedures used, the two better formulas are the *Cosine Similarity* and the *Pearson correlation coefficient.*

Cosine Similarity

The *cosine similarity* is used to measure the distance between two vectors in n dimensions. It gives the cosine of the angle between them. Given the elements of the vector in a 2D graph, the formula also expresses a similarity between the trends of the curves.

Given two vectors A and B of size n, the cosine similarity is represented using a dot product and the norm of vectors. The following is the formula:

$$cosSim = \frac{A \cdot B}{\|A\|\|B\|}.$$

Cosine similarity is 1 when the two vectors are equals and -1 when they are exactly opposite.

Pearson's correlation coefficient

Another way to obtain a measure of distance between a vector x and a vector y is by using the *Pearson's correlation coefficient.* Since it is based on the use of the covariance, this coefficient is optimal when there is a linear relationship between two variables.

Attributes of a vector can be seen as a series of measurements of X written as x_i and y_i where $i = 1, 2, ..., n$. Given two different vectors and measurements, the Pearson coefficient can be used to estimate the correlation c between X and Y. It is written in the following way:

$$Corr_{XY} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{(n-1)\sigma_x\sigma_y},$$

where μ_x and μ_y are the sample means of X and Y, σ_x and σ_y are their sample standard deviations.

Chapter 3

Our algorithm

Love is guided by rationality and irrationality. It is not simply an emotion, but many of them built during a wonderful time.

3.1 GRASP

A set of techniques to best solve the problem of biclustering have been studied. All are deployed using as framework GRASP (Greedy Randomized Adaptive Search Procedures) [44, 45]. This is a multi-start metaheuristic made up of:

- a **construction phase**, where a feasible solution is built in a greedy, randomized and adaptive manner;
- a **local search phase**, which starts from the constructed solution and searches a locally optimal solution.

GRASP has been used to produce high-quality solutions for hard combinatorial optimization problems and it has been applied in many fields recently and in the past. An overview on the fields where GRASP has been applied is reported in [46, 47, 48].

GRASP is a multi-start procedure: a number of iterations occurs before that it converges to an approximate solution. This number is decided by the user and could be: a maximum number of iterations, a set of iterations without improvements or one can decide to run the algorithm until a certain quality of the solution is reached.

3.1.1 Construction phase

During the construction phase a new element has to be added: the choice is determined by the so-called *candidate list* C created using a **greedy** function $g : C \to \mathbb{R}$ which in our case is given by the mean square error (H). All elements, that hold the cost to be feasible, enter in a *restricted candidate list* (RCL) (Figure 3.8 line 10). The heuristic is **adaptive** because the costs associated with the elements change reflecting the improvement obtained with the last entry. The **probabilistic** component of our algorithm is given by the random choice of the element to be inserted from RCL (Figure 3.8 line 11). Therefore it is not necessarily the candidate that maintains the H lower.

Then, assume that $g : C \to \mathbb{R}$ gives the incremental cost associated with the insertion of the element in the solution and that g_{min} and g_{max} are costs minimum and maximum respectively, i.e.

$$g_{min} = \min_{c \in C} g(c), \qquad g_{max} = \max_{c \in C} g(c). \tag{3.1}$$

The RCL is composed of elements $c \in C$ with which one can get the best incremental costs $g(c)$. To build the RCL has been implemented the *value-based* (VB) method. So the creation of RCL is associated with a parameter $\alpha \in [0, 1]$ and a threshold value $\mu = g_{min} + \alpha(g_{max} - g_{min})$. All the elements, whose insertion causes to not exceed the threshold μ, are taken.

In this work has been implemented the *reactive* version of GRASP meta-heuristic framework. The term *reactive* means that, during the

construction of the RCL, the parameter α is selected from a finite set of elements with a probability that depends on the insertion of the elements made so far.

One of the ways to obtain the reactive behavior is to use the formula proposed in [49].

Assume that the set of possible values for α is given by: $\Delta = \{\alpha_1, \alpha_2, \ldots, \alpha_\ell\}$ (Figure 3.5 line 1). At the first iteration all elements ℓ have the same probability of being selected (lines 2–4):

$$p_{\alpha_i} = \frac{1}{\ell}, \qquad i = 1, \ldots, \ell. \tag{3.2}$$

At each iteration the incumbent solution value is $\hat{z}$ and A_i the average of the values of all the solutions found using $\alpha = \alpha_i$, $i = 1, \ldots, \ell$. Probabilities are recomputerd (Figure 3.10 line 13 and Figure 3.11 line 13) using the following function:

$$p_i = \frac{q_i}{\sum_{j=1}^{\ell} q_j}, \tag{3.3}$$

If one of the α_i leads to better results then that α is taken leading, in this way, to a general improvement of the incumbent solution. This operation is performed at each iteration.

To attempt to improve each iteration of the algorithm a **local search** replaces the current solution with a better one in its neighborhood.

3.1.2 Local search

The *local search* is a process that is used in many heuristic methods for combinatorial optimization. It can be seen as a step sometimes necessary to find good approximate solutions. The idea is to have a set S of solutions and a cost function $g : S \to \mathbb{R}$ and each solution $s \in S$ maps to a real value that is its cost. The local search has the intent to minimize the cost of the objective function g:

$$min\{g(s), s \in S\} \tag{3.4}$$

A neighborhood $\mathcal{N}$ for a solution s is given by the set of solutions $\bar{s_1}, .., \bar{s_n}$ that can be obtained with small modifications of s.

A solution $\hat{s}$ is a local minimum of g for the neighborhood $\mathcal{N}$ when:

$$g(\hat{s}) \leq g(y), \forall y \in \mathcal{N}(s) \tag{3.5}$$

This is performed in a series of successive iterations that may be of various types. For example, a greedy approach would choose the best solution obtainable throughout the set of neighbors but, since often the neighborhood is very wide, one can choose different techniques such as taking the first solution that improves the current or the best one.

The fact is that the larger the neighborhood the more are the evaluations carried out and the higher the computational complexity of the algorithm.

The reactive GRASP has lead to many improvements over the generic framework. They can be seen both in the planning of networks for energy transmission [40] or into problems of capacitated location [50].

3.2 GRASP-like algorithms

The algorithms have been designed taking inspiration from GRASP and modifying it to obtain some variants that change the behaviour of the algorithm for certain conditions of the microarray data biclustering problem. For example, dataset to analyze can be different, values and dimensions are not always the same, the number of genes can be more than conditions and cluster columns first and then rows is slightly different. These and some other aspects of

the problem lead to decide to hybridize GRASP with other heuristic/metaheuristic approaches.

An algorithm has been designed that implements biclustering beginning from a clustering of rows and columns [51], then by combining the best of the resulting in some bicluster seeds that can be increased in a greedy-random way until the score H of the bicluster remains below a certain threshold. This technique seemed limited for three main reasons:

1. it is only possible to insert elements and not to erase them. After clustering, the larger sized seeds are discarded because they have a higher H score, with only very small seeds. It is then assumed that doing only the insertion is a limit for the construction of a good bicluster;

2. during the insertion, the neighborhood chosen is the whole set of elements that remain outside of the bicluster. This choice gives always the best value but worsens the algorithm in terms of performance;

3. the computational complexity [1] of this local search depends from the number of genes and from the computational cost of the H score: $\mathcal{O}(|X|(|I||J|)^2)$. The overall GRASP-like algorithm builds the bicluster inserting, in the worsen case, all the elements of the dataset and at each iteration applies the local search: $\mathcal{O}(|X|^2(|I||J|)^2)$. This is done first for rows and then for columns. The value is the same as the columns are usually less then the rows.

[1] The computational complexity of a procedure (for a given input) is the number of elementary instructions that it executes. This number is computed with respect to the size n of the input data. The notation $\mathcal{O}$ (big "O" notation) is a symbol used to describe the asymptotic behavior of a function. In particular it is used to understand how quickly a function increases or decreases. It is used an "O" because the growth rate is called the "order." Thus we can write: $p(n)$ grows in the order of n^2 as $p(n) = \mathcal{O}(n^2)$ [52].

Our technique started from this algorithm and then searched for ways to improve its efficiency especially in terms of computational complexity.

3.2.1 Overview on the algorithms design

An algorithm that performs a local search has been implemented, it is based on a 3-step quick insertion/elimination of elements taken randomly from the whole and that are somehow similar to those already present within the bicluster. In this first step only the speed of the algorithm has been improved while, instead comparing the results obtained with the database of Gene Ontology it was found that biological significance of results are similar.

The next step is represented by the attempt to improve the clustering technique used in the procedure. Therefore it has been chosen to implement a MST clustering based on the algorithm of Prim [53].

A further attempt has been to introduce some sort of intensification in the local search with iterated local search.

3.3 Clustering algorithms

GRASP-like algorithm starts with the creation of biclusters seeds using a simple clustering procedure. It builds a set of clusters of rows and another for columns and combine them together to obtain the best ones as a starting point.

Two clustering algorithms have been implemented in order to obtain the bicluster seeds required for our purpose: the first is based simply on kMeans and the second on the theory of the minimum spanning tree.

3.3.1 kMeans

kMeans [54] is a well known and frequently used algorithm for the clustering of objects. In kMeans algorithm, objects have to be represented with numerical values. In addition the user has to specify the number of groups (referred to as k) to be identified.

In Lloyd's algorithm each object is represented by n features. All these features together create a vector in an n dimensional space and they can be thought as positioned in this space and, so, they can be ordered by proximity.

When the algorithm starts, one has to choose the number k of points in that space that will be the initial centers (centroids) of the clusters (they may also be randomly chosen). Then the algorithm begins with a series of iterations, each of which is composed of a step in which all objects are assigned to the nearest center. Also the distance measure can be chosen by the user, determined by the kind of task.

At the end of the i-th iteration, the centroids are recomputed by averaging the vectors of all objects assigned to it (see Figure 3.1). The process will continue re-assigning elements to the closest centers and re-positioning them. The algorithm is proven to converge after a finite number of iterations: the iterations stop when the centroids do not change anymore or their variability is very slow in the last iterations.

The computational complexity of the kMeans algorithm depends on the size of the feature vector from the k chosen and from the number of entities n that have to be grouped. It is: $O(n^{k+1} \log n)$.

3.3.2 MST clustering

A *weighted undirected graph* $G = (V, E)$ is formed by a set of points called **nodes** (elements of the set V) and a set of pairs of

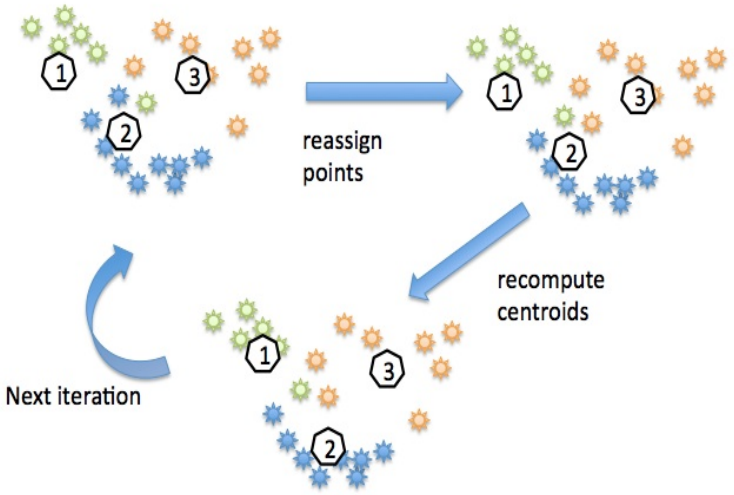

Figure 3.1: A single iteration of the kMeans algorithm. The points are all assigned to the nearest centroid and then these ones are recomputed to assume the position of a perfect center.

points that will form the **edges** (elements of the set E). For each edge is assigned a weight.

A graph is said **connected** if it has at least one path for each pair of nodes.

A **tree** is a connected graph without circuits

A **spanning tree** is a tree in G that contains all nodes.

Suppose that each edge has a weight and call "weight of the graph" the sum of the weights of the edge contained, then we can give the definition of **minimum spanning tree**. It will be the one whose weight among all the spanning trees is minimum.

In Figure 3.2 are shown: a) a connected graph; b) a spanning tree that is not a minimum; c) a minimum spanning tree, obtained by choosing the edge of minimum weight to create the trees.

Let us define some ways to describe elements:

- a partition of nodes of graph G is a division into disjoint nonempty subsets $P_1, .., Q_1$;

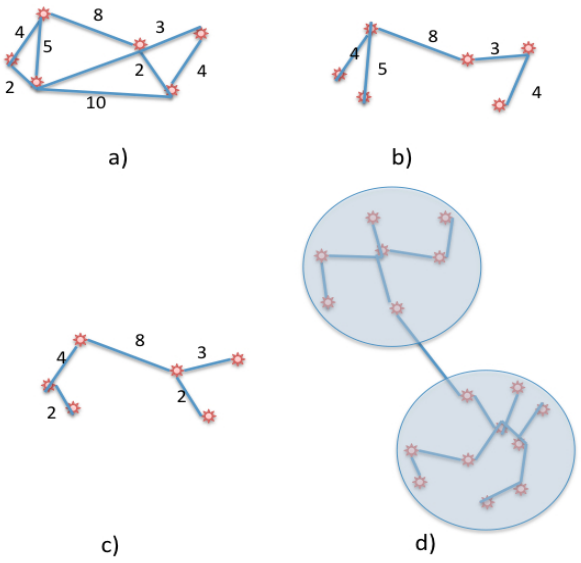

Figure 3.2: An example of graph and spanning tree. (a) Weighted linear graph; (b) Spanning tree; (c) Minimal Spanning tree; (d) Example of a MST clustering. When the inconsistent edge is cut there will be two different groups.

