PhD Dissertation

Dottorato in Scienze Computazionali e Informatiche

Università degli Studi di Napoli
Federico II

MULTI-AGENT AUTOMATED NEGOTIATION FOR MANAGEMENT OF SERVICES IN SMART CITIES

Dario Di Nocera

Tutor:

Prof. Silvia Rossi

Università degli Studi di Napoli Federico II

Advisor:

Prof. Massimo Benerecetti

Università degli Studi di Napoli Federico II

February 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The objective of this PhD thesis is the study, the design and the prototype implementation of an agent-based automated negotiation mechanism to support the development of smart Service Based Applications having as a reference scenario the smart cities of the future intended as a realistic market of services. The research activity has been carried out in the contest of two research projects: $OR.C.HE.S.T.R.A.$ and $S^2 - Move$.

1.1 The Context of the work

According to the latest directives of the European Commission, several initiatives have started to support research projects for the Smart Cities and Social Innovation, having as main objective the development of software applications, the re–qualification and the technological innovation of urban areas relying on the modern Information and Communication Technology, now spread everywhere. Nowadays, there are more than four billion mobile devices forming a wireless network of sensors, thirty billion of RFID, ten billion devices equipped with chips and sensors able to exchange a paramount amount of information in real time. All this information can be processed and aggregated in different ways, so to provide added value applications exposing functionalities hardly delivered by monolithic applications that can be easily consumed by users satisfying their requests. The research activity has been carried out within the scope of two research projects (P.O.N.) funded by the MIUR, OR.C.HE.S.T.R.A. and S^2-Move, whose common objective is to increase the diffusion and the utilization of the modern technologies of information and communication in urban areas, in order to provide Service Based Applications (SBA)

aimed not only to improve the liveable level of such areas, but also to be able to provide such applications in an "intelligent manner", so to satisfy users' requests, also with respect to the requested quality of the provision, known as the Quality of Service (QoS) of the application. In order to reach this objective, it is necessary to adopt software technologies able to address the dynamic nature of the provision of these applications, in a realistic market of service, that characterize the Smart Cities reference scenario of this work. Of course, the work will only focus on ad hoc designed SBAs, even though the applications domains for SBAs that can be provided to the public range from the health domain, to the fruition of Cultural Heritage, urban mobility, just to report only a subset of the ones more relevant and addressed by the research community.

1.2 The addressed problem

An efficient use of the ICT nowadays available, allows to develop Service Based Applications designed to manage, analyse, and spread the enormous amount of data available on the Web, through the innovative technological infrastructures available. These applications process big volumes of heterogeneous data gathered from different types of available sources in real time such as: sensors, tablets, smartphones, control units.

In the context of Smart Cities, it is crucial to be able to integrate different platforms and different applications available on the Internet to develop services useful for the public, for the tourism, for the Public Administration to manage the needs of users communities (e.g. mobility, security, and so on). It is well known that these types of applications are composed of different and autonomous services, so providing added value applications that single monolithic solutions may be unable to provide. In the present work it is assumed that distributed services are developed according to well–established standards [68] that guarantee their interoperability, a fundamental request for composing SBAs. The evolution of the Internet towards the Internet of the Future, is leading to the transition from a document-centric approach to a service-centric approach, requiring that services are characterized not only by the functionality they provide, but also by the quality of the provision in terms of performance, cost, reliability, and so on. These parameters are usually referred as Quality of Service parameters, and their values may differ from one service to another, even if they provide the same functionality. So, once interoperability and communication between autonomous and independent services is pos-

sible, it is necessary to develop methodologies that allow for the automatic composition of services by taking into account the variability and dynamism of events occurring in a realistic market of services such as the one offered by the Smart Cities. In this work, it is proposed to extend the representation of a service, so allowing for the implementation of a "smart behaviour" able to cope with the dynamic provision of services according to users requests characterized also by a required QoS level of satisfaction. The problem addressed in this work is to design a middleware software mechanism that can be integrated in the development of SBAs to automatically compose services with a goal:

1. to provide SBAs satisfying users requests both from a functional and non-functional (QoS) point of view

2. to allow for the provision of a service also when there are QoS requirements conflicting with the QoS provided.

1.3 Our Solution

The Service Oriented Computing (SOC) approach to programming allows for the development of SBAs starting from the composition of network-available services rather than building new applications from scratch to accomplish a task. A service is a programmatically available application logic exposed over the Internet [53] that is not subject to a centralized control, and it is independently developed from the others. The approach proposed in this work is to model a Smart City as a market where providers and consumers interact to reach their own objectives: the first ones to provide their services in order to get a profit (in the most general sense), and the second ones to obtain a composition of such services delivering the application, sometimes complex, they require that satisfies also their requirements in terms of global QoS (end-to-end requirements). The characterizing features of multi–agent systems make them a suitable approach to model SBAs providers and consumers, so allowing the simulation of their interactions through the use of advanced methodologies to automate the composition process of an application that should satisfy specific QoS requirements. Among these methodologies, automated software agent negotiation is widely recognized as a suitable mechanism to implement market-based interactions such as the ones considered in this work. The work carried out confirms that negotiation can be adopted as a useful mechanism to help finding a

solution both for the problem of the dynamic composition of QoS–aware SBAs, and for the problem of accommodating conflicting interests w.r.t. the considered QoS parameters when providing services for the mobility in urban areas (parking solutions). For the first problem, automated an automated negotiation mechanism is proposed as the possibility to find, if successful, a selection of services that once aggregated provide the required application with the required (or at least acceptable) QoS values, assuming that in a realistic market of services, it is likely that more services providing the same functionality, but with different QoS values may be available. Agent bidding strategies that model the providers behaviour when providing their services have been designed and their adoption in the context of service composition is simulated and analyses. In addition, the mapping of these strategies when dealing with single-issue, or multi–issues negotiation is investigated together with the possibility to find near optimal agreements also when dealing with service composition. For the second problem, it is shown how automated negotiation can be used to model a Decision Support System helping users to select a solution to their problem also when their QoS requirements are in conflict with the ones offered by the providers of the composed service they require. The application domain chosen for the development of the mechanism is the Smart Parking for a Smart City.

1.4 Innovative Aspects

The proposed bidding strategies for an automated negotiation process based on a Gaussian distribution function represent an original contribution of this work. It is shown how the adoption of such strategies can be easily used to model different negotiation types with the same negotiation framework in terms of protocols and strategies. Furthermore, the approach is distributed since negotiation occurring with different providers can be carried out concurrently, so addressing the problem of the negotiation cost crucial for SBAs. The adopted negotiation protocol allows to address many of the requirements of negotiation for service composition. An evaluation of the negotiation trends in different market configurations, is reported to show the pros and cons of its use. In addition, the same negotiation mechanism equipped with a different set of negotiation strategies is proposed to support drivers to select a parking space in the city of Naples, according to both their requirements in term of cost and location, but also the requirements of parking vendors and city regulation needs (sometimes unforeseen and conflicting with the drivers

ones). This approach results an innovative use of negotiation to model a support decision system. A prototype of this Smart Parking system has been implemented and shown as a demo at the "PAAMS 2014" international conference.

1.5 List of Publication

- D. Di Nocera, C. Di Napoli and S. Rossi, "Evaluating the Social Benefit of a Negotiation-based Parking Allocation", Proceedings of the 13th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2015), June 3-5 2015, to appear

- C. Di Napoli, D. Di Nocera and S. Rossi, "Computing Pareto Optimal Agreements in Multi-issue Negotiation for Service Composition (Extended Abstract)", Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul (Turkey), May 4-8 2015, to appear.

- S. Rossi, D. Di Nocera and C. Di Napoli, "Gaussian-based Bidding Strategies for Service Composition Simulations", Recent Advances in Agent-based Complex Automated Negotiation, Studies in Computational Intelligence, Springer 2015, to appear.

- C. Di Napoli, D. Di Nocera, and S. Rossi, "Analyzing Negotiation Trends in a QoS-aware Market of Services", SCS M&S Magazine, Vol. IV. Issue 3, December 2014.

- C. Di Napoli, D. Di Nocera, P. Pisa and S. Rossi, "A market-based coordinated negotiation for QoS-aware service selection", In Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets, Lecture Notes in Business Information Processing series 187, Springer International Publishing, pp. 26-40, 2014.

- C. Di Napoli, D. Di Nocera and S. Rossi, "Agent negotiation for different needs in smart parking allocation", in Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, Lecture Notes in Computer Science 8473, Springer International Publishing 2014, pp.98-109.

- C. Di Napoli, D. Di Nocera and S. Rossi, "Using Negotiation for Parking Selection in Smart Cities", in Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, Lecture Notes in Computer Science 8473, Springer International Publishing 2014, pp.331-334.

- S. Rossi, D. Di Nocera and C. Di Napoli, "Normal Distributions and Multi-issue Negotiation for Service Composition", in Trends in Practical Applications of Heterogeneous Multi-Agent Systems, The PAAMS Collection. Advances in Intelligent Systems and Computing 293, Springer International Publishing 2014, pp.1-8.

- D. Di Nocera, C. Di Napoli, and S. Rossi, "A Social-Aware Smart Parking Application", Proceedings of the 15th Workshop "From Objects to Agents", Catania (Italy), September 2014, CEUR workshop proceedings, vol 1260 .

- C. Di Napoli, D. Di Nocera and S. Rossi, "Negotiating Parking Spaces in Smart Cities", in Proceeding of the 8th International Workshop on Agents in Traffic and Transportation, in conjunction with AAMAS, Paris, France, May 6, 2014.

- S. Rossi, D. Di Nocera and C. Di Napoli, "An Orchestrated Multi-issue Negotiation with Normal Distributions for Service Selection", in Proceeding of the Seventh International Workshop on Agent-based Complex Automated Negotiations, in conjunction with AAMAS, Paris, France, May 6, 2014.

- C. Di Napoli, D. Di Nocera and S. Rossi, "Evaluating Negotiation Cost for QoS-aware Service Composition", Proceedings of the 14th Workshop "From Objects to Agents" co-located with the 13th Conference of the Italian Association for Artificial Intelligence (AI*IA 2013), Turin (Italy), December 2013, CEUR workshop proceedings, vol 1099, pp. 54-59.

Chapter 2

Basic Concepts

This chapter provides an overview on the technologies used in this PhD is based. In section 2.1 we introduce the Service Oriented Computing (SOC) paradigm, and in section 2.2 a brief introduction to of Multi–Agent Systems. Finally, we introduce concepts related to Automated Negotiation mechanisms.

2.1 Service Oriented Computing

Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions. Services are self describing, platform-agnostic computational elements that support rapid, low-cost composition of distributed applications.[52]

The paradigm of Service Oriented Computing is based on the abstract concept of service. A service is any software entity self contained with a specific functionality, able to communicate through well-defined protocols that allow interoperability between different software systems. In such way, service's functionalities may be offered by service providers to different systems over internet. For this reason, to satisfy these requirements services should be the following features described in [52]:

- *Technology neutral*: they must be invocable through standardized lowest common denominator technologies that are available to almost all IT environments. This implies that the invocation mechanisms (protocols, descriptions and discovery mechanisms) should comply with widely accepted standards.

- *Loosely coupled*: they must not require knowledge or any internal structures or conventions (context) at the client or service side.

- *Support location transparency*: services should have their definitions and location information stored in a repository and be accessible by a variety of clients that can locate and invoke the services irrespective of their location.

A service can be simple and composed of a collection of exiting services that can communicate with each other. Communications may take place both in the data transmission and in the coordination of the activities of two or more services. Service Based Applications are developed composing services, allowing the definition of complex business processes. The services used to build SBAs are offered by providers that implement and provide their maintenance.

Hence, Service Oriented Computing approach (SOC) allows the development of SBA, obtained by composition of the network-available services, encouraging reuse of existing services rather than developing new applications from scratch to accomplish a complex task.

Normally, these services are provided by different suppliers, and for this reason there is the need to define standard protocols widely shared to ensure their interoperability. In this context, the industry is very interested in the relevance and benefits of such technology. Hence, they defined protocols for describing the ways on which services can interact [14]. Consortia promoting these standards protocols are mainly two W3C (World Wide Web Consortium) and OASIS (Organization for the Advancement of Structured Information Standards).

BPEL (Business Process Execution Language) is a language that describes business processes formally through the activities, so it identifies actors and the tasks for which they are responsible. BPEL allows to specify the set of actions within sequential structures, conditionals, loops, and parallel. BPEL is a standard OASIS [50].

WSDL (Web Services Description Language) language is used to describe the high-level functionality of the service, in the standard XML. It is used within the SOA, with the communication protocol SOAP (Simple Object Access Protocol), which deals with the exchange of messages without giving any information about the type of messages exchanged. The task of the WSDL is to describe the type of messages

exchanged by the web service, the access point and the interface with which the service can exchange messages. WSDL is a standard W3C [66].

SOAP/RESTful There are two main ways for developing services: the traditional SOAP Web service and conceptually simpler, RESTful Web services. SOAP Web service is based on three important standardization initiatives: WSDL document that describes the protocol bindings and message formats; SOAP is a protocol for exchanging structured information in XML format; UDDI a register to locate web services. RESTful Web services use the REST model, in which services are identified by URIs, which offer a global addressing space for resource and service discovery. RESTful Web services interact through a uniform interface, which comprises a fixed set of operations in the context of the Web and the Hypertext Transfer Protocol (HTTP): GET, PUT, DELETE and POST [61].

2.1.1 Service Based Applications

The creation process of new services, as described, that leverage the reuse of existing services, is a more complex procedure with respect to a simple correlation between selected web services. In fact, the development of web services providing an add value has to follow three processes: composition, discovery and selection.

Composition is a process by which it is defined, in an abstract description, the dependencies of activities, which together reach a certain business goal. An activity is a functionality that contributes to the achievement of the overall objective. The dependences among activities are defined specifying the structure of the workflow of services is based on some fundamental constructs that determine the relationships between activities, such as:

sequential: activities are carried out one after the other, and an activity is only activated at the end of a previous;

parallel (AND split): after the end of an activity, multiple tasks in parallel are launched;

conditional (OR split): at the end of an activity, one of the other possible activity is initiated, specifying the start up conditions for the activities or the choice could be made by a non deterministic process;

join (AND/OR): to continue in the process instantiation of the workflow, all previous activity (case AND), or at least one of the previous activities (OR) must be completed;

loop: activities are iterated in their execution by specifying the conditions of termination of the iteration.

2.1.2 Quality of Services

In many cases, it is useful to have qualitative information of the selected services. In fact, it may happen that equivalent functionality are provided with different values of not functional parameters, i.e. parameters that are related to how a service is offered and not to the functionality that it provides.

This kind of information is the *quality of service (QoS)* and belongs to the category of non-functional characteristics of the service [44, 37, 45].

Moreover, a user can request that a service is composed with specific values of not functional parameters. In this hypothesis, the service compositor has to make its own choices by analysing the parameter not functional values of services, selecting one for each activity. This is a complex task for a service compositor because it cannot simply choose a generic web service compatible for each abstract service, but it has to evaluate the requests and market supply.

The concept of quality of service depend on the context in which it is used because for the estimation of the quality, in some contexts, some parameters may be more influential than others. In general, the parameters used for the evaluation of the quality of a service are the following:

- *cost* - the amount that a service requester needs to pay to execute the service;
- *time* - the execution time between the requests sent and results received;
- *reliability* - the ability of a service to function correctly and consistently;
- *availability* - the probability that the service is ready to be invoked;
- *performance* - related to the service response time and latency;
- *security* - related to confidentiality and access control aspects.

The concept of quality of service can be extended to the composed application that is the result of a service composition. In the literature, this value is referred as the global or total QoS. The value of a QoS global parameter is related to the value of the QoS parameters of the invoked individual services, when an operation is performed by the application. This means that the function for the calculation of this value is influenced by the following factors:

1. the nature of the parameter taken into account and the metric used to measure the value;

2. the internal structure of the application, i.e., how the services are composed.

In fact, let us consider two applications composed with the same services, but in a different way, where: the first application invokes two services in a sequential way, whereas the second invokes them in parallel. In this scenario, the cost for the transaction is the same for both applications, and is equal to the sum of the costs of individual services. The time, however, is different for each application. In the first case time is equal to the sum of the response time of the individual service. In the second case, instead, time is equal to the maximum response time of the two services because they are invoked independently of each other.

The level of abstraction of Service Oriented Computing paradigm, and the independence from service providers, make it plausible to imagine scenarios in which there are more services available with the same functionality. This could happen either with different suppliers providing equivalent services, or with the same provider offering the same service with different levels and quality parameters.

A scenario like this opens for the possibility to specify constraints and/or simple preferences for the parameters of quality of the application or activity, thus, requiring that the composite application complies with them. The goal, then, is the selection of a service for each activity, in order to meet all the constraints, getting as close as possible to the preferences.

2.2 Multi-Agent Systems

Multi–agent systems are a viable approach in this direction allowing handling sophisticated interaction patterns. Agent-orientation is an appropriate design paradigm to enable

automatic and dynamic collaborations, especially for e-Business systems with complex and distributed transactions.

Software Agents paradigm is used to model independent and heterogeneous entities, with different objectives potentially conflicting with each other that act autonomously in dynamic environments. By the definition of software agents given in [71]:

"Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like."

Software agents are programs that execute operations autonomously, on their behalf, and more generally they are able to plan the actions to do, in order to reach goals or tasks for which they are designed. The main features and behavior that a generic software agent should have, are defined in [72]:

- *autonomy*: an agent operates without human supervision and it has some kind of control over his own actions, and on its internal state;
- *reactivity*: agents perceive their environment (which could be the physical environment, a user that acts to through a graphical interface, a collection of other software agents, internet) and respond promptly to its changes;
- *proactivity*: agents act not only in response to their environment, but they are able to take initiatives, performing actions guided by their objective.
- *social ability*: agents interact with other agents and humans through some defined protocol.

Another feature that should be of interest for developing the agents, is the software rationality [72], which can be summarized as the assumption that an agent will act with the aim to achieve its goals and will not act so to impede the achievement of its objectives, at least to the extent that its representation of the state of the world allows. In reaching their targets, rational agents try to maximize their own utility. The utility is a function where the domain is the set of all possible states of the environment, and the co-domain Real numbers, indicating the preference level of the agent on each state of the world:

$$u : S \rightarrow \Re$$

In this representation the agent will prefer the state s_1 to the state s_2 if and only if $u(s_1) > u(s_2)$. An agent is defined as self-interested [71], if it has its own preferences on the possible states of the world, and it acts in an attempt to achieve the most preferred states. Usually, states of the world which are preferred by an agent, could be also preferred by other agents. This means that in order to achieve these states, the agent should enumerate all reachable states as a result of one of its possible actions, taking into account also the other agents. This computation can be too complex when one considers that the agent's memory and computational capacity are limited. For this reason, in real applications often the agents have bounded rationality, i.e. they try to maximize their utility under simplified models of the world.

An environment where the agents interact with each other is called a multi-agent system (MAS). In this context, each agent needs a further feature, namely the ability to interact with other agents in the same environment (social ability), using a certain type of communication language [71]. The most common mechanisms of interaction between software agents are:

- *mechanisms of coordination* which coordinate their actions;
- *mechanisms for cooperation*, to achieve common goals by distributing the workload or exploiting the specific capabilities of each agent.

2.3 Automated Negotiation

In multi-agent systems automated negotiation is used to model the process of decision making to reach an agreement that meets the constraints of two or more parties in the presence conflicting interests. The conflict of interest refers to the fact that agents have conflicting interests, as in the case that occurs between a seller and buyer. As described in [32], automated negotiation is essentially a distributed search in the space of potential agreements between the different negotiators represented by autonomous agents, which involves the exchange of relevant information and aims to find an agreement that is acceptable to all participants.