- the distance $\rho(P, Q)$ between two partition P and Q is the minor weight among all edges which have one end node in P and the other in Q;
- the distance between two nodes i and j will be defined as $\rho(i, j)$;
- upper cases will differentiate partitions from nodes.

Prim Algorithm

Prim's algorithm [53] is a greedy algorithm that, starting from a connected weighted undirected graph, finds a minimum spanning tree. To do this, it looks for subsets of edges that create a tree containing all vertices. The characteristic of MST is that the total weight of the edge within the tree is minimal.

The algorithm was designed in 1930 by the czech mathematician Vojtech Jarnik and, only later, the computer scientist Robert C.

Prim made a first implementation (1957) [53]. The algorithm was then taken up by the dutch Edsger Dijkstra in 1959. Sometimes it is in fact called the DJP algorithm.

The only spanning tree of the empty graph (with an empty vertex set) is again the empty graph. The following description assumes that this special case is handled separately.

The algorithm continuously increases the size of a tree, one edge at a time, starting with a tree consisting of a single vertex, until it spans all vertices (Figure 3.3).

procedure `prim-algorithm` $(G = (V_0, E_0))$
/*G is a non-empty connected weighted graph (weights can be negative)*/
1 $V_1 = \{x\}$, /* where x is an arbitrary chosen vertex in V, $E_1 = \{\} * /$ $\mathcal{O}(1)$
2 **repeat**
3 Take an edge (u, v) s.t. $u \in V_1$ and $v \notin V_1$ and $\rho(u, v)$ is minimal weight $\mathcal{O}(|V|^2)$
/*if more than one edges have the same weight, any of them must be picked)*/

4 Add v to V_1 and (u, v) to E_1 $\mathcal{O}(1)$
5 **until**
6 $V_1 = V$;
7 **return** V_1, E_1
end

Figure 3.3: Prim's Algorithm. Algorithm for MST Finding. The computational cost is $\mathcal{O}(|V|^2)$

The Prim algorithm has been realized using the adjacency matrix graph representation and using an array of weights. The computational cost is $\mathcal{O}(|V|^2)$ because of the search of minimal weight edges in the adjacency matrix.

Edge cutting

After the creation of a minimum spanning tree, a set of sub-trees is obtained. These sub-trees are the final clusters. The only problem is that they are still connected by edges which then must be cut.

For a completely greedy approach, longer edges must be cut [55], but this approach does not lead always to good results. The first used is therefore a completely random approach where, given the graph $G = (V, E)$ one erase k random edges: $e_1, .., e_k$. This approach is proved to be good when matched with our previous algorithm implemented with kMeans clustering.

Greedy-randomized cut

A second implementation was the *greedy randomized*, where one collects a series of longer edges in an RCL (restricted candidate list). This approach is similar to that explained in the GRASP method. The RCL is made up of elements $e_i \in E$ with the best incremental costs (in this case longer edges). There are two main mechanisms to build this list: a *cardinality based* (CB) and a *value based* (VB) mechanism. In the CB case, the RCL is made up of the k elements with the best incremental costs, where k is the parameter. In the VB case, the RCL is associated with a parameter α and a threshold value. All the candidate elements whose incremental cost is no greater than the threshold value are inserted in the RCL:

$$RCL \leftarrow \{e \in E | \rho(e) \leq e_{min} + \alpha(e_{max} - e_{min})\}$$

where

$$e_{min} \leftarrow min\{\rho(e) | e \in E\}$$

$$e_{max} \leftarrow max\{\rho(e) | e \in E\}$$

E is the edge set, α is the RCL threshold parameter, $\alpha \in [0..1]$.

The amount of greedyness and randomness are controlled by the α parameter. When $\alpha = 0$ the algorithm is completely greedy, instead for $\alpha = 1$ the algorithm is completely random.

The next element to be added to the partial solution is randomly chosen from the elements in the RCL.

```
procedure mst-clustering(k,distMatrix)
1  distMatrix := initialize − matrix();     /*distances matrix O(|V|^2)*/
2  Prim-algorithm(distMatrix);                          O(|V|^2)
3  greedy-random-cut(distMatrix, k);                    O(k · |E|)
end
```

Figure 3.4: Pseudo-code of `mst-clustering` procedure. It is invoked in our GRASP-like algorithm. The computational complexity is given by $\mathcal{O}(|V|^2 + |E|)$.

The overall MST clustering algorithm is represented in Figure 3.4 where there are only three main steps. After the initialization of matrix of distances (line 1), Prim's algorithm is executed to create the minimum spanning tree from the graph (line 2) and finally the cut is applied (line 3). The algorithm used for the cut is the greedy-random one just explained.

The computational complexity of the overall algorithm depends from the computational time in the Prim's algorithm and from `greedy-random-cut` that erase the longer edges from the list of vertex in a greedy-random way and depends from the number k of cuts and that of edges. It is given by: $\mathcal{O}(|V|^2 + |E|)$.

3.4 Reactive GRASP-like algorithms for biclustering

In [51] a GRASP for biclustering is implemented. It is implied only in the construction of the biclusters and, since a GRASP is involved for all the process to find a solution for that problem, it is more commonly viewed as GRASP-like.

To have a complete GRASP all the biclusters found by the algorithm have to be considered together as the solution for the problem.

The cited solution takes one single bicluster and applies on it a procedure where elements are inserted considering a set of iterations where the best element are chosen using the restricted candidate list.

To search for the best element to insert at each iteration the neighborhood used is composed by all the possible elements (genes or conditions) of the whole matrix in input. As said before this method is expensive in terms of computational time.

In our approach, the neighborhood used has been changed in the way that will be explained in the following paragraphs.

algorithm `GRASP-like-biclustering`($\mathcal{A}$,`MaxNoImpr`,`MaxDist`,δ)
1 $\Delta := \{\alpha_1, \ldots, \alpha_\ell\}$; /* $\alpha_i \in [0,1]$, $i = 1, \ldots, \ell$ */
2 **for** $i = 1$ to ℓ **do**
3 $p_{\alpha_i} := \frac{1}{\ell}$;
4 **endfor**
5 $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$:=`filtered-Kmeans`($\mathcal{A}$); /* $H(\mathcal{B}_q) \leq \delta$, $q = 1, \ldots, k$ */ $\mathcal{O}(|X| \log |X|)$
6 **for** $q = 1$ to k **do**
7 $\hat{\mathcal{B}}_q$:=`grasp`($\mathcal{B}_q$,Δ,$\mathcal{A}$,`MaxNoImpr`,`MaxDist`); $\mathcal{O}(|X|(|I||J|)^2)$
8 **endfor**
9 **return** ($\hat{\mathcal{B}} = \{\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_k\}$);
end

Figure 3.5: Pseudo-code of the proposed GRASP-like algorithm.

In our GRASP-like the stopping criterion is the achievement of a maximum number of iterations without improvements (`MaxNoImpr`) and we implemented the reactive type of the metaheuristic framework. The pseudo-code is reported in Figure 3.5.

The computational complexity of the overall GRASP-like algorithm depends obviously from the applications of the procedures inside it. Therefore it will be explained after all the algorithms.

3.4.1 Biclustering using a social dynamic

This new approach is cheaper in execution time with respect to the approach of Dharan Nair in [51]. This latter, although robust and elegant, has a local search too heavy and that does not guarantee results, in terms of biological results, better than those obtainable using faster processes.

It considers in the neighborhood not all the elements of the matrix and tries to fetch them in an intelligent way. The question is: what neighborhood may be appropriate in the case of gene expression data?

The real problem is that one never knows what might be the right neighborhood for a bicluster because at any given time its composition depends from elements that are not helpful in understanding what are the best required. A bicluster, at the moment of its creation, is an object with elements almost not in accord. The "almost" is used because, both in the Daharan and Nair method and in our approach, it has been chosen to use a clustering algorithm to create starting biclusters (such bicluster seed). The seeds contain elements that are similar in terms of spatial proximity, but each condition of the matrix is considered. This could be misleading also because it has to be considered that the bicluster seed usually consists of less than ten elements for both rows and columns. So it is like having the correct recipe to cook a good meal with not all the correct ingredients: the correct recipe alone will not guarantee the right taste. So, finally, what could be the correct neighbourhood of these biclusters?

To overcome this drawback a local search strategy has been designed in which a different neighborhood of a bicluster is used formed by biclusters which have an element more or one less. The element to be removed or added is chosen on the basis of both the proximity to

the bicluster (distance from the mean vector) and the improvement obtainable in terms of objective function. The objective function used is the MSR (minimum square residual) previously described and indicated with H.

This approach is based on a *social dynamic*. Suppose the existence of a group of friends. This happens when some people initially know themselves and constitute a group (this will be a **seed bicluster**). This group is coherent because the people share some similar behavior or interests (this fact is represented by the bicluster coherent evolution). During the time the group can change: someone goes because of different interests or someone goes because his behavior is no more compatible with the group. The latter happens when new people take part to the group and the interests are changed.

This natural dynamic has been represented as a GRASP-like algorithm that makes some choices during the *local search*.

Each time that a new element is taken from the RCL it is considered like a feasible friend to add to the group. The neighborhood of this candidate is formed by other elements suitable to the cluster (that keeps H in a determined threshold) so a simple local search where an element more is added or not has been chosen to obtain this behaviour. It is extracted randomly from all the possible choices and if it is near to the solution and improve the objective function it will be inserted. Possible situations can be seen in Figure 3.6. In red the mean vector of the bicluster. Randomly another element from all is extracted. If it is:

- (violet) far from red and from the cluster, do not insert it;
- (blue) near to red and not in the bicluster, add it;
- (green) near to red and already present, no action;
- (yellow) far from red and still present, erase it;

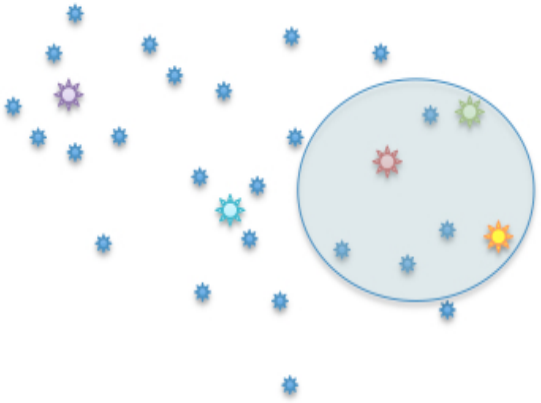

Figure 3.6: Graphical example of the proposed local search. Red is the mean vector of the bicluster. A new element is extracted randomly from all. Depending on the distance from the mean vector it will be erased or inserted.

This is the idea that a friend of someone entered the group and other people make considerations on his behavior. This friend may be accepted by the group or not. Then two other action are performed: one is to erase the element farthest from the mean and to remove a random element taken from the bicluster far from the mean (these eliminations are done if and only if the solution remains suitable). These actions corresponds to reject persons that are no more suitable to the group, because the new friend permit to the group to modify it is behavior and preferences coherently with its insertions.

3.4.2 Inside our GRASP-like algorithm

The overall algorithm takes inspiration from Dharan-Nair [51]. It begins with a partial solution formed by a set of k biclusters $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$ created by applying a clustering algorithm. The partial solution will consists of the so-called "seeds" that will be made such that $H(\mathcal{B}_q) \leq \delta$, where delta is a given input parameter.

The method continues iteratively (Figure 3.7) trying to get the best local solution considering first the columns (lines 8–15) and then the rows (lines 1–7). The iterations continue until `MaxNoImpr` of them are reached without improving the current solution.

```
procedure grasp(B_q,Δ,A,MaxNoImpr,MaxDist)
1   count := 0;
2   repeat
3       (c,B̄_q):=build-columns(B_q,Δ,A);                          O(|Y|(|I||J|)^2)
3       (bool,B'_q):=local-improvement-columns(c,Δ,B̄_q,A,MaxDist); O(|I|^2|J|)
4       if (bool) then count := 0;                                 O(1)
5       else count := count + 1;                                   O(1)
6       endif
7   until (count =MaxNoImpr)
8   count := 0;
9   repeat
10      (c,B̄_q):=build-rows(B'_q,Δ,A);                            O(|X|(|I||J|)^2)
11      (bool,B'_q):=local-improvement-rows(c,Δ,B̄_q,A,MaxDist);    O(|I||J|^2)
12      if (bool) then count := 0;                                 O(1)
13      else count := count + 1;                                   O(1)
14      endif
15  until (count =MaxNoImpr)
16  return (B');
end
```

Figure 3.7: Pseudo-code of `grasp` procedure. It is invoked in our GRASP-like algorithm.

At the end of the procedure is given in output the best incumbent solution (line 16).

Both procedures `build-rows` (Figure 3.8) and `build-columns` (Figure 3.9) take as input a partial bicluster $\mathcal{B}_q$ and try to insert elements in order to increase the number of consistent items.