In the following the main features which characterize the automated negotiation are described:

- *Negotiation Protocol* is the set of rules that govern the interaction and defines who are the actors of the negotiation, the states that characterize a trade (for example, when a negotiation has begins or ends), the events that determine the change of actors' status, and messages that can be sent by the actors in a particular state.

- *Negotiation Object* is identified by the set of parameter values which must be reached for an agreement, known as Agenda [23]. To dynamically change the agenda by adding or removing parameters may be allowed by the protocol.

- *Bidding Strategies* (model of decision making agent) can be defined formally as an apparatus which allows the agent to determine the content of the action that it will perform consistently with the protocol. In general, for a given set of negotiation protocol there are many strategies compatible with it, each of which can determine a different action. This means that a strategy can work well with a given protocol, but does not work with others. So, the choice of strategy depends on the protocol in use and by the trading scenario.

The negotiation protocol and negotiation strategies define the negotiation mechanism [39]. The following is a summary of the properties that characterize the design of a negotiation mechanism:

- *Computational Efficiency*: the strategy adopted by the agent must be a rational output within a reasonable time, i.e. an offer that corresponds to a high expected utility;

- *Efficiency in communication*: the exchange of messages between agents must take place in a reasonable time and limiting the overload in the message processing for each agent;

- *Individual Rationality*: an agent participates in the negotiation only if it is in its interests.

- *Pareto efficiency*: the output of the negotiation is Pareto efficient, if there is no other outcome that increases the utility of an agent, without there is another agent that gets worse in its utility.

In the design of the negotiation mechanism, it is difficult to get the satisfaction of all properties; for this reason, it is necessary to make a trade-off between the properties

described above. The size of the space of potential agreements is determined by the structure of the object of negotiation, where each parameter is associated with a dimension. Each proposal corresponds to a point of the space of potential agreements. Each agent is associated with a region of space that defines the points of agreement, or the points at which it can accept the offer. For each point, it is given a preference by a private utility function. An agreement is possible if the intersection of these regions defined for each agent is not empty.

The ability to produce and respond to a proposal must be specific to an agent who negotiates. In particular, each agent must be able to:

1. propose a possible point of agreement;

2. respond to a proposal indicating whether it is acceptable or not.

There are other ways to make the negotiation, in which the agents are also able to provide feedback on the proposals received (Argumentation). A feedback can be a comment on the values of the parameters or a counter-proposal that is an alternative proposal generated in response to the received offer. From this feedback, the bidder will be able to generate an offer, which takes into account the specific requests made by the buyer. In competitive environments, where there are agents that compete to supply the same service albeit with different QoS, the strategies should be private. This factor affects the efficiency of the protocol. In fact, if strategies are more like to be private, agents do not know how the deals are generated during trading, and so they are not able to determine beforehand the space of shared agreement. So, agents can start negotiating although there is no point of agreement. On the other hand, an agent that reveals its strategy, gives up to maximize the gain in utility (negotiator's dilemma [65]).

In this section, the main characteristics that must be considered in the design of a mechanism for automated negotiation are described in details. As previously outlined, the number of interactions in the negotiation depends on the number of involved agents, and on their characteristics, as well as on the object of the negotiation. Moreover, it is important to identify the most suitable negotiation protocol, so that it is able to satisfy the properties such as: computational efficiency, communication efficiency, individual rationality, Pareto efficiency.

2.3.1 Bidding Strategies

Each agent that participates in a negotiation should have a role. In most domains the set of roles is clear and it is reduced to a couple: seller, buyer. There are also domains where there is another role the Mediator, which has the task of mediating between buyers and sellers to accelerate the timing of negotiations, and to find a solution that is optimal for all agents involved in bargaining.

The cognitive abilities of each agent is based on a model of decision making, i.e. the degree of knowledge on the object traded and the strategies of the other agents. The social skills of the agent determine whether the agent is acting as an individual entity with private interests and/or as a collaborative entity with the other agents. The model of decision making allows the agent to: make an offer, reject the proposal of the other agents, formulate a possible counter proposal, accept an offer and leave the trading. Moreover, this model describes how to calculate the values of the parameters to be offered.

In [22] the main models of decision making are described: responsive decision making, and deliberative decision making.

The responsive mechanisms

The responsive mechanisms shape a reactive behavior to a number of environmental factors. These mechanisms generate offers from the linear combination of simple decay functions, called tactics, where each function models a pattern of concessions, which indicates how a counteroffer is generated, i.e. decreasing the utility of the proposal made in the previous negotiation interaction. In other words, the agents make a discount on their total utility, giving up the utility maximum and encouraging the other party to accept the offer.

Since it depends on the type of trading and conditions that are created, it can be convenient to use a tactic rather than another. The tactics are made taking into account conflicting interests between the seller and buyer agents. These tactics can be divided into the following groups: time dependent, resource dependent, imitative.

Time–dependent: if an agent has a time deadline[1] by which it must reach an agreement.

These tactics model the offer of an agent that concedes when approaches to the

[1]Deadline is the maximum time within which the thread of negotiation must end. Timeout is the maximum time available for each agent to generate an offer or counteroffer.

deadline of negotiations. Therefore, this tactic is a function time–dependent and belongs at the following subgroups of functions (Boulware and Conceder tactics.).

- Boulware: tactic keeps the value of the offers close to the initial one in the initial rounds. While, if the agent approaches to the deadline, it makes great concessions in utilities, in order to reach an agreement, providing offers near its reservation value, which represents the value lower in utility that can be offered by the agent;
- Conceder: The agent, at the beginning of negotiation, in utility, making quickly offers that come close to the reservation offer.

Resource–dependent: these tactics model the pressure to reach an agreement depending on the available resources remaining. The functions of this set are similar to those of the previous group except that the domain of the function is now the amount of resource available. The difference with the time dependent tactics resides in the fact that the time is growing, while the number of resources have a performance that is not linearly related with time. Therefore, they may use functions which estimate the change in the number of resources on the basis of how many potential buyers of the resource are negotiating.

Imitative: describes the agents that reach agreement using imitative tactics, where the counteroffer depends on the behavior of the other party in the negotiation. The tactics of this family depend on the behavior of the counterparty and how it is imitated. There are three sub-groups of tactics:

- Tit-For-Tat relative: for a given parameter, the agent gives a counteroffer in the same proportion of concession of the counter–party.
- Tit-For-Tat absolute: for a given parameter, the agent gives a counteroffer in the same amount of concession of the counter–party.
- Tit-For-Tat average: given a time window the average concession made by the other party is evaluated, and that amount is the concession made by the agent in the counteroffer.

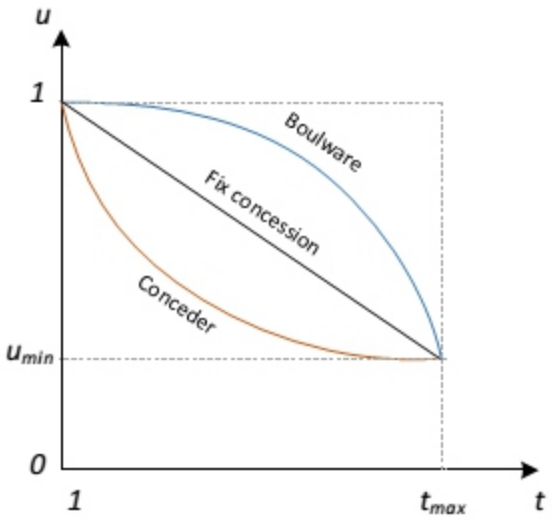

Figure 2.1: Boulware and Conceder tactics.

Deliberative mechanisms

Deliberative mechanisms are used when the object of negotiation is represented as combination of multi–parameters. They try to make the offer more attractive to the other party, without reducing in terms of utility, unlike responsive mechanisms that allow to concede in utility in iterative manner until the agreement is reached. Hence, deliberative mechanisms are designed with the objective of obtaining results that are closer to be Pareto optimal. Such mechanisms do not discriminate the offers by the utility values corresponding to the individual parameters, but through the utility value of the whole offer. These mechanisms, are divided into mechanisms of trade-off, and issue set manipulation [22].

Trade–off mechanism generates deals changing the values of attributes offered in the previous round without concession in terms of utility. In other words, a new bid is calculated by varying the values of the individual issue, but the utility remains unchanged.

Issue–set–manipulation mechanism has as objective to increase the probability of an agreement by adding and removing parameters to the offer. So, the structure of the negotiation object is dynamically altered. This strategy allows to restrict or increase the region where the point of agreement is searched.

2.3.2 Negotiation object

The object of negotiation can be characterized by a single parameter or multiple parameters. In the first case, the utility is closely related to the value assumed by the single parameter. In the second case, the object of the negotiation is composed by multiple parameters. To calculate the utility of an offer, these parameters must be in some way aggregated. Furthermore, in the case of multiple attributes there is the need to take into account the exponential growth of the space of potential agreements. In defining a negotiation, the set of parameters on which agents negotiate must be determined, together with the trading process, i.e., how to negotiate the values of the parameters. In [23], three ways to negotiate on multiple parameters are classified.

Package–deal: in the offer a value for each parameter is proposed. The parameters are dependent on each other, that is the value of a given parameter depends on the values of the other parameters contained in the offer.

Issue–by–issue: each offer contains the value of a single parameter, negotiating parameter by parameter an agreement is reached. In this procedure, the agents must also decide the order of parameters into the negotiation, also in this case, the parameter value depends on the others.

Simultaneous: similar to the package deal, but the parameters are independent, as if there are copies of the agent that negotiate independently on single parameter.