The computational complexities of the `build-rows` and the `build-columns` procedures are given respectively by $\mathcal{O}(|X|(|I||J|)^2)$ and $\mathcal{O}(|Y|(|I||J|)^2)$ due to the computation of the H scores in the loop

```
procedure build-rows(B_q,Δ,A)
1  C := ∅; g_min := large; g_max := 0;        /* B_q = (I_q, J_q), A = (X, Y)  O(1) */
2  for each x ∈ X \ I_q do
3      C := C ∪ {x};                                                    O(1)
4      g(x) := H(I_q ∪ {x}, J_q);          /* mean squared residue */   O((|I||J|)^2)
5      if (g_min > g(x)) then g_min := g(x);                            O(1)
6      if (g_max < g(x)) then g_max := g(x);                            O(1)
7  endfor
8  α :=extract(Δ);                                                      O(1)
9  μ := g_min + α(g_max − g_min);                                       O(1)
10 RCL:= {c ∈ C | g(c) ≤ μ};                                            O(|X|)
11 c :=extract(RCL); I_q := I_q ∪ {c};                                  O(1)
12 return (c,B_q = (I_q, J_q));
end
```

Figure 3.8: Pseudo-code of `build-rows` procedure. It is invoked in our GRASP-like algorithm.

```
procedure build-columns(B_q,Δ,A)
1  C := ∅; g_min := large; g_max := 0;        /* B_q = (I_q, J_q), A = (X, Y)  O(1) */
2  for each y ∈ Y \ J_q do
3      C := C ∪ {y}                                                     O(1);
4      g(y) := H(I_q, J_q ∪ {y});          /* mean squared residue */   O((|I||J|)^2)
5      if (g_min > g(y)) then g_min := g(y);                            O(1)
6      if (g_max < g(y)) then g_max := g(y);                            O(1)
7  endfor
8  α :=extract(Δ);                                                      O(1)
9  μ := g_min + α(g_max − g_min);                                       O(1)
10 RCL:= {c ∈ C | g(c) ≤ μ};                                            O(|Y|)
11 c :=extract(RCL); J_q := J_q ∪ {c};                                  O(1)
12 return (c,B_q = (I_q, J_q));
end
```

Figure 3.9: Pseudo-code of `build-columns` procedure. It is invoked in our GRASP-like algorithm.

executed for each element of the dataset. In the computation of the overall complexity the one for columns will not be considered

because it can be ignored.

The proposed local search

The last step of the algorithm is the local search. In Dharan-Nair approach the neighbourhood is defined as composed by all possible biclusters that differs from the current one for the presence of one other element of the dataset. In the approach presented the neighbourhood has been built using the algorithm which has been discussed above, based on a social dynamic. This procedure replaces at each iteration a bicluster $\bar{\mathcal{B}}_q = (\bar{I}_q, \bar{J}_q)$ with a new bicluster taken from the neighborhood of $\bar{\mathcal{B}}_q$ which will be formed by all the biclusters that differ from $\bar{\mathcal{B}}_q$ for an element more or/and one less. The element that will be removed or added will be chosen on the basis of its difference from the mean vector of the bicluster (Figures 3.10 and 3.11 lines 3–7) and on the basis of the objective function whose value is given by the mean squared error (Figures 3.10 and 3.11 lines 8–11).

If a best value of the objective function is found, then the probability of selection of the α are re-computed in agreement with this new situation.

After that a new value from the RCL has been extracted it will be inserted in the incumbent solution and then the local search is composed of three steps. Assuming that μ is the mean vector of the bicluster, $\mathcal{T}$ is an inputed threshold and with H is intended the H score, they are the following:

1. extracts a random element x' from the whole set and if $\rho(x', \mu) < \mathcal{T}$ then inserts x' in the incumbent solution (Figures 3.11 and 3.10 lines 2-7);

2. extracts a random element x'' from the bicluster and if $H(\bar{I}_q, D) <$

procedure `local-improvement-rows`(c,Δ,$\bar{\mathcal{B}}_q$,$\mathcal{A}$,$\mathcal{T}$)
1 $H\_score := H(\bar{I}_q, \bar{J}_q)$; /* $\bar{\mathcal{B}}_q = (\bar{I}_q, \bar{J}_q)$, $\mathcal{A} = (X, Y)$ */ $\mathcal{O}(1)$
2 $D := \bar{I}_q$; $new :=$`extract`(X); $\mu :=$ `bicl_mean`$(\bar{\mathcal{B}}_q)$; $dist := \rho(new,\mu)$;
3 **if** $(new \in D)$ **then**
4 **if** $(dist > \mathcal{T})$ **then** $D := D \setminus \{new\}$; $\mathcal{O}(1)$
5 **else**
6 **if** $(dist \leq \mathcal{T})$ **then** $D := D \cup \{new\}$; $\mathcal{O}(1)$
7 **endif**
8 $new := argmin\{d \in D | H(D \setminus \{d\}, \bar{J}_q)\}$; $\mathcal{O}(|\bar{I}_q|)$
9 $D := D \setminus \{d\}$; $new :=$`extract`(D); $\mathcal{O}(1)$
10 $dist := \rho(new,c)$;
11 **if** ($dist > \mathcal{T}$ and $H(D \setminus \{new\}, \bar{J}_q) < H(D, \bar{J}_q)$) **then** $D := D \setminus \{new\}$; $\mathcal{O}(|\bar{J}_q|)$
12 **if** $(score > H(D, J_q))$ **then**
13 `recompute-probabilities`(Δ); $\mathcal{O}(\|\Delta\|)$
14 $\bar{I}_q := D$; $bool := true$; $\mathcal{O}(1)$
15 **else**
16 $bool := false$; $\mathcal{O}(1)$
17 **endif**
18 **return** $(bool,\bar{\mathcal{B}}_q = (\bar{I}_q, \bar{J}_q))$;
end

Figure 3.10: Pseudo-code of `local-improvement-rows` procedure. It is invoked in our GRASP-like algorithm.

$H(\bar{I}_q, D/x'')$ then remove x'' from the incumbent solution (Figures 3.11 and 3.10 lines 9-11);

3. remove the element x''' from the incumbent solution where $H(\bar{I}_q, D/x''')$ is the minimum respect to all the $H(\bar{I}_q, D/y)$ resulting from the removal of all other y of the bicluster (Figures 3.11 and 3.10 lines 8-9).

Each of this removal is performed if and only if the $H(\bar{I}_q, D) < \delta$, where δ is an a priori decided threshold chosen to build the biclusters.

The computational complexity of these procedures is given re-

```
procedure local-improvement-columns(c,Δ,B̄_q,A,T)
1  H_score := H(Ī_q, J̄_q);          /* B̄_q = (Ī_q, J̄_q), A = (X, Y) */        O(1)
2  D := J̄_q; new :=extract(Y); μ := bicl_mean(B̄_q); dist := ρ(new,μ);
3  if (new ∈ D) then
4      if (dist > T) then D := D \ {new};                                    O(1)
5  else
6      if (dist ≤ T) then D := D ∪ {new};                                    O(1)
7  endif
8  new := argmin{d ∈ D|H(Ī_q, D) \ {d}};                                     O(|J̄_q|)
9  D := D \ {d}; new :=extract(D);                                           O(1)
10 dist := ρ(new,c);
11 if (dist > T and H(Ī_q, D \ {new}) < H(Ī_q, D)) then D := D \ {new};      O(|Ī_q|)
12 if (score > H(I_q, D)) then
13     recompute-probabilities(Δ);                                           O(‖Δ‖)
14     J̄_q := D; bool := true;                                               O(1)
15 else
16     bool := false;                                                        O(1)
17 endif
18 return (bool,B̄_q = (Ī_q, J̄_q));
end
```

Figure 3.11: Pseudo-code of `local-improvement-columns` procedure. It is invoked in our GRASP-like algorithm.

spectively by $\mathcal{O}(|I|(|I||J|)^2)$ and $\mathcal{O}(|J|(|I||J|)^2)$ due to the computation of the H scores in the loop acting for each element of the dataset. The difference from the previous local search is significant because the choice of the neighbourhood have a complexity of $\mathcal{O}(|I|)$ due to the search of the worsen element. Instead, the overall complexity depends from the whole gene set. In the computation of the overall complexity the one for columns will not be considered because it can be ignored.

3.4.3 Alterned insertion of rows and columns

One of the limitation of GRASP-like created is that it does not support contemporary insertion of rows and columns. So, it has been thought to include this possibility.

Since bicluster seeds were created using a clustering algorithm, they are a good starting point. The problem is that when the bicluster becomes greater, some columns that have not been added could have been crucial in the bicluster construction. So, it is natural to think to insert columns also after having inserted them in the first step of the algorithm.

For this purpose a variant of GRASP-like algorithm has been created that proceeds in this way: first, it uses the procedure to insert columns and, inside the procedure that inserts rows, a column is inserted each time that is needed. Since the genes are always more than the conditions, it has been thought to add a column each `totRows/totColumns` iterations (Figure 3.12, lines 11-18).

With this different way to proceed, computational complexity is not increased because the alternate insertion of elements complexity can be ignored.

3.4.4 Iterated local search

A local search algorithm tries to find a solution locally optimal $\hat{s}$ starting from a good solution s. To do that, it analyzes the neighborhood of the solution s looking for a solution to have the best value for the objective function g. The point is that using this technique corresponds to search within a local minimum. Figure 3.13 illustrates how it is possible that, if the space of value for g is seen as a curve, it exists the possibility that the solution found is at a point where you can only find local optima.

It was the problem of local optima to push researchers to design

```
procedure graspAlterned(B_q,Δ,A,MaxNoImpr,MaxDist totRows,totCols)
1   count := 0;                                                          O(1)
1   alternRate := totRows/totCols;                                       O(1)
2   repeat
3       (c,B̄_q):=build-columns(B_q,Δ,A);                                 O(|Y|(|I||J|)^2)
4       (bool,B'_q):=local-improvement-columns(c,Δ,B̄_q,A,MaxDist);       O(|I|^2|J|)
5       if (bool) then count := 0;                                       O(1)
6       else count := count + 1;                                         O(1)
7       endif
8   until (count =MaxNoImpr)
9   count := 0;                                                          O(1)
10  repeat
11      if (alternCount < alternRate) then
12          (c,B̄_q):=build-rows(B'_q,Δ,A);                               O(|X|(|I||J|)^2)
13          B̄_q=B̄_q ∪ c;                                                 O(1)
14          alternCount++;                                               O(1)
15      else
16          (c,B̄'_q):=build-columns(B̄_q,Δ,A);                            O(|Y|(|I||J|)^2)
17          B̄'_q=B̄'_q ∪ c;                                               O(1)
18          alternCount=0;                                               O(1)
19      (bool,B''_q):=local-improvement-rows(c,Δ,B̄''_q,A,MaxDist);        O(|I||J|^2)
20      if (bool) then count := 0;                                       O(1)
21      else count := count + 1;                                         O(1)
22      endif
23  until (count =MaxNoImpr)
24  return (B''_q);
end
```

Figure 3.12: Pseudo-code of `grasp` procedure with columns introduction. It is invoked in our GRASP-like algorithm.

new algorithms in order to obtain local searches more versatile and able to avoid this type of error.

The iterated local search allows to avoid the constraint of using only the nearest neighbors of a solution s. From this, in fact, a perturbation is first applied that leads to a new solution s'. If this new solution can pass a test then s' becomes a temporary solution

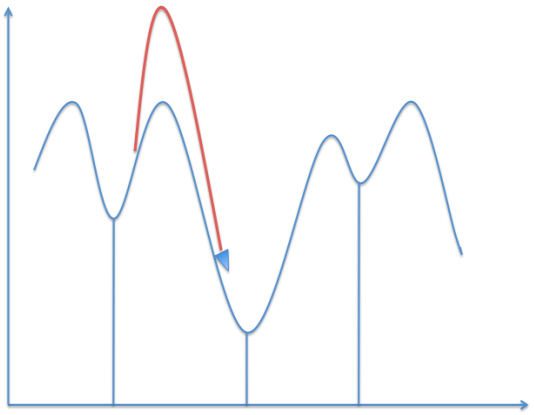

Figure 3.13: An example of iterated local search. One can fall in a local minimum while ideally a global minimum is needed.

whose use is required to arrive to the, possibly global, optimum.

The perturbation of the solution allows, theoretically, to move from one point of the curve of the solutions to another randomly (or guided, depending on how one chooses to implement the algorithm).

The procedure is correct if the perturbation is made in such a way that it is neither too small nor too large. If it is too small there is a risk of being too close to the solution s and have a search space too small. Conversely, if the perturbation is too large, it may happen that the solution s' is really too far from s and this leads the algorithm to become totally random.

The iterated local search has as advantage the fact that leads, in most cases, to better solutions than the one found. One of the problems is that, once the perturbation is done, one might want to go back if the solution produced is not as good. This can be avoided by storing a history of perturbations made. An iterated local search algorithm is in Figure 3.14 where you can see that each of the steps is described: it generates an initial solution (line 1), makes a normal local search (line 2), the solution is perturbed and is found a new

local solution (lines 3-4), a criterion says if it can continue with this solution (line 5).

```
procedure iterated-local-search
1       s_0 = CreateInitialSolution;
2       s = searchLocally(s_0);
3   repeat
4       s' = PerturbateSolution(s, history);
5       s'' = searchLocally(s');
6       testSolution(s'', s', history);
7   until termination condition
end
```

Figure 3.14: Pseudo-code of `iterated-local-search`. This procedure is invoked in our GRASP-like algorithm.

The computational complexity of ILS depends in great part from the history. The overall algorithm can be optimized adjusting the acceptance criteria and the perturbation step that can be wrote using a good transformation dependent from the problem. For example it can be a rule that guides the algorithm in the transformations. An ILS algorithm can have a wide range of complexity that can be added step by step. It is an appealing algorithm also because computes its local searches with interesting speed.

3.4.5 The iterated local search implemented

The iterated local search implemented in this work refers to the work of [56]. This follows the general pattern of the ILS. It can start from any biclustering algorithm and uses the hill-climbing strategy to explore the neighborhood. During each of the iterations one moves within the neighborhood following the objective function chosen. The objective function used in [56] is a new one that aims to evaluate the bicluster. It is based on the use of a different matrix created from the original one.