2.3.3 Negotiation Protocol

In order for heterogeneous agents to collaborate through a negotiation mechanism, there is the need of a defined standardized protocol, shared by all agents, in which a set of rules that characterize the negotiation, and the specifications of the exchanged messages format are described. For this reason, in 1996 was found a Swiss not-for-profit organization, the Foundation for Intelligent Physical Agents (FIPA), with the aim of defining a full set of standard rules for both implementing agent-based systems, specifying how agents should communicate and inter–operate in a standard way with each other [24].

FIPA has defined different kinds of negotiation protocols that describe the most common interactions between agents. The protocol defines the roles, messages and interactions models for the definition of a contract to supply of a good.

Another point characterizing the negotiation protocol, is the number of agents involved in the negotiation that is a measure of the trading complexity. The type of communication that can be established between the agents defined in the negotiation protocol, can be of three types:

one–by–one: in which two agents which negotiate with one another are involved. Typically, an agent is the seller and the other is the buyer of a good.

one–to–many: a single agent negotiates with many agents. An example are auctions, where the auctioneer sells an asset to the highest bidder, among all the agents who make an offer.

many–to–many: in which agents negotiate with each other simultaneously. In the worst case each agent is involved in negotiating with the others.

In the following, the most used the negotiation protocol defined by FIPA, Iterated Contract Net Interaction Protocol (ICNIP) in [24] is described.

Iterated Contract Net Interaction Protocol

The FIPA ICNIP [24] identified two main roles: initiator, participant. The agent who assumes the role of initiator is the agent that has an interest in receiving the supply of a good (Buyer), while the participant is a potential providers of the good (Seller). Mainly, two messages between initiator and participant are exchanged, offers and counteroffers. The flow of interaction, as represented in 2.2, is composed of five phases.

1. the initiator sends a Call for Proposals (CFP) to participants involved in the negotiation, specifying the goods requested and the terms of the contract for the supply.
2. each participant analyzes the CFP and, according to its strategies, makes its own proposal.
3. the initiator chooses participants who have sent the best deals, and sends them a message of acceptance for the supply contract of the good.
4. the initiator send a rejection message to the remaining participants.
5. once completed the provision of the good, the participant sends a message to the initiator with the outcome of the supply.

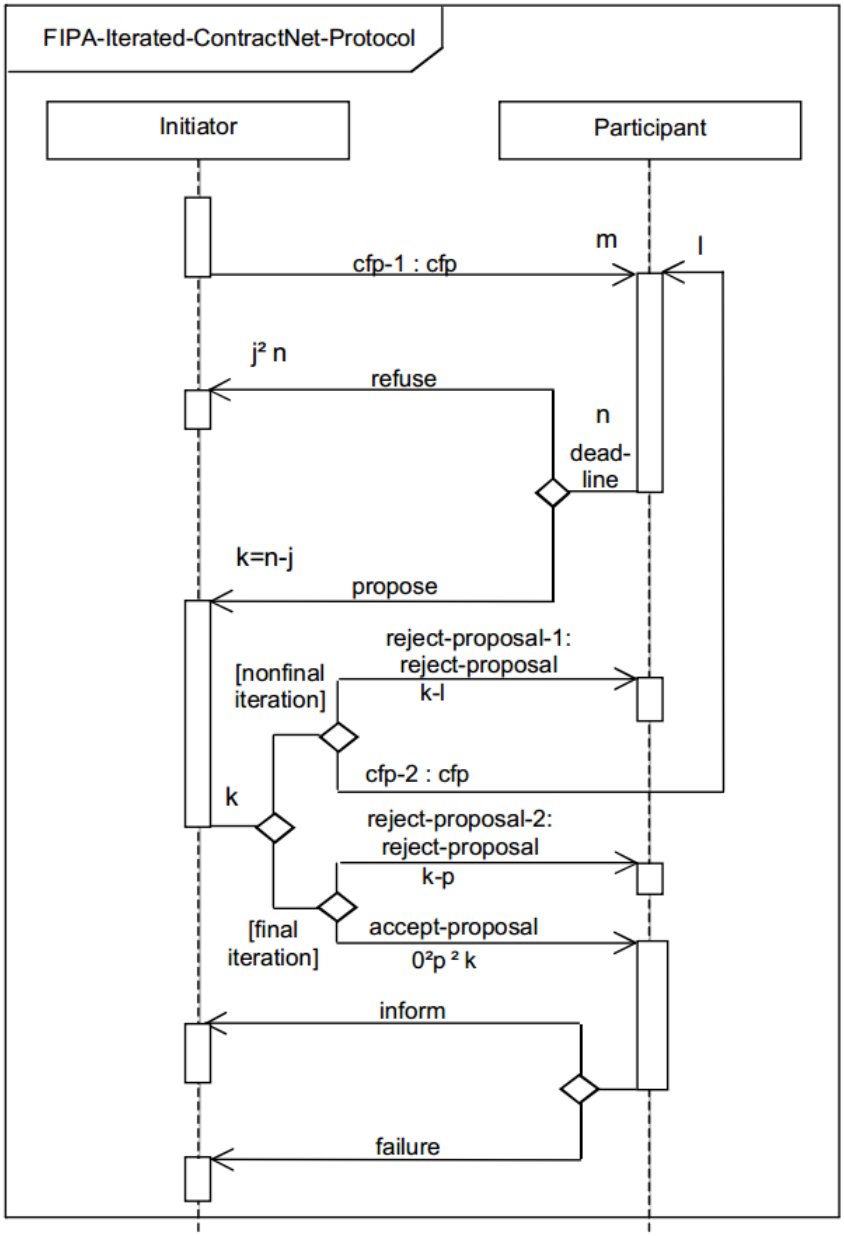

Figure 2.2: FIPA Iterated Contract Net Interaction Protocol [24].

ICNIP adopts an iterative process of offers negotiation, in which, after stage two, the initiator may decide to respond with a counteroffer. This iteration process can continue until one experiences one of following conditions:

- the initiator accepts one or more proposals (Agreement);
- all participants send a message of rejection;
- the initiator sends a message of rejection to all participant.

The protocol also allows to mark the exchanged messages in a thread of negotiation with a unique identifier, allowing agents to establish a history of negotiation and to define strategies based on the past behavior of the agents during interaction. Moreover, ICNIP supports multi–round negotiation, where increases the chances of reaching an agreement with respect to the basic version, which offers a simple exchange of single offer and counteroffer.

2.3.4 Different approaches to negotiation

Once analysed the basic elements of the automated negotiation and the features to be considered in its design, in the following there is a classification [32] of the various approaches to automatic trading presenting briefly.

2.3.4.1 Game theory

Game theory studies the interactions between self-interested agents, and for this reason it is considered into the negotiation context. The aim of this model is the pursuit of the best action that an agent can do in a given state of the environment, or that maximizes its utility function.

In order to determine the best action, the agent should think in a strategic way, i.e. foreseeing the next action, taking into account the actions carried out previously by the opponent, it is assumed that the other agent will always select the action that maximizes its own function utility. Once the negotiation scenario is defined, game theory can be applied to design the negotiation protocol and the strategy that the agent will use to maximize its utility function. An example of negotiation based on game theory is the monotonic concession protocol adopting the strategy Zeuthen [59].

The negotiation protocol between two agents, consists of a sequence of rounds and in each round, each agent sends a proposal. If the proposal made by an agent matches or exceeds the one made by the other agent in terms of utility, it is accepted and the negotiation ends, otherwise it proceeds with the next round, in which agents can make a concession or send the previous offer. If neither of the agents makes a concession, the negotiation ends without success.

If agents are to use this type of protocol to negotiate, a possible strategy that they may adopt is the strategy of Zeuthen:

- concessions are determined by the willingness to risk a conflict. An agent will be more willing to risk the failure of the negotiation, if the difference in utility between the current bid and conflict deal is low. Conversely, if this difference is high, the agent will be less inclined to risk and therefore more inclined to concede;

- the agent gives the minimum in order to avoid failure of the negotiations, so that in the next round the other agent have to concede.

2.3.4.2 Heuristics Methods

These methods, unlike game theory, address the problem of computational cost arising from determining the best possible offer. Indeed, a non exhaustive search in the space of the agreements is made, producing a solution acceptable rather than rationally excellent. This technique, therefore, has the great advantage, with respect to the one defined by game theory, to be based on realistic, assumptions, and therefore applicable to real domains. An example of a heuristic model is the set of mechanisms responsive and deliberative. In such instances, it is not possible to calculate the risk factor of the counterpart, and therefore it is not possible, as in Zeuthen strategy for game theory, to calculate the minimum concession that prevents the failure of the negotiation.

The mechanisms adopting a heuristic approach have the following disadvantages:

- allow to find a solution sub–optimally since, the entire space of possible outcomes is not fully examined;

- require an empirical analysis and simulations to be evaluated, since it is impossible to predict precisely how the system and agents will behave in each situation.

2.3.4.3 Argumentation

The idea of the argumentation is to extend communication between the agents involved in the negotiation, allowing the exchange of a wider set of information about the objects of negotiation. This information can be of different type and is called arguments [34]. They are exchanged between the agents in order to provide more information to the rejected offers, providing the reasons why the proposal was not accepted.

Similarly, the agent can provide with its proposal, argumentation indicating the reason why the other party should accept the offer. The exchange of argumentations gives the possibility to agents to make more explicit the respective regions of agreement, accelerating the trading process.

Moreover, this model allows the agent, to alter the preference of the counterpart through threats or the promise of rewards. Therefore, one agent may change the region of agreement, based on information that is not directly related to the object of the negotiation, but the representation of the environment of the agent preferences.

Models based on the argumentation are much more complex than the heuristic models and game theory based. Indeed, they require a process for evaluating, generation and selection of the argumentation, and this process leads to a further computational load for an agent.

Chapter 3

Negotiation for Smart Service Composition

This work is aimed at designing a computational mechanism automate to the composition of services in the context of Smart Cities. In particular, the research was carried out within the research project OR.C.HE.S.T.R.A. in which we investigated the use of software negotiation to compose services.

The main task of OR.C.HE.S.T.R.A. (ORganization of Cultural HEritage and Smart Tourism and Real-time Accessibility) project is to realize a platform for tourists and citizens that offers a set of solutions oriented for the intelligent enhancement of cultural heritage in Naples, with the aim to make the city smart, sustainable and green. This project addresses several problems, for example: the identification of the touristic itineraries, the analysis to model tourism flow, and the development of a monitoring system for mobility. Hence, the OR.C.HE.S.T.R.A. project is based on the realization of an intelligent multimedia platform that is able to reprocess the information present in the city, in order to offer these information as services with added value for the city, providing smart solutions for citizens and tourists. For this reason, it is crucial for the project the development of software technologies able to address the dynamic nature of the provision of these applications, that characterize the Smart Cities.

In the following section, we will present our automated negotiation mechanisms adopted for composition of Service Based Applications to provide applications satisfying users requests.

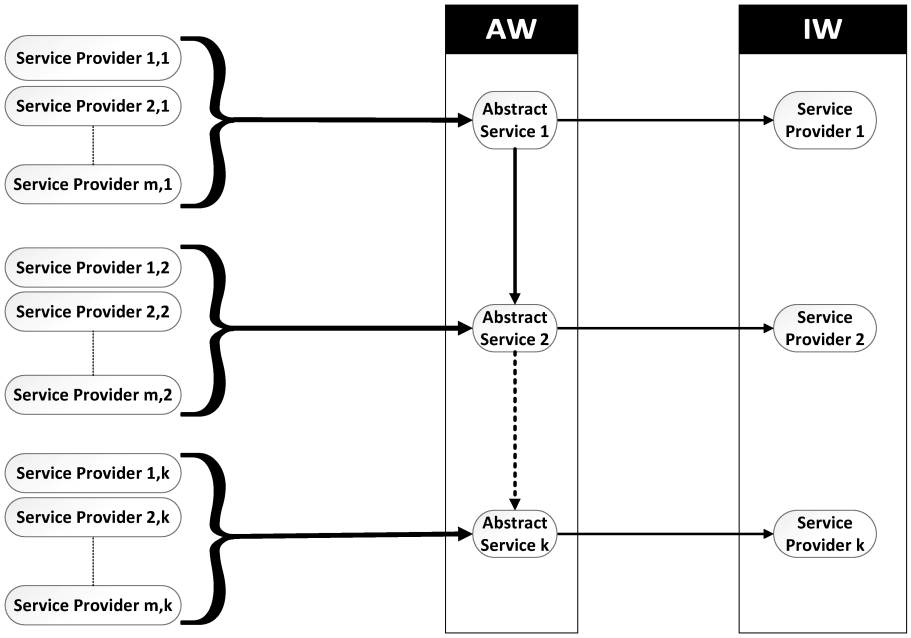

Figure 3.1: The composition process.

3.1 QoS-aware Service Composition

The service composition process, as described in section 2.1, usually starts from an abstract representation of a composition request, we refer to as an *Abstract Workflow* (AW). A simple representation of an AW, also known as the workflow structure [76], is a directed acyclic graph (DAG) $AW = (AS, P)$ where $AS = AS_1, \ldots, AS_n$ is a set of nodes, and P is a set of directed arcs. A DAG–based workflow structure contains sequence, parallelism, and choice patterns, but not loops or cycles.

Each node represents an *Abstract Service* (AS_i), i.e. a service description that specifies a required functionality. Each directed arc that connects two nodes represent a *precedence relation* among the corresponding ASs. A precedence relation $p = (AS_i, AS_j)$ of P implies that an instance of the Abstract Service AS_j cannot start its execution until an instance of the abstract service AS_i finishes its execution due to a dependence relation between AS_i and AS_j.

In order to provide a required SBA, each AS_i in the AW have to be bounded to a *concrete service* (we will refer to just as *service*), i.e. a Web service implementing the functionality specified by the corresponding AS_i. In our cases of studies services are provided by different agents, and they may be characterized by quality attributes referring

to the service non–functional characteristics Quality of Services (QOS).

It is becoming of vital importance to take into account the value of QoS attributes when selecting services to provide an executable workflow, since different customers requiring an SBA may have different expectations and requirements on its end–to–end QoS values. In fact, when requiring SBAs users specify their QoS preferences at the *workflow level* rather than at service level since they are usually not involved in the service composition process, so they are not aware of how to split a global preference at the level of single services. In order to evaluate the end–to–end quality values of an SBA, aggregation functions have to be defined for the components attributes, and these functions depend on the nature of the considered attributes, and on the workflow structure.

One step of the service composition process is to identify the optimal service selection to meet the user's QoS requirements [77]. In general, service selection can be modeled as a *Multi–dimension Multi–choice Knapsack Problem* (MMKP), which is known to be an NP–hard problem [2]. Exact solutions require a long-time computations for large problems, so heuristics approaches are necessary [30].

By the way, optimization–based approaches consider that the provider's offered values for service QoS attributes are pre–determined and not customizable, but this is unlikely in the context of a dynamic market of services. In fact, the dynamic nature of Web services, and their provision on the Internet–based market of services, require to make the following assumptions:

- the user's QoS requirements are usually expressed as end–to–end requirements on the whole SBA, and they may change according to dynamic market demand–supply conditions,
- the set of services available may change in time,
- the QoS values of services may change according to market demand–supply mechanisms,
- the dynamic nature of QoS values makes not possible to advertise these values to users at the application design–time.

These assumptions make global optimization–based approaches, as the ones proposed in [6, 78, 80] unfeasible in our scenario. For this reason, in this work a negotiation–based approach allowing to consider flexible and negotiable QoS attribute values, is adopted.

In our approach, it is assumed that service providers are modeled as software agents, we refer to as *Service Provider*s (SPs), negotiating with a *Service Compositor* agent (SC) acting on behalf of a user. Negotiation is used for the dynamic selection of the SPs able to provide services whose QoS values, once aggregated, fulfill the user's QoS preferences that represent end–to–end constraints on the entire application.

Software agents are a natural way to represent service providers and consumers and their defining characteristics are essential to realize the full potential of service-oriented systems [25]. Software agents are autonomous problem solving entities, situated in an environment, able to reach their own objectives and to respond to the uncertainty of the environment they operate in, and they are equipped with flexible decision making capabilities [33]. These characteristics make software agents a useful computational paradigm to model respectively providers that offer services at given conditions, and consumers that require services at other, sometimes conflicting, conditions.

Providers and consumers, interacting according to specified protocols and interfaces, have to establish their agreed conditions to respectively provide and consume services. Software agent automated negotiation is one of the approaches adopted for reaching agreements, so it can be used to select services in a service composition.

3.1.1 Negotiation Requirements for Service Composition

Negotiation usually takes place between two agents willing to come to an agreement on conflicting interests [32]. Most approaches in service composition that use negotiation mechanisms for each required service independently from the others relying on bilateral one–to–one negotiation mechanisms [62, 54].

In these cases, negotiation models used are of the type one–to–many in which the object of negotiation, is characterized by multiple attributes that represent the QoS parameters. In particular, one–to–many negotiation is constituted by a set of bilateral negotiations, one–to–one, not independent. A single element of this set of bilateral negotiations defines a thread of negotiation, in which offers and counteroffers are generated from the tactics as defined by the agent. Each negotiation thread is characterized by alternating offers and counteroffers, which continues as long as an agreement is achieved, or the negotiation ends due to is reached the deadline of negotiation begins reached.

In our approach negotiation is used to dynamically select the Service Providers that offer services with suitable QoS attribute values, but it is assumed that all the agents

offering services are involved in the negotiation process.

Hence, given an AW composed of n ASs (with $n \geq 2$), and k SPs (with $k \geq 1$) for each of the n ASs in the composition, the number of potential negotiating agents may vary from $n + 1$ to $n * k + 1$ agents, where 1 SC agent is in charge of finding the optimal selection of SPs, according to the QoS user's constraints, to instantiate each AS. Hence, the negotiation is necessarily one–to–many.

In order to prepare an offer $\mathbf{x}_i^t$ at negotiation round t, a service provider agent i uses a set of negotiation strategies to generate values for each negotiated issue [32]. Of course, agents must be equipped with algorithms to evaluate the received and proposed offers. The value of a specific offer is represented in terms of agent utility. Hence, the utility U_i for an agent i is a function that depends on the specific agent i, and on an offer $\mathbf{x}_i^t$ such as $U_i(\mathbf{x}_i^t) \rightarrow [0, 1]$.

Usually in SBA negotiation the strategies and utility functions adopted by the provider agents are private information. In fact, when SBAs are provided in an open, dynamic and competitive market of services, it is not realistic to assume that their strategies are shared. Furthermore, these strategies may change depending on the market demand–supply trends, so making their shared knowledge unfeasible without causing communication overheads. For these cases, negotiation mechanisms have to be designed so that negotiators can come to an agreement even though they have no prior knowledge (complete or partial) of the utility functions of the other agents involved in the negotiation. Hence, negotiation occurs in an incomplete information setting where agents utility functions, reserve values in terms of utilities, and concession strategies are private information.

The communication occurs only between the Service Providers and the Service Compositor (SPs offers are not broadcast to all participant negotiators). In addition, also SC constraints on the QoS of the composition may be private. However, even in the case of public constrains, SPs are not able to directly evaluate such constraints since they are not aware of the other offers. The negotiation mechanism allows to establish a sort of Service Level Agreement (SLA) for QoS-aware SBAs between the SC and the selected SPs. As already said, in a composition of services, its global value is given by the aggregation of the QoS values, each one provided by a service for each AS. That means that the offers received by the SC for a single AS cannot be evaluated independently from the ones received for the other ASs, so a coordination among negotiations for the single abstract services is necessary. A negotiation mechanism for service composition should

allow both to negotiate with the SPs providing services for each required functionality in the AW, and also to evaluate the aggregated QoS value of the received offers. So a *coordination step* is necessary. This type of negotiation can be very time–consuming, so the possibility for the SC to concurrently negotiate with the SPs of each AS at the same time is advisable. Generally, a buyer obtains more desirable negotiation outcomes when it negotiates concurrently with all the sellers in competitive situations in which there is information uncertainty and there is a deadline for the negotiation to complete [4]. The coordination step occur, at the end of each negotiation iteration, when the SC evaluates the aggregation of the received offers in order to allow SPs to adjust their successive offers if an agreement is not reached.