The matrix M' is achieved by combining a pair of columns (conditions) from the input matrix $\mathcal{A}$. Since the initial matrix has $|X|$ rows and $|Y|$ columns exist $|X|(|X|-1)/2$ possible combinations of columns represented by J''. It will be defined as:

$$M'[i,l] = \begin{cases} 1 & \text{if } \mathcal{A}\text{[i,k]} < \mathcal{A}\text{[i,q]} \\ -1 & \text{if } \mathcal{A}\text{[i,k]} > \mathcal{A}\text{[i,q]} \\ 0 & \text{if } \mathcal{A}\text{[i,k]} = \mathcal{A}\text{[i,q]} \end{cases}$$

where $i \in [1 \dots n]$, $l \in [i+1 \dots J'']$, $k \in [1 \dots m-1]$, $q \in [1 \dots m]$ and $q > k+1$.

From this matrix the behaviors of genes through the chosen conditions can be observed .

Given the incumbent solution $s = (I', J')$ the quality of s is evaluated using the following function:

$$\mathcal{S}(s) = \frac{\sum_{i \in I'} \sum_{j \in I', j < i+1} \mathcal{F}_{ij}(g_i, g_j)}{|I'|(|I'|-1)/2}$$

where $\mathcal{F}_{ij}$ is described by:

$$\mathcal{F}_{ij}(g_i; g_j) = \frac{\sum_{l \in J''_{s_0}} T(M'[i,l]) = M'[j,l]}{|J''_{s_0}|}$$

where

- $T(Func)$ is true, if and only if $Func$ is true, and $T(Func)$ is false.
- $i \in I'$, $j \in I'$ and $i \neq j$, when $\mathcal{F}$ is used by $\mathcal{S}$ and, $i \in I$, $j \in I$ and $i \neq j$ otherwise.
- $|J''s_0|$ is the cardinality of the subset of conditions in M' obtained from s_0,
- $0 < \mathcal{F}_{ij}(g_i, g_j) \leq 1$.

Each result of the function $\mathcal{F}$ attests the quality of a pair of genes subjected to a series of conditions. A high value indicates that the genes are related, whereas a low value indicates the opposite. In the first case the value will be close to 1 otherwise to 0. To compare two pairs of genes will be sufficient to compare: $(g_i; g_j)$ is better than $(g'_i; g'_j)$, when $\mathcal{F}_{ij}(g_i; g_j) > \mathcal{F}_{ij}(g'_i; g'_j)$.

Instead the function $\mathcal{S}(s)$ expresses an average of functions $\mathcal{F}_{ij}(g_i; g_j)$ for each gene and in the same manner the more the value is close to 1 and the more bicluster will be good, this means that genes are correlated. Conversely, a value close to 0 belongs to an insignificant bicluster. When one is going to compare two solutions s and s', s is better than s' if $S(s) > S(s')$.

The algorithm described is shown in Figures 3.15. The algorithm stops when one can no longer find a neighbor in the neighborhood that improves the solution. So the last solution found is considered a local optimum. Once this research phase is over, the algorithm perturbs the solution in order to generate a new starting point. The perturbation changes a certain quantity of items within the bicluster. In particular, it was decided to change a percentage of 10% of the genes between the better ones that are still in the incumbent solution (the bicluster s).

The algorithm terminates when it is unnecessary to go further because the solution does not improve anymore, or has been reached the a priori fixed iterations.

The computational complexity of the iterated local search just described depends from the creation of the behaviour matrix M' that is created each time that the ILS starts. It is: $\mathcal{O}(|X|^2|Y|)$.

```
procedure bils-iterated-local-search(B_0,λ,A)
1   M_B :=create-behav-matrix(A);                          O(|X|^2|Y|)
2   Γ :=compute-F(B_0);                                    O(1)
3   B := B_0;                                              O(1)
4   repeat
5       repeat
6           (g_i, g_j) :=select(B); /* F_ij(g_i, g_j) < λ */   O(|I||J|)
7           (g_j, g_r) :=select(B); /* F_jr(g_j, g_r) ≥ λ */   O(|I||J|)
8           V = {g_v | F_rv(g_r, g_v) ≥ λ}                 O(1)
9           B' ← V                                         O(1)
10          B' → g_i                                       O(1)
11          if (S(B') ≥ S(B)) then B := B';                O(1)
12          endif
13      until (no improving neighbour can be found)
14      B=perturbe-solution(B)                             O(1)
15      until (stop condition is verified)
16  return B
17  end
```

Figure 3.15: Pseudo-code of `bils-iterated-local-search` procedure. It is invoked in the GRASP-like algorithm.

3.5 A multi-start way to proceed

The whole algorithm can be inserted in a multi-start procedure. In this way the algorithm is computed a number of times and the best solution is kept.

This way to proceed is named *Multistart*. Multistart is certainly one of the earliest global procedures used: it has also been used in local optimization for increasing the confidence in the obtained solution [57]. One drawback of multistart methods is that, when many starting points are used, the same minimum will eventually be determined several times. In order to improve the efficiency of multistart this should be avoided.

We give here an example of the execution of our algorithms in multi-start mode.

algorithm `Multi-Start-GRASP-like`($\mathcal{A}$,`MaxIters`,`MaxNoImpr`,`MaxDist`,δ, $\mathcal{B}_0$)
1 $best = \mathcal{B}_0$;
2 **for** $i = 1$ to $MaxIters$ **do**
3 $\Delta := \{\alpha_1, \ldots, \alpha_\ell\}$; /* $\alpha_i \in [0, 1]$, $i = 1, \ldots, \ell$ */
4 **for** $i = 1$ to ℓ **do**
5 $p_{\alpha_i} := \frac{1}{\ell}$;
6 **endfor**
7 $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$:=`filtered-Kmeans`($\mathcal{A}$); /* $H(\mathcal{B}_q) \leq \delta$, $q = 1, \ldots, k$ */
8 **for** $q = 1$ to k **do**
9 $\hat{\mathcal{B}}_q$:=`grasp`($\mathcal{B}_q$,Δ,$\mathcal{A}$,`MaxNoImpr`,`MaxDist`);
10 **endfor**
11 **if** $\hat{\mathcal{B}}$ is better than *best* **then**
12 $best = \hat{\mathcal{B}}$
13 **endfor**
14 **return** (*best*);
end

Figure 3.16: Pseudo-code of our multi GRASP-like algorithm. Here computational complexity contribution are not underlined because they are equal to that in Figure 3.5

As it can be clearly seen, the only addition to the general GRASP-like algorithm is that all the operations are executed a certain number of times. This is because in practice one wants to obtain more than one different global solution for the biclustering and retain only the best one. For each of the iterations the solution is computed and matched with the previous and, if it brings to improvement of the objective function, then the global one is substituted (Figure 3.16 lines 11-12). In this way at the end of the process only the one with the best objective function value (*best*) will be returned (Figure 3.16 line 13). Supposing that are given k biclusters $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$ and their volumes are $V_1, .., V_k$ and their H score values are $H_1, .., H_k$ then the mean for volumes and scores are respectively V_{mean} and H_{mean}. The relations between them V_{mean}/H_{mean} has been used as value to match between the results. If the last is better then the

best obtained, the incumbent solution is substituted.

The computational complexity of a multi start algorithm increases the overall complexity and it depends from the number of time that one wants to re-start the algorithm to have different starting point. In our algorithm they have been not more of three or four times. This very low number is due to the fact that the overall procedure can otherwise be very slow.

3.6 Computational complexity

The computational complexity of the overall GRASP-like algorithm depends on the size of the dataset and those of the biclusters while they are been created.

The procedure `GRASP-like-biclustering` is the reference main algorithm. It calls the procedure `grasp` composed by the following procedures:

- `build-rows`: $\mathcal{O}(|X|(|I||J|)^2)$;
- `local-improvement-rows`: $\mathcal{O}(|I|^2|J|)$;
- `build-cols`: $\mathcal{O}(|Y|(|I||J|)^2)$;
- `local-improvement-cols`: $\mathcal{O}(|J|^2|I|)$.

And before than GRASP-like, one of two clustering procedures is executed:

- `kMeans`: $\mathcal{O}(|X| \log |X|)$;
- `mst-clustering`: $\mathcal{O}(|X|^2 + |E|)$, where $\mathcal{O}(|V|^2 + E)$ became $\mathcal{O}(|X|^2 + E)$ because the vertices of the graph are the genes.

For the overall cost is not considered the part relative to columns because the time to execute is very low respect to that of executing

on rows. The complexities have to be summed obtaining (without the clustering):

$$\mathcal{O}(|X|(|I||J|)^2 + |I|^2|J|),$$

that becomes:

$$\mathcal{O}(|X|(|I||J|)^2),$$

because the first factor is bigger than the second.

This is thus the computational complexity of the `grasp` procedure. It is inserted in `GRASP-like-biclustering` procedure whose complexity is the following:

in the case of the use of kMeans algorithm,

$$\mathcal{O}((|X|(|I||J|)^2) + (|X| \log |X|)),$$

and when MST clustering is used it becomes,

$$\mathcal{O}((|X|(|I||J|)^2) + (|X|^2 + |E|).$$

Chapter 4

Application to microarray data

If love had not been such a wonderful thing it would take far less time to understand

4.1 Introduction

In order to evaluate their performances, the different versions of our algorithm were tested on several microarray data extracted from literature, and available in public datasets. The first dataset used is based on the study of the cell cycle of the yeast *Saccharomyces cerevisiae*, in which the same genes are present in 17 different time series. The other datasets used are from Human lymphoma, *Arabidopsis thaliana* and another one for *Saccharomyces cerevisiae* (called *Yeast environmental*) in which all experiments were made with different conditions or patients. A more detailed description of the dataset is given in the next section.

The biclusters obtained by our algorithms have been evaluated using the GO (Gene Ontology) annotation database and tools online. They comprise a large number of genes that researchers continuously update to give a significance for each cluster. In GO annota-

tion, terms describing biological processes (BP), cellular components (CC), and molecular functions (MF) are assigned to genes, so that a list of genes can be analyzed, looking for terms associated. The statistical significance to which the genes match with the different GO terms or categories can be indicated by the p-value.

The most known databases of Gene Ontologies are **DAVID** [58] and **PANTHER** [59]. We choose to test our algorithms with these websites since they are very simple to use and complete in their functionality.

4.2 Datasets

Here is a little description of the datasets used to assess the quality of the developed algorithm. Indeed, the "consistency" of the bicluster, defined by H-score, is definitely the first parameter to be evaluated, but it is also necessary to consider what is the biological significance of clusters identified, and the ability of the algorithm to evaluate different biological data from different sources and with different purposes. To do this, four datasets of gene expression data from different organisms have been tested.

4.2.1 Yeast time series

This dataset has been taken from [60] and is formed by 2885 genes and 17 time series selected according to [61]. The authors selected these genes as the most variable ORFs using the normalized dispersion in expression level of each gene across the time points. This is the only dataset containing columns representing exactly the same gene taken at different times.

The cells were assessed at intervals of 10 minutes in 17 subsequent times. The intervals allowed to follow two full cell cycles. The cell

cycle phasing has been determined by using the size of the bud, the position of the nucleus in the cell, and the induction pattern.

From each sample, RNA has been isolated, converted to cDNA and hybridized to an array containing the whole yeast genome (6218 genes). The array has been read using a specially designed confocal scanner created by Affymetrix.

4.2.2 *Arabidopsis*

This is a dataset containing 795 genes and 69 conditions based on Wille et al [62].

Plants were grown at 70% humidity and cycles of 16h of light at 21°C and 8h during the night at the same temperature.

The different samples came out from different environmental and grown conditions. RNA has been also extracted from transgenic seedlings or roots exposed to hormonal treatments. The experiments are based on extraction of RNA from wild-type and are mutant seedlings, leaf and seedling in a baseline, a root inducible system exposed to hormonal treatments, seedlings exposed to light and dark conditions in a time-course experiment and to ozone. Another experiment has been performed to assess the effect of inhibitors of pathways on the expression of genes involved in isoprenoid biosynthesis.

To extract mRNA, frozen seedlings and leafs of Arabidopsis were prepared using Trizol (a reagent to deliver high quality RNA from tissue) and purified using RNEasy columns. Fifteen micrograms of RNA were used to prepare the cDNA using a kit according to instructions of Affymetrix that produced the gene chip.

The array has been scanned using the confocal scanner Agilent GS 2500 and raw data processed with the Affymetrix Microarray Suite 5.0 using the default parameters.

The dataset obtained contains only those genes for which relevant

structures from 56 metabolic pathways have been found.

4.2.3 Yeast environmental

This dataset is based on [63]. In this work the Authors have characterized genomic expression in yeast responding to environmental changes.

The cells were subjected to heat shock, hydrogen peroxide, superoxide generated by menadione, a sulfhydryl oxidizing agent (diamide), and a disulfide reducing agent (dithiothreitol), hyper-osmotic shock, amino acid starvation, nitrogen source depletion and progression into stationary phase. This work is important for the formulation of hypothesis for the way in which the yeast responds to environmental changes and stress.

Each condition has been controlled to preserve at least the 80% cell viability. For almost all experiments the samples were collected during the 2-3 hours elapsing between the beginning and the end of the procedure. Cells have been compared also for changes in temperature to check the response to heat-shock.

In the experiment 142 different mRNA samples have been taken by the hybridization of the whole genome.

All the cells have been subject to the extraction of RNA and creation of cDNA samples. Then they have been hybridized on a microarray. The dataset used contains 2993 x 142 genomic data.

4.2.4 Lymphoma

The diffuse large B-cell lymphoma is a disease of the mature B-lymphocites. It attacks every year 25,000 people. Although many patients respond well to chemotherapy initially, only few of them continue with good results. For this reason it is important to know what are the genes that induce resistance to the treatment.

The lymphoma dataset is taken from [60] and based on the work in [64]. The authors realized a special microarray called *Lympho-chip* with genes that are expressed in lymphoid cells with a role in the processes of immunology or cancer.

Only a subset from the overall genes were chosen from a B-cell library because they are suspected to be important for the generation of the non-Hodgkins lymphoma. Some other genes have been added from follicular lymphoma, mantle cell lymphoma and chronic lymphocitic leukemia.

The cDNA has been put on the microarray in 2-3 copies to hybridize the samples obtained.

This dataset is formed by 4026 genes and 96 conditions according to [10], with the expression level reported as an integer value. All array elements for which the fluorescence was greater than 1,4 times the local background were considered well measured. Only the genes with at least the 80% of the mRNA samples well measured were not excluded.

4.3 The Gene Ontology

To evaluate biological significance of biclusters it has been chosen to use the *Gene Ontology*. This is a project whose goal is to unify the way of genes annotation and to have a database where all information about them are stored. Several bioinformatics centers collaborate into the project. The GO project is part of the *Open Biomedical Ontologies* (OBO) [65].

The project provides coverage for three domains:

- **cellular component** (CC), which describes the different cellular compartments or the extracellular content:
- **molecular function** (MF), which describes the activities of

the elements at the molecular level in the genes, such as catalysis and binding;

- **biological process** (BP), which describes the molecular events that belong to living units, such as cells, tissues and organs, which have a beginning and an end well defined.

Each element created by the GO Project is constituted by an unique alphanumeric identifier. For each term there is a description and an indication of the domain to which it belongs. The terms may also have a synonym that exactly belong to the same class. Each term may correspond to different descriptions and can have connections to other databases or comments on how the term is done and how it should be used [65].

The GO project has organized things in such a way that the terms are independent of the species: this means that there can be a GO term reference, for example, to the same gene in both human and in mouse. In addition, these terms can be applied to single and multicellular organisms as well as both prokaryotes and eukaryotes.

Each gene can also refer to more GO terms depending on whether these describe its biological process, molecular function or cellular component. And for each of the descriptions there may be more words that describe it.

The GO ontology is a dynamic project that is constantly evolving. Researchers from all over the world can propose additions, corrections and changes to the team that is responsible for managing the project. The changes suggested are being examined and approved by the maintainer if they are correct [65].

4.3.1 Annotation

When genes are annotated, information are gathered about them and written in an archive. For example after the sequencing and assembly of a genome, alignment procedures can be used to obtain information on their sequences. In the case of the GO, the members submit their annotations to the website where they can be reviewed and approved [65].

Each element of the GO database has an unique alphanumeric identifier and a description. The description contains: a reference to the publication of annotation, an *evidence code* to understand the source of the annotation, creation date and creator of the annotation. The *evidence code* comes out from a controlled vocabulary.

Some of the codes are as follows: *Traceable Author Statement* (TAS) means that the record was taken from a paper; *Inferred from Sequence Similarity* (ISS) means that the output of a similarity search has been used; *Inferred from Electronic Annotation* (IEA) are those records that have been created through an automated process. Currently, more than 95% of the records were inferred automatically.

4.3.2 Tools

There are a large number of tools available both online and offline that use the data provided by the GO project. The vast majority of these come from third parties. Some of them have been used in this work to confirm a biological significance of biclusters found.

The Saccharomyces genome database

The GO tool for Yeast environmental [66] is located at [67] and provides information about genes, proteins and other features of the budding yeast. The website is composed of tools that permit

to have the functional description of budding yeast and to match it with higher organisms.

It also maintans the reference genomic chromosomal sequence of the S. cerevisiae functioning as a central hub where researcher can add and get information.

The database permits to have information about groups of genes given in a form input on the website and gives in output a graphical chart showing the connections and their importances. It also gives a p-value for each function, process, component found.

PANTHER

The website of PANTHER [68] has been realized in order to allow the analysis of large clusters of genes [59]. It contains functional information about genes and can therefore be used:

- to study biological processes, molecular functions, cellular components and pathways;
- to generate lists of genes that have functions in common or that participate in the same biological processes;
- to extract general information about groups or individual genes of specific interest;
- to check which biological processes, molecular functions or cellular components or pathways have in common genes.

Because of this last feature, PANTHER was chosen as one of the systems to validate the biclusters created during this work.

The PANTHER database can be used starting from files that contain lists of genes that should be evaluated. They must be uploaded. The type of identifier has to be chosen and also the organism. The query returns a GO classification corresponding to the molecular functions, biological processes or cellular components in

common with different genes. The different GO terms found for each cluster are associated to a p-value assessing their statistical significance (the lower the p-value, the higher the validity of the statistical association).

DAVID

DAVID is a service developed by the Laboratory of Immunopathogenesis and Bioinformatics (LIB) [69] and is open access [58].

This service provides a functional interpretation of a list of genes that is given as input. The whole system consists of five annotation tools: the DAVID Gene Functional Classification Tool, the DAVID Functional Annotation Tool, the DAVID Gene ID Conversion Tool, the DAVID Gene Name Viewer and the DAVID NIAID Pathogen Genome Browser.

DAVID is a tool which connects a large variety of identifiers and terms from bioinformatic public databases. It also allows conversions of names, provides tools that allow to group lists of genes based on functionalities and allows to graphically display a set of relationships between genes on the basis of data identifiers. It also has the same functionality of PANTHER providing information about biological processes, molecular functions, cellular components and pathways.

DAVID offers various tools for the analysis of Gene Ontology: the conversion of gene names, the functional gene classification and the gene annotation. The latter has been used to evaluate the biclusters.

Unfortunately, the genes do not have a universally accepted identifier but different types have been created to meet different needs. For example, there are names created by the manufacturer Affymetrix, Unigene or EntrezGene. These names are all different and are often made in order to give information on the organism and the type of gene.

The GO database did not provide a way to identify all possible genes. Several genes exist and some databases, as DAVID, have a converter that could not be able to identify all the gene names.

The motivation for the usage of two different datasets is that when we execute queries the systems with our group of genes obtained from clustering it is difficult that all these genes are recognized by a system. In a large number of cases they have to be converted and not all the converters give results for all the possible genes.

In particular PANTHER (Protein ANalysis THrough Evolutionary Relationships) classification system has been largely used with the yeast time series and Lymphoma datasets whereas DAVID was used to classify the results obtained with also the other two datasets.

4.4 Experimental Results

4.4.1 Assessing validity of the technique

The Reactive GRASP-like algorithm has been implemented in C language, compiled with the Apple Xcode 3.1, and ran first on a MacBookPro 2GHz Intel Core Duo running MAC OSX 10.6. Then it has been executed on a 2GHZ AMD Opteron Processor 248 because of the high number of tests. Several iterations have been performed adopting the stopping criterion that counts a maximum number of iterations without improvement of the incumbent solution and inspecting the results obtained.

The initial experiments have been conducted on the Yeast time series dataset [61] and on the Lymphoma dataset [64] to evaluate the quality of the proposed algorithm. In these first tests only our local search algorithm has been implemented without the change of clustering procedure and the multi-start type of execution. This

tests has been performed only on these two datasets to assess the correctness of our procedure.

In Table 4.1 results for 33 biclusters generated for Yeast time series, and 11 biclusters generated for Lymphoma are shown. they are in terms of mean number of genes, mean number of conditions, mean volume, mean squared residue H and mean running time over 10 trials using 10 different random number created by a generator.

Table 4.1: First results with Yeast time series and Lymphoma dataset. In the table the Yeast time series dataset is indicated with **Yeast TS**). Results are in terms of mean p-values. The same results have been retrieved from a set of random biclusters to assess the validity of the proposed method (H_r).

Statistics	Yeast TS Dataset	Lymphoma Dataset
mean number of genes	97,33	59,63
mean number of conditions	10,52	8,18
mean volume	1000,06	478,93
mean H value	195,73	0,03
mean running time (in secs)	4044,43	5012,03
mean H_r value	1821,76	0,56

Mean volumes and number of genes and conditions depend from the bicluster dimensions and object collected by the algorithm. The mean H value is the mean MSR score computed on the final biclusters for each of the 33 (for Yeast time series) and the 11 (for Lymphoma), while the H_r value is the mean of the MSR scores for, respectively, 33 and 11 biclusters randomly created. This has been done to compare our algorithm with a completely random approach to find biclusters. It is evident that this approach outperforms a simple random approach and assess its correctness.

Moreover, looking at the graphical behaviour of the curves for the bicluster (in Figure 4.1 and 4.2) one can note that they are similar in many cases under a subset of conditions. This has proven that

the biclusters from gene expression data identified are coherent.

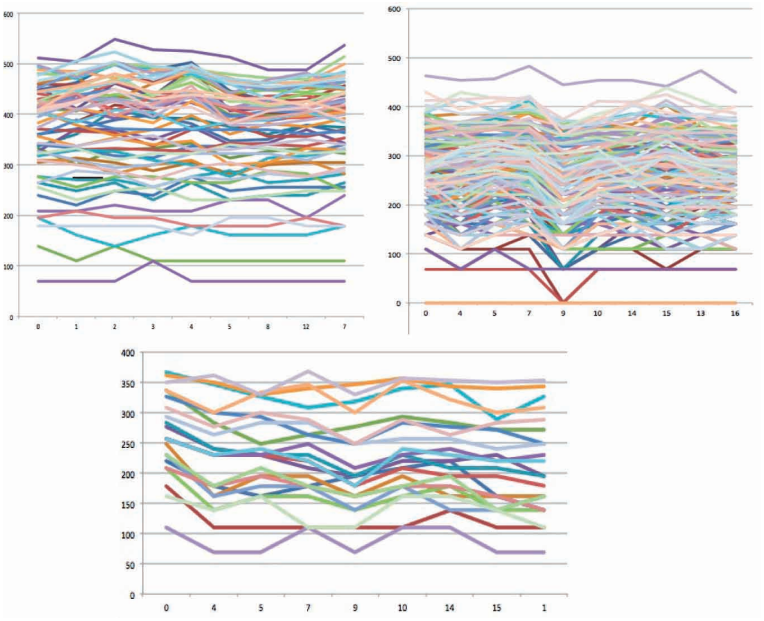

Figure 4.1: Chart of the results with Yeast time series dataset. This is a graphical representation of the expression levels for a sample biclusters obtained in our analysis on Yeast time series dataset [61]. On the rows we have the gene behaviour and on columns the conditions. Different colors of the curves represent different genes.

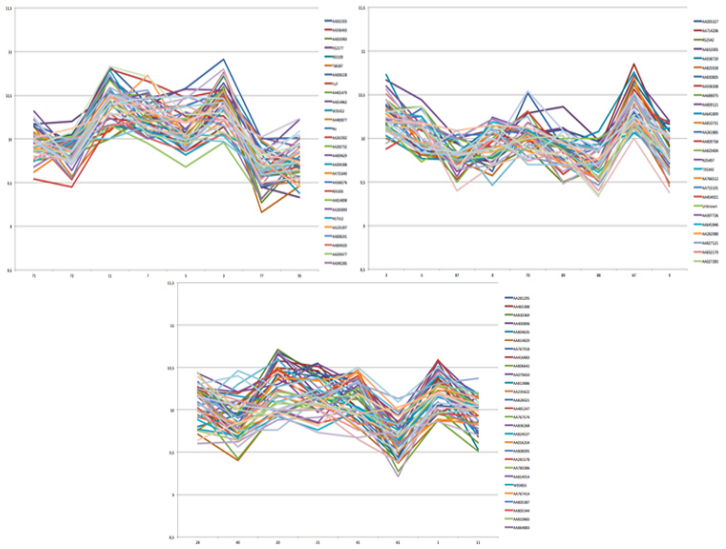

Figure 4.2: Chart of the results with Lymphoma dataset. This is a graphical representation of the expression levels for sample biclusters obtained in our analysis on Lymphoma dataset [64]. On the rows we have the gene behaviour and on columns the conditions. Different colors of the curves represent different genes.

In Table 4.2 we inserted a summary of results using the Yeast GO ontology [67]. They are all in terms of p-value. The first set of rows shows the mean of p-value for biological process, molecular function and cellular component associated to the biclusters found. These value has been taken from the output of the GO website. When a significant p-value is obtained it has been selected and in the case that two or more are significantly associated then only the lowest p-value has been taken. In the GO result a plot can also be shown with the connections between the functions and components found.

Table 4.2: Statistics on results of biclustering on the Yeast time series and the Lymphoma datasets. Yeast time series is indicated with **Yeast TS**. The table shows a summary of the results in terms of p-values.

Mean of minimum p-value	Yeast TS Dataset	Lymphoma Dataset
mean for BP	1,83E-03	1,15E-03
mean for MF	9,28E-04	5,88E-03
mean for CC	1,60E-03	1,38E-01
min for BP	3,89E-15	5,25E-05
min for MF	5,08E-17	3,27E-08
min for CC	6,62E-22	1,05E-03

In Figure 4.3 a graphical example of connections between molecular functions is shown. This is a result for a bicluster where an high percentage of genes belong to the class of structural constituent of ribosome. For this class the color is orange since the *S. cerevisiae* ontology database assigns this color to p-values lower than 1^{e-10}. So this is indicative of an high correlation of elements found inside the biclusters. Also this has been a clear proof that our algorithm performed well on this dataset.