3.1.2 Related work on QoS-aware Service Composition

Several efforts have been carried out in the areas of QoS-based service selection for Service Based Applications [79]. Some works propose algorithms to select service implementations relying on the optimization of a weighted sum of global QoS parameters as in [80] by using Integer Linear Programming (ILP) methods. Nevertheless, ILP-based algorithms for selecting services are suitable when QoS data are accurate and the problem size is small (i.e. with a limited number of nodes for the Abstract Workflow and a limited number of potentially available services for each node) due to the ILP high complexity [78]. In such cases, instead of optimal solution procedures, heuristic algorithms have been proposed in the literature [10]. In this context also Genetic Algorithms have been proposed to address scalability problems as in [12, 38]. Approximation-based approaches are more efficient than linear approaches as they can handle large number of services better than linear methods. In [6] local constraints are included in the linear programming model used to satisfy global QoS constraints. In [3] Mixed Integer Programming is used to find the optimal decomposition of global QoS constraints into local constraints representing the service skyline for each service class, so allowing to prune the service candidates that are not likely to be part of the optimal solution. Typically, these works rely on static approaches assuming that QoS parameters of each service does not change during the selection process, i.e. they are predetermined, and focus on optimality and performances of the provisioning methods. Such approaches do not take into account the possibility to dynamically change the provided QoS values during the selection process that represents the basic motivation for the approach proposed in our work.

Other approaches rely on negotiation mechanisms to select services according to the QoS values [54, 62]. In most of these approaches negotiation occurs when the service provider has already been selected, and it negotiates the values of the service parameters it provides for the service. So, the negotiation process is one-to-one between service requester and the selected service provider [80]. Other negotiation-based approaches use negotiation as a mechanism to dynamically select the appropriate the service provider whose provided services best matches the service requester's non-functional requirements [27]. But usually negotiation is carried out for each required service independently from the others. So negotiation consists of multiple negotiation sub-processes each one associated with one of the required service. Each negotiation sub-process, in turn, may include multiple negotiation threads, one for each candidate provider, to choose the best service for the specific component service.

In this work, we propose a coordinated negotiation mechanism, where negotiations occur concurrently with all providers of the different required services in the composition. Coordination occurs at each negotiation steps when the aggregated QoS values offered by different SPs are collectively evaluated to decide whether to accept or not a set of offers, so to take into account the dependencies among different negotiation processes because in a composition of services the attributes values for one services cannot be determined independently from the other services in the composition.

3.2 One-Sided Negotiation Protocol

The negotiation process between two agents x and y is a bilateral interaction that consists of an alternate succession of offers and counteroffers. The process continues either until an offer is accepted by the other agent, or one of the agents terminates the interaction (e.g., because of a dead-line). An agent x accepts an offer j of y if the value of the utility the agent x obtains for that offer is greater than the utility value of the counter-offer the agent x would send back, i.e. $U_x(j_y(t)) \geq U_x(j_x(t+1))$ [32].

In order to prepare a counteroffer, an agent uses a set of tactics to generate new values for each negotiated object [21]. Of course, both agents must be provided with strategies to formulate offers and counteroffers, and they must be equipped with algorithms to evaluate the received offers. In this work for service selection, we consider a one-sided iterative negotiation mechanism allowing only the SPs to formulate new offers, and only the SC to

evaluate them.

The rationale of this choice is twofold: on one hand it makes it possible to simulate what happens in a real market of services where an SC does not have enough information on the SPs strategies to formulate counteroffers; on the other hand it takes into account that the offers for a single functionality cannot be evaluated independently from the ones received for the other functionalities. In fact, the proposed negotiation mechanism allows both to negotiate with the SPs providing services for each required functionality in the AW, but at the same time, to evaluate the aggregated QoS value of the received offers. Indeed, the SC is not able to make single counter-proposals with respect to each received offer because the change of a value of a particular QoS can influence the constraints to be fulfilled by the QoS of the other services. In other words, negotiating over the attributes of the single AS cannot be done independently from each other. Since SC does not provide counteroffers, the negotiation could be modeled simply as an auction mechanism as in [74].

However, in order to model a real market of services, it cannot be assumed that providers providing different functionality follow the same rules when bidding (such as Vickrey, English, and so on), as it happens in auction mechanisms. In fact, rules may vary according to the type of provided service (i.e., its functionality), and above all according to the trends of the market that may vary quickly, and not in the same way for all the QoS attributes. With auction mechanisms, each bidder may have its own strategy, but once the type of auction is decided, then all bidders know the rules and they have to stick to them until the auction ends. Moreover, a simple auction mechanism cannot be used in our setting because of the interdependence among the QoS attributes of the component services. In fact, it would not be possible to award an auction winner without evaluating the offers for a given AS with respect to the ones received for the other ASs in the AW.

We argue that these solutions do not model what happens in real markets of services where predefined bidding mechanisms cannot be assumed and fixed for all the service types and for all the considered QoS attributes. Therefore, traditional methods with the protocols and strategies hard-coded in the agents would not work in real market of services that are open systems. This is why a hybrid negotiation approach was used, where an auction-like protocol models the bidding of single component services, but without relying on a specific auction mechanism to allow SPs to adopt their own private strategies when bidding, and also to change them if required by the market trends.

3.2.1 The Negotiation Protocol

In this work we adopt the iterated negotiation mechanism proposed in [16], starting from the assumption that SLAs for QoS-aware SBAs have to be set by coordinating the single agreements of each component service. In the proposed approach a Service Compositor, acting on behalf of a service consumer, issues an SBA. It is assumed that for each AS, specified in the request, a set of Concrete Services (CSs) are available on the market, each one provided by a specific Service Provider with QoS attributes whose values are dynamically set by the corresponding SP. In [16], we presented a one–to–many protocol for the dynamic selection of services, based on the Contract Net Protocol (CNP) [63] that is one of the most used protocols for negotiating SLAs [54]. CNP works like a business market where manager agent asks for bids from the contractor agents and then awards tasks to suitable contractor agent. There are numerous issues related to CNP, which have been modified and added later on to it. Here, we refer to a slightly modified version of the Iterative version of the CNP as formalized by FIPA [24]. As described in Figure 3.2, the negotiation occurs between the SC, that is the initiator of the negotiation, and the SPs available for each AS of the AW, and it may be iterated for a variable number of times until a deadline is reached or the negotiation is successful. Each iteration is referred to as a negotiation round, and the deadline is the number of allowed rounds.

In this protocol the SC may set the deadline according to estimate of parameters influencing the negotiation progress, so the negotiation may take place for a variable number of iterations based on the specific situation occurring when a request is issued. The SC prepares m different call for proposals (cfps), one for each AS in the AW and, assuming that there are n SPs for each of the m AS, it sends $m*n$ cfps at each negotiation round. After waiting for the time set to receive offers, if there are not offers for each AS in the AW, the SC rejects the received proposals (reject proposal) since it is not possible to find a CS corresponding to each AS. Otherwise, it evaluates the received offers, and, if the QoS value obtained by aggregating them does not meet the preference value specified in the user's request, it starts another negotiation round sending $m * n$ reject proposals, and, at the same time, new $m * n$ cfps. If the QoS values of the received offers, once aggregated, meet the user's preference, it accepts the offers sent by the corresponding SPs (sending m accept proposals and $m * (n - m)$ reject proposals). If the deadline is reached without a success, the negotiation ends with a failure.

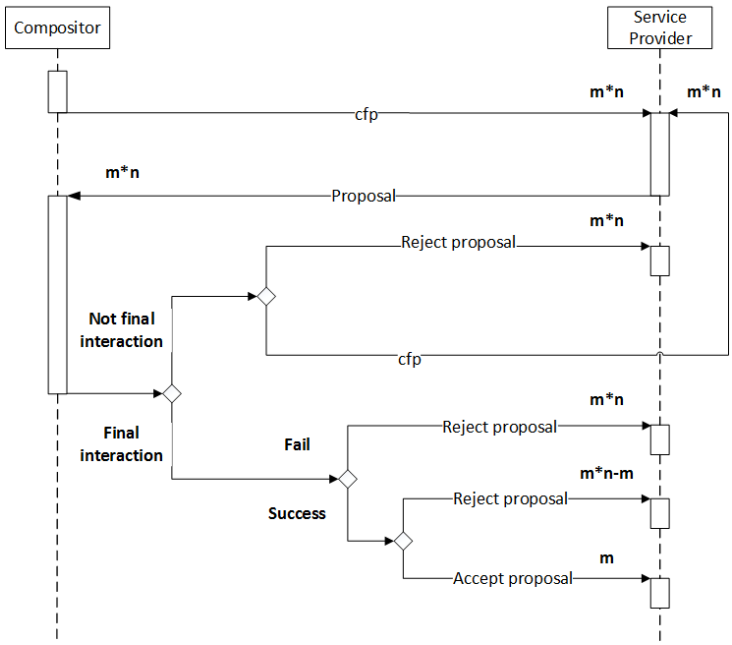

Figure 3.2: The negotiation protocol.

3.3 Gaussian Distributions for Agent Bidding Strategies

In this work, we simulate a market of services where utility functions as well as concession strategies of service provider agents are modeled through probability distributions [60][16]. The use of such distributions allows simulating the stochastic behavior of service providers with zero-intelligence [28] that can be used to approximate the trends of a volatile and open market of services. Moreover, stochastic behavior can be often observed in practical multi-agent negotiation applications [8], and probability distributions can be implemented in bounded rational agents.