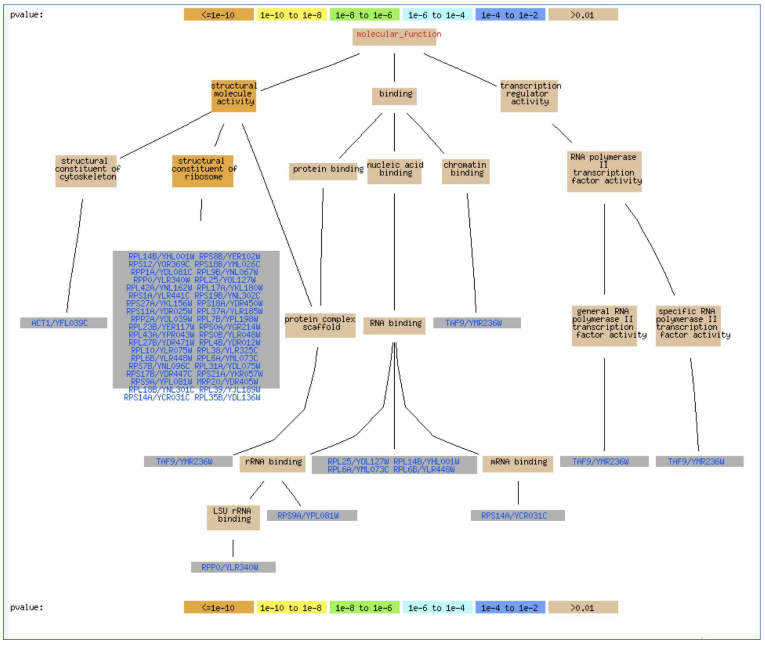

Figure 4.3: Graphical example of molecular functions for Yeast time series dataset. The graphical output comes from the Saccharomyces Genome Database. Colors express the biological significance. The higher is for the orange, the lower is in brown.

These tests showed that there is at least a GO term significant for 29 out of 33 biclusters examined for Yeast time series dataset and for 11 out of 11 in the Lymphoma dataset.

This analysis confirms the coherence of the bicluster analysis with our methods, being most of the gene clusters characterized by a common molecular function, or cellular component, or by the involvement in a biological process. This assesses the goodness of our local search.

4.4.2 Different combinations of algorithms

Assumed that our local search algorithm has a good performance, it was decided to put it in different combinations of heuristics already known in the literature, and that might give better performances to our GRASP-like algorithm. In addition, the same combination

of heuristics was tested, for comparison, on the Dahran-Nair algorithm [51]. Then the results have been evaluated using the GO systems.

The biclustering algorithm can be sub-divided in 3 parts:

1. a clustering step needed to found the initial seeds to upgrade during the process. We implemented two algorithm for this: kMeans (kM) and MST clustering (MST);

2. a step for the "growing" of the bicluster. It has been done with a GRASP-like technique. In this step the local search distinguishes our algorithm (MyGR) from the Dahran-Nair (DNGr) algorithm [51];

3. a final optional step that can be considered as included in the second and that can improve the final solution. This is represented by the iterated local search (ILSAy). A step that can be done after the typical local search and can improve the final solution;

All the process can be executed several times in a Multi-Start procedure (MS).

The overall tests computed are for nine different versions of a biclustering algorithm coming from 9 different combinations of the heuristics previously described. They are shown in Table 4.3.

These results demonstrate that MST clustering algorithm works better than the kMeans. The mean p-value for cellular component is always an high value. In any case the best improvements we find were those obtained with the multi start method and our GRASP-like algorithm.

Table 4.4 represents two different test: the first is realized with the Dharan-Nair algorithm and the following are results with our best version.

Table 4.3: Statistics on the Lymphoma dataset using PANTHER. The table shows a summary of the results for each combination of heuristics in terms of p-values. BP, biological process; MF, molecular function; CC, cellular component. These results have been obtained using PANTHER.

Algorithm	BP	MF	CC
kM+MyGr	1,15E-03	5,88E-03	1,38E-01
MST+MyGr	2,90E-02	1,11E-02	3,70E-01
MST+MyGr+ILSAy	5,19E-03	5,50E-03	8,58E-02
MST+DNGr+ILSAy	7,30E-04	7,86E-04	1,69E-01
MS+MST+DNGr	2,85E-06	1,21E-05	1,51E-01
MS+MST+DNGr+ILSAy	3,21E-06	1,64E-04	1,53E-01
MS+MST+MyGr	3,22E-06	5,72E-05	4,39E-02
MS+MST+MyGr+ILSAy	5,34E-07	1,22E-05	2,31E-02

Table 4.4: Comparing Dharan-Nair algorithm and ours with Lymphoma dataset. Tests realized with the Dharan-Nair algorithm (first three lines) and the best version of proposed algorithm (second three lines) and using Arabidopsis Dataset, Yeast environmental and Lymphoma datasets with PANTHER. (BP, biological process; MF, molecular function; CC, cellular component)

Dataset	BP	MF	CC
Yeast environmental	1,27E-03	2,38E-02	5,29E-01
Arabidopsis	8,52E-04	7,22E-02	1,15E-01
Lymphoma	3,92E-04	3,10E-03	8,43E-01
Yeast environmental	8,77E-04	4,88E-04	2,25E-02
Arabidopsis	4,54E-05	5,20E-02	7,85E-02
Lymphoma	3,95E-05	2,20E-04	1,15E-01

4.4.3 Tuning of parameters

During this work we searched the best measures to use for clustering of microarray data and for biclustering. Using prior knowledge we evaluated the power of the biclusters obtained and we could calibrate

our algorithm to work fine with these type of data.

While experiments were carried out, some parameters were adjusted to obtain the best output. Since this algorithm does not allow for biclusters to overlap, the dimensions of the bicluster are a measure to quantify the effectiveness of this algorithm. The parameters used by our software were related to:

- number of biclusters to create;
- minimum number of rows to achieve in a bicluster;
- minimum number of columns to get in a bicluster.

Inside the clustering methods:

- number of cluster (both for rows and for columns);
- minimum distance to see if two elements are close to or less.

The number of bicluster to be created throughout the work has been the subject of great attention in order to understand if it would be preferable to include the whole set of genes into the biclusters or to optimize the number of biologically significant biclusters.

The most significant decisions on this subject have been taken during the tests carried out on Yeast environmental and Lymphoma datasets.

From our observations we found that it is preferable to obtain a number of bicluster between 10 and 20 and then to increase their size to 50-100 genes.

It has been found that the larger the bicluster and the easier is to get a high biological significance.

The number of clusters of rows that we chose to include is about 150-180. This is because usually the dataset used have a number of genes between 2000 and 4000 and it needs to have very small initial clusters as a starting point.

An automatic choice of the number of clusters has been also provided. If the dataset is bigger than these dimensions, the software decides to set up a number of clusters equal to one tenth of the cardinality of the dataset.

4.4.4 Number of genes considered by the algorithm

The following are some statistics extracted from the composition of the biclusters obtained. They illustrate, for the set of bicluster obtained in a single execution of our algorithms, what is the number of genes considered. It is useful to understand what is the possibility of grouping all the genes in each dataset.

As it can be clearly seen in Table 4.5, Table 4.6 and Table 4.7, the number of genes involved in a result is sometimes far from the overall number of genes.

This is because it is possible that some initial clusters have to be selected several times for the creation of bicluster, because of their very low H-score.

4.4.5 Evaluation of biclusters with DAVID

In the following three tables there is the final evaluation of the results obtained with different combinations of algorithms.

Table 4.5: Number of genes retrieved with Lymphoma dataset. The numbers come from an execution of our best algorithm with Lymphoma dataset. The genes in the column **Num of Genes** are present in a number of biclusters wrote in the second column **Num of Biclusters**. In example 234 genes are present only one time for all the biclusters retrieved.

Num of Genes	Num of Biclusters
234	1
129	2
86	3
36	4
38	5
18	6
8	7
3	8
3	9
1	10
0	> 10
322	comparing in more than one biclusters
556	total genes considered

Table 4.8: Statistics on the Lymphoma dataset. The table shows a summary of the results in terms of p-values using the DAVID GO database. For each combinations of heuristics there is a result for BP, MF, CC.

Algorithm	BP	MF	CC
DNGr	3,5E-5	5,7E-3	1,0E-4
MST+MyGr	1,5E-2	1,98E-2	1,92E-2
MST+DNGr+ILSAy	6,0E-4	1,4E-2	4,6E-3
MST+MyGr+ILSAy	8,4E-3	2,0E-2	2,0E-02
MS+DNGr+ILSAy	1,2E-2	7,4E-2	9,5E-3
MS+MST+MyGr+ILSAy	1,2E-2	1,6E-2	2,5E-2

Table 4.6: Number of genes retrieved with Yeast environmental dataset. These are statistics on the number of genes retrieved on each execution of our best algorithms with Yeast environmental dataset. For description see Table 4.5.

Num of Genes	Num of Biclusters
703	1
176	2
58	3
11	4
0	5
1	6
1	7
0	8
0	9
0	10
0	> 10
247	comparing in more than one biclusters
950	total genes considered

Table 4.9: Statistics on the Yeast environmental dataset. The table shows a summary of the results in terms of p-values using the DAVID GO database. For each combinations of heuristics there is a result for BP, MF, CC.

Algorithm	BP	MF	CC
DNGr	3,3E-3	6,9E-3	1,3E-2
MST+MyGr	2,1E-2	3,1E-2	4,3E-2
MST+DNGr+ILSAy	2,5E-3	3,8E-3	2,4E-3
MST+MyGr+ILSAy	1,9E-2	1,7E-2	2,3E-2
MS+MST+DNGr+ILSAy	1,4E-2	2,5E-2	2,4E-2
MS+MST+MyGr+ILSAy	1,55E-3	1,14E-3	2,7E-2

Table 4.7: Number of genes retrieved with Arabidopsis dataset. These are statistics on the number of genes retrieved on each execution of our best algorithms with Arabidopsis dataset. For description see Table 4.5.

Num of Genes	Num of Biclusters
154	1
172	2
113	3
83	4
39	5
16	6
7	7
3	8
0	9
0	10
0	> 10
433	comparing in more than one biclusters
587	total genes considered

Table 4.10: Statistics on the Arabidopsis dataset. The table shows a summary of the results in terms of p-values using the DAVID GO database. For each combinations of heuristics there is a result for BP, MF, CC.

Algorithm	BP	MF	CC
DNGr	3,1E-3	1,9E-2	2,0E-3
MST+MyGr	3,7E-4	2,5E-2	8,1E-3
MST+DNGr+ILSAy	1,41E-5	2,6E-2	3,8E-4
MST+MyGr+ILSAy	1,4E-6	1,5E-2	1,3E-3
MS+MST+DNGr+ILSAy	4,5E-11	8,3E-3	2,8E-5
MS+MST+MyGr+ILSAy	3,1E-3	2,03E-2	7,9E-3

In Figure 4.4 and Figure 4.5, two different plots can be seen . The first figure corresponds to the test on the Arabidopsis dataset made by the Dharan-Nair algorithm and the second is obtained with our

procedure. It has been evident that with bigger biclusters a large part of the curves is not similar.

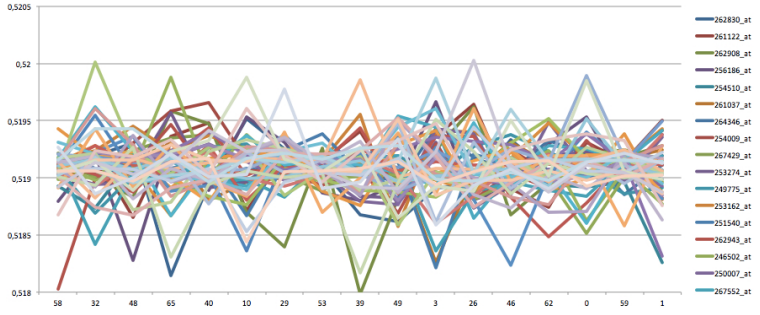

Figure 4.4: Plot of one bicluster output of the Dharan-Nair algorithm on the Arabidopsis dataset. The dataset is Arabidopsis. The curves are colored and each of them represent a gene. In abscissa the number represent a condition. In ordinate the number represent the value of expression found in the matrix $\mathcal{A}$.

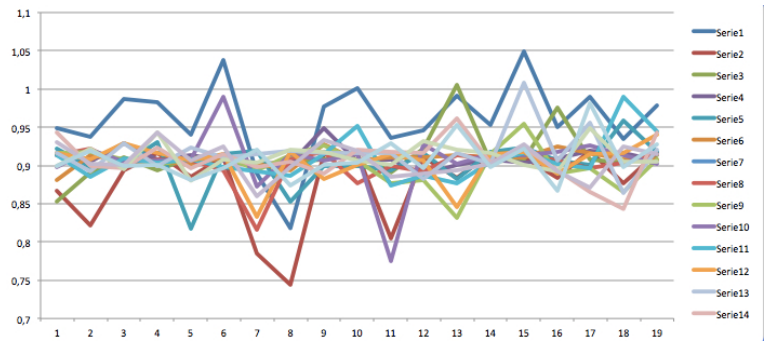

Figure 4.5: Plot of one bicluster output of the designed algorithm. The dataset is Arabidopsis. The curves are colored and each of them represent a gene. In abscissa the number represent a condition. In ordinate the number represent the value of expression found in the matrix $\mathcal{A}$.

Two other plots in Figures 4.6 and 4.7 are the expression of the fact that these algorithms work well. In effect the behaviours of the curves are very similar even if they are built using parameters that permit a large inclusion of elements. The bigger is the bicluster the lower is the similarity among the genes as curves in the plot. When bicluster are very big it can happen that more than one group are present inside it.

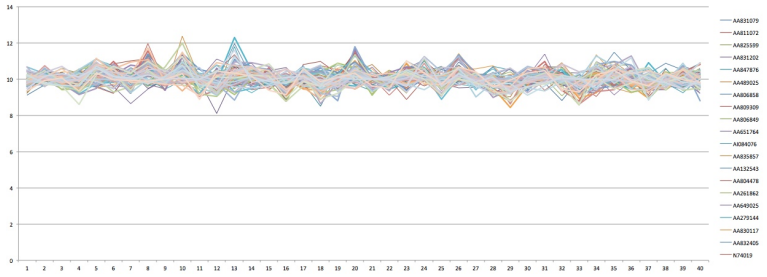

Figure 4.6: Plot of one bicluster with a modified version of Dharan-Nair algorithm. It is the algorithm in [51] with the addition of the ILS procedure and inserting it in a multi-start context. The dataset is Lymphoma. The curves are colored and each of them represent a gene. In abscissa the number represent a condition. In ordinate the number represent the value of expression found in the matrix $\mathcal{A}$.

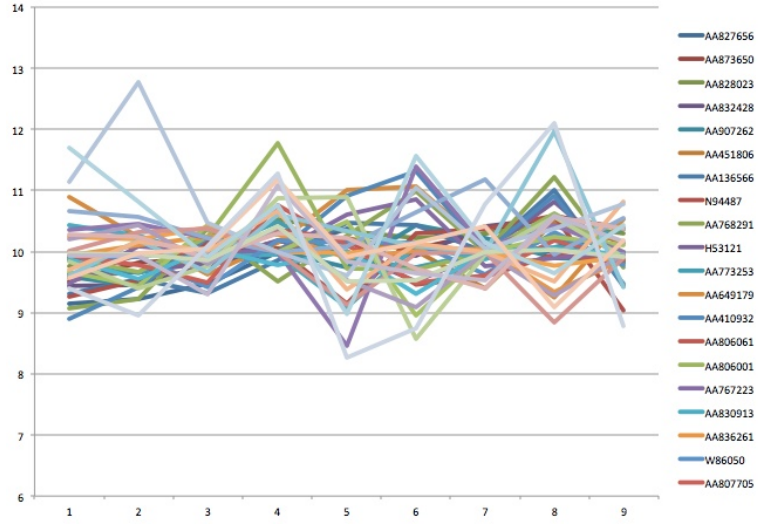

Figure 4.7: Plot of one bicluster using our GRASP-like on the Lymphoma dataset. The dataset is Lymphoma. The curves are colored and each of them represent a gene. In abscissa the number represent a condition. In ordinate the number represents the value of expression found in the matrix $\mathcal{A}$. The behaviour of the curves are similar but not all together. This happens when the threshold is higher and permits the introduction of more genes.

4.5 Comparison with algorithms from literature

In order to compare the results obtained with our approach to those of other approaches already described in literature, we selected three

algorithms, taken from the most popular for biclustering and we applied them on the same datasets used to test the proposed algorithms. They are the Cheng and Church algorithm [10], the ISA algorithm [26], the OPSM algorithm [21]. They all have been already described in Chapter 2. The experiments have been conducted running the software *BicAT* [70] that is implemented in Java and analysing the biological significance of the results using the DAVID GO web service. Results are shown in Table 4.11 and 4.12 together with those of our best version (MST+MyGr+ILSAy). It is possible to note that the performances of our algorithms on Lymphoma and Arabidopsis datasets are very similar to those of some of the most known algorithms for biclustering. Instead, on the Yeast environmental dataset the performances of the three algorithms from the literature are slightly better than those obtainable with ours. This behaviour is due to the fact that the algorithms can perform better on certain types of data and in any case the values shown, as we said before, are strictly correlated with the dimensions of the bicluster obtained. In many cases these algorithms permit to obtain very large clusters. When a set of clusters is given to a GO system like DAVID, it returns a value that, experimentally, is also dependent from the cardinality of that set.

Table 4.11: Results on Lymphoma dataset with DAVID using ISA, CC, OPSM. The table shows a summary of the results in terms of p-values using the DAVID GO database and the algorithms CC, ISA and OPSM. MST+MyGr+ILSAy is our best algorithm. **CC** is intended for Cheng and Church algorithm in [10], **ISA** is the ISA algorithm in [26] and **OPSM** is the one in [21].

Algorithm	BP	MF	CC
MST+MyGr+ILSAy	8,4E-3	2,0E-2	2,0E-2
CC	5,4E-4	2,3E-3	9,3E-3
ISA	2,1E-3	1,4E-2	5,0E-3
OPSM	1,3E-4	3,2E-3	2,6E-3

Table 4.12: Results on Yeast environmental dataset with DAVID using ISA, CC, OPSM. For description see Table 4.11.

Algorithm	BP	MF	CC
MST+MyGr+ILSAy	1,9E-2	1,7E-2	2,3E-2
CC	3,4E-5	5,6E-3	4,7E-3
ISA	8,3E-7	1,2E-4	1,8E-4
OPSM	2,4E-4	4,2E-7	1E-5

Table 4.13: Results on Arabidopsis dataset with DAVID using ISA, CC, OPSM. For description see Table 4.11.

Algorithm	BP	MF	CC
MST+MyGr+ILSAy	1,4E-6	1,5E-2	1,3E-3
CC	2,2E-3	4,0E-2	1,1E-2
ISA	7,9E-4	2,1E-2	9,2E-3
OPSM	6,2E-4	1,5E-2	3,7E-3

So it has been decided to change the parameters of our best version of hybridized algorithm to obtain more objects to be included in the clusters (Table 4.14) to see the significance in this

case. It has been done only for one of the best algorithms implemented (MST+MyGr+ILSAy). Only the biclusters with around one hundred elements have been taken and used with the DAVID system. The results show that the mean of the best p-values are higher than the previous.

Table 4.14: Results on the three datasets with DAVID using our best algorithm. The best algorithm include the iterated local search and the alternate insertion. Average number of genes in the biclusters resulting from the experiments is approximately 110 for Lymphoma and Arabidopsis and 70 for Yeast environmental. The values represent the p-value for respectively biological process, molecular function and cellular component.

Dataset	BP	MF	CC
Lymphoma	2,0E-5	2,0E-3	8,8E-5
Yeast environmental	1,2E-2	1,0E-2	3,8E-2
Arabidopsis	9,7E-17	6,8E-07	8,6E-07

In Figure 4.8 is shown a boxplot of the results of the algorithms. The algorithm created in this work and ISA are the ones that allow to have the more linear distribution of values. This means that our biclusters have biological significances (in terms of p-value) while in other cases, i.e. with the OPSM algorithm, the biclusters found have different p-values. The test is referred to the Arabidopsis dataset and the specific case of biological process results.

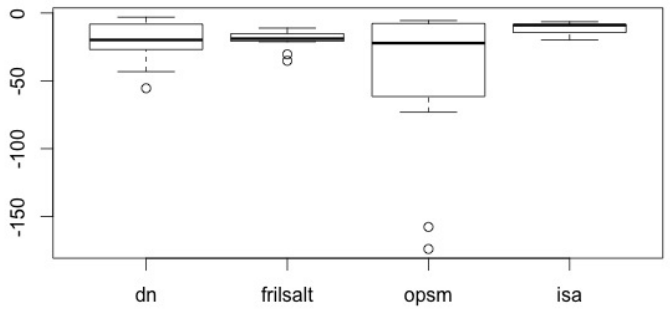

Figure 4.8: Boxplot of the p-values. In order from left are the DN algorithm, the MyGR+ILSAy designed, then the OPSM and ISA algorithm. These plots show what is the distribution of values obtained for the case of Arabidopsis dataset. The case is of one generic bicluster.

Chapter 5

Discussion

The best ideas arrive when you are not thinking about your goal.

The main purpose obtained in this thesis was to overcome some of the limitations of the most common biclustering approaches. Some of them can find only a certain type of bicluster and need the use of many parameters to define the final result.

The work is begun with the assumption that the problem of biclustering of gene expression data is NP-Complete. To solve this problem one can use the appropriate heuristics that allow to get an eligible solution in a reasonable time.

This work started with the analysis of a GRASP-like approach [51] and from the consideration that it implemented the use of all possible solutions as a neighborhood of the current solution. This approach appears to be not ideal, since it obtains all possible neighbors and so needs to enter any missing element in turn and calculate the score of the bicluster obtained. This takes away a significant amount of time even if, searching for several solutions, it leads to good results.

In the new GRASP-like algorithm, the first thing that has been attempted to improve was the local search. The algorithm takes

inspiration from a social dynamic: a group of friends can be created for bonds due to common choices (e.g. school or social events of any type) and it can grow up and become a solid group. As time goes, people leave the group or they are always be added on the basis of a similarity of interests. If the individual is not compatible with the group it will come out easily. Otherwise it enters in the group and it becomes a constituent part of the group

Then, the insertion of a procedure for the intensification of local search, the iterated local search, allowed to move in the space of solutions looking for the global minimum.

Finally the whole procedure was repeated several times to obtain a multi start logic: starting from multiple points of the space of solutions different results have been obtained and the best one is kept at the end of the procedure.

We defined as "best algorithm" a version of our algorithm that has the lower p-values in most of the experiments and it does not need a long execution time. It is MST+MyGr+ILSAy. It is very similar in results to the version with the multi-start procedure (MS+ MST+MyGr+ILSAy) but the first one has been chosen because of the long execution time that the multi-start procedure needs.

Another issue is represented by the choice of measures of similarity and distance. Distance cosine measure has been used because, empirically, it seems to be the one capable of giving a meaning to the proximity of two vectors that can be equal only for some of their features. However, the measure of Cheng and Church (MSR) still seems to be the one that best quantifies the goodness of the biclusters.

The comparison of the GRASP-like procedure previously published [51] with the one developed in this work shows that:

1. the Dharan-Nair algorithm is slower than ours because the local search implemented is very different. In the first case the search for a better candidate is among all the genes and not only among a subgroup. As a consequence, this computation is considered very expensive. In contrast our algorithm is faster because it chooses as a candidate the random extracted from the RCL and executes some simple operations only inside the bicluster or randomly on all the set;

2. Dharan-Nair algorithm does not have an erasing step and so the presence of outliers is more probable.

Our new GRASP-like seems very similar, in terms of biological results, to the one in Dharan-Nair that in a few cases is better than ours. This can be seen in Table 4.8 where DNGr has values better than the MS+MyGr algorithm or in the case of the Yeast environmental in Table 4.9. However, our local search is faster. We seen that to create a bicluster our algorithm requires few minutes (8-15) while the algorithm of departure may take among 10 and 45 minutes. A computational complexity depending from the number of genes in the dataset has been improved with a complexity depending on the number of genes in the bicluster only.

Moreover both the iterated local search and the use of a multi-start approach can improve the results for all the algorithms. In six cases out of ten an improvement of this type can give better results to the algorithm DNGr this is shown comparing the results in the Tables 4.8, 4.9 and 4.10. The fact that the MST+DNGr+ILSAy or MS+DNGr+ILSAy have a lower p-value in more than 70% of cases it suggests that is better to use these type of algorithms. However, in the case of a multi-start procedure, it must be remembered that the computational cost is increased multiplying the normal cost for the number of times one is deciding to restart the overall algorithm.

Another thing that has been noted is that the greater are the biclusters and higher their biological significance. But in any case this is not a rule. The problem is that when clusters are very big they contain more than one group of elements with the same properties, and this can be misleading during a bioinformatic analysis. This is the reason why in the experiments has been chosen to keep the biclusters with limited dimensions.

Methods to avoid building overlapping biclusters have not been explored in this thesis.

With this work has been assessed that, even if a meta-heuristics have not been so largely used for the biclustering problem it can be a good starting point. Several combinations of heuristics can still be tested, and those used in this work are sufficient to establish that GRASP-like is a good approach to solve the problem of biclustering where there is not *a priori* knowledge of the elements to be treated.

Some other useful steps to improve the clustering algorithm based on the social dynamic can also been tested and found.

Appendix A

Creation of the statistics

The tool *Clone/GeneId Converter* provided in [71] has been used to convert gene names (for example, to convert an Accession Number in a Unigene identifier). Thus has been filled the gap of the absence of gene identifiers in the GO systems. The id-converter website [72] provides a form where one can choose the organism, the type of identifier and then convert to a specific identifier. A set of genes can be inserted and will be returned in an output file with all the names that the tool is able to convert.

Once identifiers are converted, the sets of genes may be transmitted to the GO system. For example DAVID has a function similar to that of the converter described. Steps are the following: create and upload a list of genes, choose an identifier type and submit the list. Most of the times the system can recognize the gene names but in some cases it will need to convert them.

A set of results for the biological process, molecular function and cellular components will be given in output for each of the clusters of genes given in input. From this results one can take the p-value assigned to each of the function, component, process by the system and that has been written in a table. This table can be also downloaded to use it later. Only the first p-value (the best one) has been taken. It has been inserted in a table that for each bicluster has

a row and for each of the results (BP, MF, CC) a column. Fifteen biclusters has been collected for each algorithm run. The mean of the fifteen has been taken and inserted in a final results table. This table has been used to assess the validity of the algorithms.

Acknowledgments

Scarcity in employment is the bane of our society. People are out of work and do not have a future employment opportunities. I do not know exactly what made me reject a sure job to follow just a passion. Maybe, because it is true that Love and passion for work are the fuel on which the engine of life chugs. Without them, life cannot continue. Now I know "what it takes" to keep the flame alive in the self. And I know that this all is not right.

Some time ago I decided to leave my studies but now here I am and I owe it to my tutors, Paola Festa and Anna Marabotti. They encouraged me "to do it" in this journey of mine along with Angelo Facchiano. I thank prof. Petrosino who accompanied me in the university path and allowed me to chalk my own work strategy. Letting me go, perhaps at the right time. I owe it to Remo Sanges who gave me an opportunity.

I owe it to Arianna who has endured my uncertainties and my discomfort with a soul as big as a universe. I owe it to my great new colleagues and my bioinformatics lab! I owe it to the b.b.m.: my strength, my certainty. I owe it to my brothers and my parents. And I owe it to the strength of my mother.