When dealing with multi–agent multi–issue negotiation, the definition of utility functions is not a simple task. In economic literature, utility functions can modeled as probability distributions, useful for their mathematical properties [62]. In fact, in multi–issue negotiation most approaches usually use linear utility functions on the issues [36], while in single issue negotiation there are rarely linear, but exponential or polynomial functions, classified as boulware or conceder tactics [32]. According to [32], time dependent and resource dependent strategies are two important classes of negotiation tactics in service-oriented domains, already used for modeling B2B interactions [6]. In our approach, we

use SPs bidding strategies based on Gaussian function that model negotiation on both single issue [16], and multiple issues [60] where the issues represent the QoS attributes under negotiation. Such functions can be used to model issues whose offered value can decrease during the negotiation process until a *reservation value* is reached.

From a computational perspective, probability distributions may have relevant properties in the negotiation process since they allow to be used in different negotiation types with a uniform strategy approach, while simultaneously modeling both the concession strategy and the utility function. Moreover, Gaussian distributions are used in regression processes to estimate the concession of an opponent [70].

We show in [60] that the use of such distributions to model agent single and multi issue concession strategies, proving that their mathematical properties are useful to simulate the behavior of different agents (with different strategies) involved in the negotiation, and the composition mechanism in a simple way.

More specifically, the same utility functions and strategies used to model a negotiation on n issues for one service can be used for the case of a single issue negotiation for n services. This means that the complexity of the negotiation for SBAs depends on the number of issues, on the number of SPs, and on the number of ASs composing the SBA.

3.3.1 Single-issue Negotiation

A Gaussian function is characterized by its parameters μ and σ:

$$G(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu - x}{\sigma}\right)^2\right\} \tag{3.1}$$

As described [16], a Gaussian distribution may represents the SP utility function, and at the same time the probability distribution of the offers the SP may issue. The best offer in terms of SP own utility is represented by the μ value of the Gaussian function, i.e. $U(\mu) = 1$. Note that μ also corresponds to the QoS value of the offer with the highest probability to be selected. The Gaussian standard deviation σ represents the attitude of the provider to concede during negotiation, and it determines the reservation value of the corresponding agent $(\mu - \sigma)$. Hence, bigger values of σ correspond to smaller reservation values. The negotiation set of an issue is $[\mu - \sigma; \mu]$. In Figure 3.3 (left), two Gaussian functions of two service providers with the same value of μ, but different σ (σ_1 and σ_2)

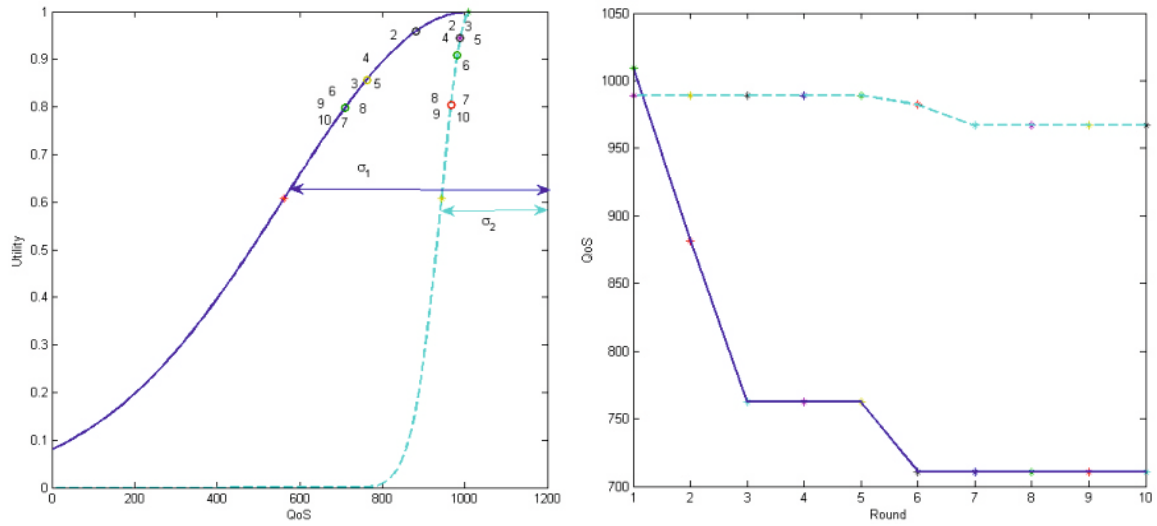

Figure 3.3: Two probability distributions for a single–issue negotiation (left), and an example of concession rate (right).

are shown, with the corresponding offers generated during a negotiation run. In Figure 3.3 (right), the trend of the offered QoS values by the two service providers is shown on varying the negotiation rounds.

At each negotiation round, an SP generates, following its probability distribution, a new utility value corresponding to a new offer. If this value is lower than the one offered in the previous round and within the negotiation set, then it proposes the new value. If this value is greater than the one offered in the previous round, or it is outside the negotiation set, the provider proposes the same value offered in the previous round. This strategy allows to simulate different and plausible behaviors of providers that prefer not having a consistent loss in utility, even though by increasing the number of negotiation rounds the probability for the provider to move towards its reservation value increases since it will always concede.

3.3.2 Multi-issue Negotiation

According to [60], the previous negotiation strategy for a single issue negotiation with multiple providers of different services, can be easily extended to a multi–issue negotiation. In this case, instead of the mono–dimensional Gaussian function, a multi–dimensional one is adopted to model both provider utility, and its attitude to concede.

The formula used to model a multi–variable Gaussian function is the simplest one,

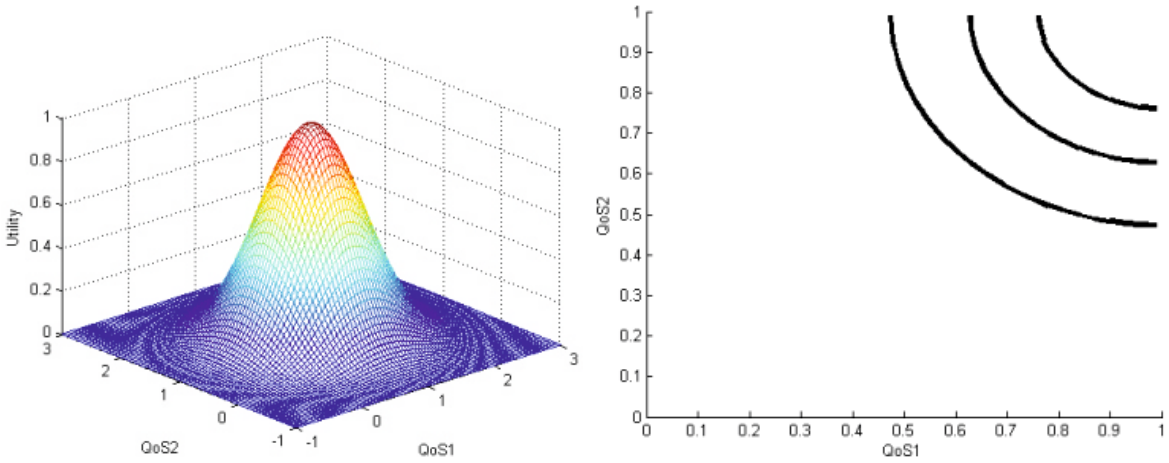

Figure 3.4: Probability distribution for multi–issues negotiation (left) and two–issues indifference curves for different utility values (right).

where we do not consider the co–variance among issues, as follows:

$$U_x(q_1, \ldots, q_n) = \prod_{i=1}^{n} \left[\exp\left(-\frac{1}{2} \left(\frac{q_i - \mu_i}{\sigma_i} \right)^2 \right) \right]$$

where, for each issue q_i, σ_i models the concession attitude when all the other $n-1$ issues are kept fixed, and μ_i is the value for the issue that corresponds to the greater value of utility ($U(q_1, \ldots, q_n) = 1$). Values of the utility function are still in the domain $[0, 1]$, that is one dimensional (see Figure 3.4 (left)).

This general representation allows to model an utility function with a non–linear dependence among different issues. Starting from this multi–dimensional Gaussian function, an utility level corresponds to an indifference curve that includes different combinations of values, one for each issue, having the same utility value for the provider. Hence, differently from the single–issue case, here, the agent can do trade–offs between values with the same utility. Trade–offs in a continuous space may become intractable since the problem is how to select a point in the space or to decide when to concede in utility. In fact, from the provider point of view, each point laying on the indifferent curve has the same utility. Some approaches suggest that the agent can make offers, proposing directly all the negotiation space (with constant utility) [36]. However, this knowledge disclosure (with the possibility to apply optimality solution theorems), cannot be appropriate in many real SBAs scenarios. Efficient heuristics to find Pareto or quasi–Pareto optimal solutions exist, and such approaches rely on the availability of counteroffers from the other agents [36, 73],

while in [17] near Pareto optimal solutions are found also for a service composition case.

In Figure 3.4 (right), we show different indifference curves generated for different values of U_x. Given an utility value, the indifference curve is a projection on the issues iper–plane that models a rational and strictly convex function (properties widely applied in economics [36]). For the same value U_x a negotiation space is individuated by a section of an ellipse that has the following equation:

$$\sum_{i=1}^{n} \frac{(q_i - \mu_i)}{a_i^2(U, \sigma_i)}^2 = 1$$

where, a_i is a value that depends on the current value of the global utility, and the concession parameter σ_i. In order for a provider agent to have the rationality requirement only a single portion of such curve is allowed.