Bibliography

[1] P. Flicek, M. R. Amode, D. Barrell, K. Beal, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gil, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, M. Komorowska, G. Koscielny, E. Kulesha, P. Larsson, I. Longden, W. McLaren, M. Muffato, B. Overduin, M. Pignatelli, B. Pritchard, H. S. Riat, G. R. S. Ritchie, M. Ruffier, M. Schuster, D. Sobral, Y. A. Tang, K. Taylor, S. Trevanion, J. Vandrovcova, S. White, M. Wilson, S. P. Wilder, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernandez-Suarez, J. Harrow, J. Herrero, T. J. P. Hubbard, A. Parker, G. Proctor, G. Spudich, J. Vogel, A. Yates, A. Zadissa, S. M. J. Searle,"Ensembl 2012", *Nucleic Acids Res.*, vol. 40 pp. D84-D90, 2012.

[2] J. C. Alwine, D. J. Kemp, G. R. Stark, "Method for detection of specific RNAs in agarose gels by transfer to diazobenzyloxymethyl-paper and hybridization with DNA probes", *Proc. Natl. Acad. Sci. U.S.A.*, vol. 74(12), pp. 5350,Äì4, 1977.

[3] R. P. Ekins, *UK Patent Application*, 8 803 000, 1987.

[4] M. R. El-Gewely, "Biotechnology Annual Review" *Elsevier*, 2003.

[5] D D. Dalma-Weiszhausz, J. Warrington, E. Y. Tanimoto, C. G. Miyada "The affymetrix GeneChip platform: an overview", *Methods Enzymol*, vol. 410, pp. 3-28, 2006.

[6] D. A. Lashkari, J. L. Derisi, J. H. McCusker, A Len F. Namath, C. Gentile, S. Y. Hwang, P. O. Brown, R. W. Davis. "Yeast microarrays for genome wide parallel genetic and gene expression analysis", *Proc. Natl. Acad. Sci. USA* , vol. 94, pp. 13057-13062, 1997.

[7] T. Barrett, D. B. Troup, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, R. N. Muertter, M. Holko, O. Ayanbule, A. Yefanov, A. Soboleva, "NCBI GEO: archive for functional genomics data sets 10 years on", *Nucleic Acids Res.*, vol. 39, pp. D1005-D1010, 2011.

[8] L. Zheng, X. He, "Classification Techniques in Pattern Recognition", *Conference proceedings WSCG 2005* , 2005.

[9] R. O. Duda, P. E. Hart, D.G. Stork, "Pattern classification". *Wiley*, 2001.

[10] Y. Cheng, G. M. Church, "Biclustering of expression data". In R. Altman, T. L. Bailey, P. Bourne, M. Gribskov, T. Lengauer, I.N. Shindyalov (eds.) *Proc. of the 8th Int. Conf. on Int. Sys. for Mol. (ISMB 2000)*, pp. 93-103. AAAI Press, Menlo Park, 2000.

[11] Mirkin, Boris, "Mathematical Classification and Clustering", *Kluwer Academic Publishers*, 1996.

[12] J. A. Hartigan, "Direct clustering of a data matrix", *J. Am. Stat. Assoc.*, vol. 67, pp. 123-127, 1972.

[13] E. Brown, "Mining Truth Table and Straddling Biclusters in Binary Datasets ", *Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University*, 2009.

[14] D. I. Ignatov, S. O. Kuznetsov, J. Poelmans, "Concept-Based Biclustering for Internet Advertisement", *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pp. 123-130, 2012.

[15] S. C. Madeira, A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey", *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 1, pp. 24-45, 2004.

[16] R. Peeters, "The Maximum Edge Biclique Problem is NPComplete", *Discrete Appl. Math.*, vol. 131, no. 3, pp. 651-654, 2003;

[17] G. Getz, E. Levine, E. Domany *Coupled two-way clustering analysis of gene microarray data*, *Proc. Natl. Acad. Sci. USA*, 2000.

[18] C. Tang, L. Zhang, I. Zhang, M. Ramanathan, "Interrelated Two-Way Clustering: An Unsupervised Approach for Gene Expression Data Analysis", *Proc. Second IEEE Int. Symp. Bioinformatics and Bioeng.*, pp. 41-48, 2001.

[19] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, "A systematic comparison and evaluation of biclustering methods for gene expression data", *Bioinformatics*, vol.22(9), 2006.

[20] J. Yang, H. Wang, W. Wang, P.S. Yu, "An improved biclustering method for analyzing gene expression profiles", *Int. J. Artif. Intell. T.*, vol. 14,pp. 77-790, 2005.

[21] A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving submatrix problem", *J. Comput. Biol.*,vol. 10, pp.373-84, 2003.

[22] T. M. Murali, S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data", *Pac. Symp. Biocomput.*,vol. 8, pp. 77-88, 2003.

[23] A. Tanay, R. Sharan, R. Shamir. "Discovering statistically significant biclusters in gene expression data", *Bioinformatics*, vol. 18, pp. S136-S144, 2002.

[24] L. Lazzeroni, A. Owen. "Plaid models for gene expression data", *Stat. Sin.*, vol 12, pp. 61-86, 2002.

[25] M. Lashkargir, S. A. Monadjemi, A. B. Dastjerdi, "A Hybrid Multi-Objective Particle Swarm Optimization Method to Discover Biclusters in Microarray Data", *International Journal of Computer Science and Information Security*, vol. 4, no. 1,2, 2009.

[26] J. Ihmels, A. Bergmann, N. Barkai, "Defining Transcription Modules Using Large-Scale Gene Expression Data". *Bioinformatics*, vol. 20, pp. 1993-2003, 2004.

[27] S. C. Madeira, A. L. Oliveira, "A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series", *Algorithms. Mol. Biol.*, vol. 10, pp.1748-7188, 2009.

[28] D. J. Reiss, N. S. Baliga, R. E. Bonneau, "Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks". *BMC Bioinformatics*, vol. 2, pp. 280-172, 2006.

[29] S. Hochreiter, U. Bodenhofer, H. M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, L. Bijnens, H. W. H. Gohlmann, Z. Shkedy, DA. Clevert, "FABIA: factor analysis for bicluster acquisition", *Bioinformatics*, vol. 26, 2010.

[30] G. Li, Q. Ma, H. Tang, A. H. Paterson, Y. Xu, "QUBIC: a qualitative biclustering algorithm for analyses of gene expression data", *Nucleic Acids Res.* , vol. 37(15), 2009.

[31] C. Xie, G. Liu, C. Xing, M. Wang, Y. Zhou, "A novel biclustering with parallel genetic algorithm". *Coll. of Comput. Sci. & Technol*, pp. 900-903, 2011.

[32] A. Chakraborty, "Biclustering of Gene Expression Data Using Genetic Algorithm", *Comp. Int. in Bioinf. Comp. Bio., 2005. CIBCB 05. Proceedings of the 2005 IEEE Symposium on*, pp. 1-8 , 2005.

[33] F. Divina, J. Aguilar-Ruiz, "Biclustering of Expression Data with Evolutionary Computation", *IEEE Trans. Knowl. Data Eng.*, vol. 18, pp. 590-602, 2006.

[34] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol.220, n. 4598, pp. 671-680, 1983.

[35] K. Bryan, P. Cunningham, N. Bolshakova, "Application of Simulated Annealing to the Biclustering of Gene Expression Data", *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 3, pp. 519-525, 2006.

[36] A. Chakraborty, "Biclustering of Gene Expression Data with simulated annealing", *High-Performance Computing in Asia-*

Pacific Region. Proceedings. Eighth International Conference on, pp. 632-638 , 2005.

[37] J. Y. Xu, S. Y. Chiu, "Effective heuristic procedure for a field technician scheduling problem", *J. Heuristics*, vol. 7, pp. 495-479, 2011.

[38] A. Srinivasan, K. G. Ramakrishnan, K. Kumaram, M. Aravamudam, S. Naqvi, "Optimal design of signaling networks for Internet telephony", *IEEE Infocom. Ser.* , vol. 2, pp. 707-776, 2000.

[39] M. Armony, J. G. Klincewicz, H. Luss, M. B. Rosenwein, "Design of stacked self-healing rings using a genetic algorithm *J. Heuristics* , vol. 6, pp. 85-105, 2000.

[40] S. Binato, G. C. Oliveira, "A reactive GRASP for transmission network expansion planning", *Oper. Res. Comp. Sci.*, Kluwer Academic Publishers, pp. 81-170, 2002.

[41] H. Jr. Faria, S. Binato, M. G. C. Resende, D. J. Falcao, "Power transmission network design by a greedy randomized adaptive path relinking approach", *IEEE T. Power Syst.*, vol. 200, pp. 43-47, 2005.

[42] A. A. Andreatta, C. C. Ribeiro, "Heuristics for the phylogeny problem" *J. Heuristics* , vol. 8, pp. 429-447,2002.

[43] D. G. Brown, T. J. Vision, S. D. Tanksley, "Selecting mapping: a discrete optimization approach to select a population subset for use in a high-density genetic mapping project", *Genetics*, vol. 155, pp. 407-420, 2000.

[44] T. A. Feo, M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem", *Oper. Res. Lett.*, vol.8, pp. 67-71, 1989.

[45] T. A. Feo, M. G. C. Resende, "Greedy randomized adaptive search procedures". *J. Global Optim.*, vol. 6, pp. 109-133, 1995.

[46] P. Festa, M. G. C. Resende, "GRASP: An annotated bibliography". *Operat. Res. Comp. Sci.*, C. C. Ribeiro, P. Hansen (eds.), Kluwer Academic Publishers, pp. 325-367, 2002.

[47] P. Festa, M. G. C. Resende, "An annotated bibliography of GRASP Part I: Algorithms", *Intl. Trans. Op. Res.*, vol. 16, pp. 1-24, 2009.

[48] P. Festa, M. G. C. Resende, "An annotated bibliography of GRASP Part II: Applications", *Intl. Trans. Op. Res.*, vol. 16, pp. 131-172, 2009.

[49] M. Prais, C. C. Ribeiro, "Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment", *Informs. J. Comp.*, vol. 12, pp. 164-176, 2000.

[50] H. Delmaire, J. A. Diaz, E. Fernandez, and M. Ortega. "Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem", *INFOR*, vol. 37, pp. 194-225, 1999.

[51] S. Dharan, A. S. Nair, "Biclustering of gene expression data using reactive greedy randomized adaptive search procedure", *Bioinformatics*, vol. 10, pp. S27, 2009.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to algorithms". *MIT Press and McGraw-Hill*, 1990.

[53] R. C. Prim, "Shortest connection networks and some generalizations". *Bell Syst. Tech. J.*, vol. 36, pp. 1389-1401, 1957.

[54] S. P. Lloyd, "Least square quantization in PCM", *IEEE Trans. Inf. Theory* vol. 28(2), pp. 129-177, 1957.

[55] C. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters ",*IEEE Trans. Comp.*, vol. C-20, pp. 68-86, 1971.

[56] W. Ayadi, M. Elloumi, J. K. Hao, "Iterated Local Search for Biclustering of Microarray Data", *Lect. Notes Comp. Sc.*, vol. 6282, pp. 219-229, 2010.

[57] C. G. E. Boender, A. H. G. Rinnooy Kan, L. Strougie, G. T. Timmer, "A stochastic method for global optimization" *Math. Program.* , vol. 22, pp. 125-140.

[58] X. Jiao, B. T. Sherman, D. W. Huang, R. Stephens, M. W. Baseler, H. C. Lane, R. A. Lempicki, "DAVID-WS: a stateful web service to facilitate gene/protein list analysis", *Bioinformatics*, vol. 28, pp. 1805-C1806, 2012.

[59] P. D. Thomas, M. J. Campbell, A. Kejariwal, H. Mi, B. Karlak, R. Daverman, K. Diemer, A. Muruganujan, A. Narechania, "PANTHER: A Library of Protein Families and Subfamilies Indexed by Function", *Genome Res.*, vol. 13, pp. 2129-2141, 2003.

[60] `http://arep.med.harvard.edu/biclustering/`

[61] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, G. M. Church, "Systematic determination of genetic network architecture",*Nat. Genet.*, vol. 22, pp. 281-285, 1999.

[62] A. Wille, P. Zimmermann, E. Vranovic, A. Feprholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, P. Barhlmann, "Sparse graphical Gaussian modeling of the isoprenoid gene network in Arabidopsis thaliana ", *Genome Biol.*, vol. 5(11), 2004.

[63] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, P. O. Brown, "Genomic expression programs in the response of yeast cells to environmental changes", *Mol. Biol. Cell*, vol. 11, pp. 4241-4257, 2000.

[64] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewi, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, L. M. Staudt, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling", *Nature*, vol. 304, pp. 503-511, 2000.

[65] `http://www.geneontology.org/GO.doc.shtml`

[66] J. M. Cherry, E. L. Hong, C. Amundsen, R. Balakrishnan, G. Binkley, E. T. Chan, K. R. Christie, M. C. Costanzo, S. S. Dwight, S. R. Engel, D. G. Fisk, J. E. Hirschman, B. C. Hitz, K. Karra, C. J. Krieger, S. R. Miyasato, R. S. Nash, J. Park, M. S. Skrzypek, M. Simison, S. Weng, E. D. Wong, "Saccharomyces Genome Database: the genomics resource of budding yeast",*Nucleic Acid. Res.*, vol. 40 , pp. D700-D705, 2012.

[67] `http://www.yeastgenome.org/cgi-bin/GO/goTermFinder`

[68] `http://www.pantherdb.org/`

[69] `http://david.abcc.ncifcrf.gov/`

[70] S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, E. Zitzler, "BicAT: a biclustering analysis toolbox", *Lect. Notes Comput. Sc.*, vol. 22 (10), pp. 1282-1283, 2006.

[71] A. Alibes, P. Yankilevich, A. Canada, R. Dizaz-Uriarte "IDconverter and IDClight: Conversion and annotation of gene and protein IDs", *BMC Bioinformatics*, pp. 8-9,2007.

[72] `http://idconverter.bioinfo.cnio.es/`