Since the distribution is built upon single concession strategies, in our approach, the concession on the utility can be evaluated by conceding on a single issue only (its marginal utility), fixing the values of all the others, and then evaluating the new corresponding utility. Hence, when conceding in utility the agent fixes $n-1$ issues and makes a concession on a single issue selecting a value from the Gaussian distribution. If this new value is greater than the previous value (our outside the negotiation space), the agent continues to make an offer with the same utility value. However, differently from the single issue case, the actual values of the offered issues may be different (while keeping the same utility). Convexity of the utility function ensures that the agent preference on each issue is monotone when fixing the others. So, if the value increases (or decreases) the utility always decreases (or increases).

3.3.3 Negotiation for the Composition

Having adopted a Gaussian function to model both the strategies and the utilities of different providers, allowed to scale from a single–issue to a multi–issue negotiation by simply scaling the Gaussian dimensions.

In the case of a single issue negotiation, split among different services composing an SBA, the end–to–end requirements expressed by the user impose a relationship on the single issues, even though they are independently provided by different providers. Such relationship is expressed by an aggregation function (e.g., multiplication, sum) to obtain the global QoS of the entire composition. From a probability point of view, the

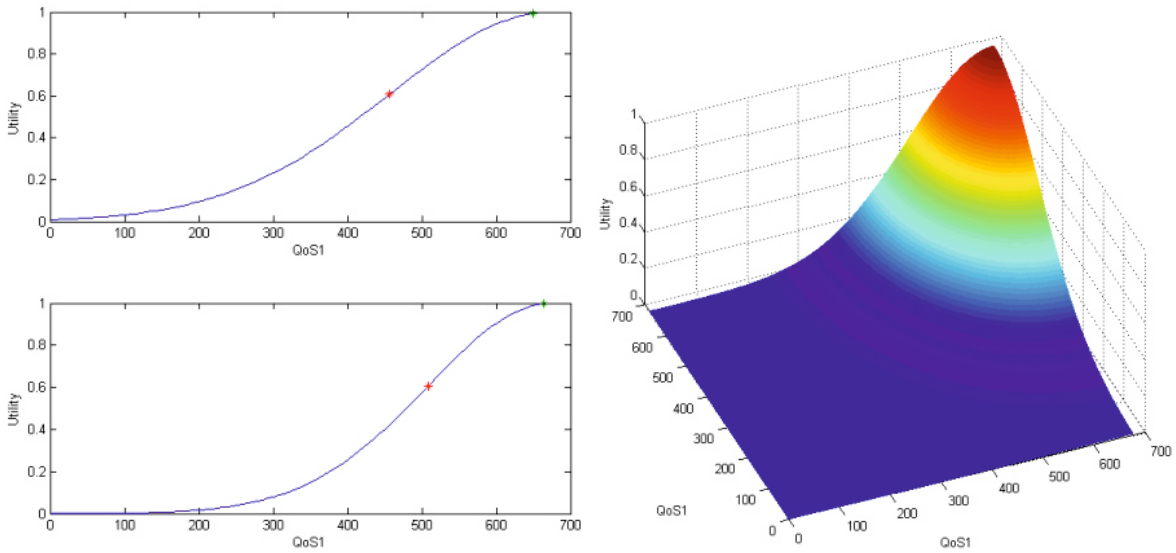

Figure 3.5: Probability distributions for a single-issue negotiation with one SP for each of the two ASs (left), and the corresponding aggregated (multiplied) probability (right).

composition of issues independently provided according to their probability distributions can be simulated with the probability distribution of the aggregated value. For this reason, the negotiation for service composition can be modeled as a multi–issue negotiation (with the probability distribution of the aggregated value) for a single service, and hence it is inherently multi–issue.

For example, let us consider the issue being the QoS *reliability* value. Such issue is typically aggregated through a multiplication operator. Moreover, let us assume to have two ASs in the composition and one SP for each AS. Each SP generates offers according to its Gaussian utility function (respectively, $P_1(x) = G(\mu_1, \sigma_1)$, and $P_2(y) = G(\mu_2, \sigma_2)$). The probability distribution of the aggregated offers ($c = x \cdot y$) is represented by $P(x \cdot y) = P_1(x) \cdot P_2(y)$, because the offers are independently provided, i.e., x and y are independent variables. This distribution is itself a Gaussian distribution since it is the product of two Gaussian ($P(c) = G(\mu_1, \mu_2, \sigma_1, \sigma_2)$). In Figure 3.5, the probability distributions of both the composed issue for the single service (right) and of the two component issues for the two services (left) are shown.

$$(P_1(x) \cdot P_2(y)) = \exp\left[-\frac{1}{2}\left(\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(y-\mu_2)^2}{\sigma_2^2}\right)\right]$$

Hence, Gaussian–based utility functions and strategies used to model a negotiation on

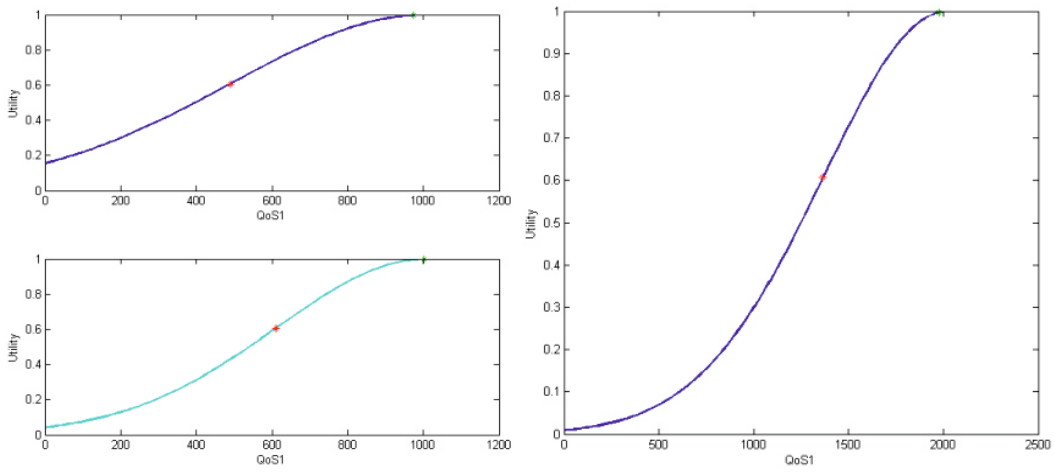

Figure 3.6: Probability distributions for a single-issue negotiation with one SP for each of the two ASs (left), and the corresponding aggregated (summed) probability (right).

n issues for one service, which represents the composition of the SBA, are the same as the ones used to model the negotiation for the composition of n services with a single issue.

In the case there are m providers for n ASs, it is necessary to consider all the possible offer combinations in order to evaluate the end–to–end constraint satisfaction. For this reason, when mapping the probability distributions of the single services into the one of the "virtually composed" service, the number of the composed probability distributions to be considered should be exactly the same as the number of all possible offer combinations, i.e., to simulate the same problem there should be m^n "virtual providers" for the single virtually composed service.

When the issues are additive (for example the *price*), the aggregation function is a sum. The relationship among independent variables, whose values vary according to probability distributions, is modeled as a convolution of their probability distributions with additive issues. This means that the probability distribution of the sum of the two independently provided issues, is the convolution of the single probability distributions of the corresponding variables ($P(x+y) = P_1(x) \otimes P_2(y)$).

$$(P_1(x) \otimes P_2(y)) = \exp\left[-\frac{(z-(\mu_1+\mu_2))^2}{2\,(\sigma_1^2+\sigma_2^2)}\right]$$

where, z is the sum variable $(x+y)$.

Hence, since the convolution of two Gaussian functions is still a Gaussian function, the resulting function can still be used to evaluate the distance of the end–to–end QoS

requirements from the aggregated QoS values received at a given negotiation round. Differently from multiplicative issues, in this case the resulting Gaussian function is still mono–dimensional. In Figure 3.6, the probability distribution of both the composed issue for the single service (right) and the two component issues for the two services (left) are shown for an additive issue.

3.4 Optimality Results

When considering multi–issue negotiation trade off among issue is possible. In these cases, as reported in the literature [73], it is mathematically proved that the adoption of an orthogonal bidding strategy leads to near Pareto optimal agreements for resource allocation problems. When adopting this strategy in service composition, the negotiation costs in terms of length of the process increase with number of considered component services and the number of available service providers.

In order to overcome this problem we proposed an extension of orthogonal bidding strategies called weighted orthogonal bidding strategies that allows on one hand to still find an agreement if it exists, and on the other hand to concurrently negotiate with all the services providers available that is not possible with the orthogonal bidding strategies. In the next paragraphs we report the details of proposed strategy.

3.4.1 Optimality in Service Composition

Let us consider an AW with n ASs (with $n \geq 2$) and m QoS issues (with $m \geq 1$) for each of them, and k SPs (with $k \geq 1$) for each of the n ASs. For each issue j the SC agent has a constraint C_j on the whole AW. The SPs formulate new offers, and the SC evaluates the aggregated value of each considered issue. In this way it is possible to simulate what happens in a real market of services where a user requesting an SBA does not have information on the SPs strategies. This means that the SC is not able to make single counter–proposals with respect to each received offer, because the change of a value of a particular QoS can impact the constraints to be fulfilled by the QoS of the other services. SC accepts an offer $\mathbf{x}_i^t = (x_{i,1}^t, \ldots, x_{i,m}^t) \in \Re^m$ of the i-th SP if the aggregated value of the offer with the values of the offers for the remaining ASs, satisfies the global constraints, so leading to an agreement.

Definition 1. *A set of exactly n offers* $(\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_n^t)$ *is an* agreement *(**A**) at round* $t \iff \sum_{i=1}^{n} x_{i,j}^t \leq C_j$, $\forall j \in m$.

If an agreement is reached with the offers sent at round t, the negotiation ends successfully at that round, otherwise all the offers are rejected and, if $t+1 \leq t_{MAX}$, the SC engages all SPs in another negotiation round until the deadline t_{MAX} is reached.

Both SC and SPs must be provided with strategies to formulate offers and counteroffers, and with algorithms to evaluate them [21]. Generally, offers are evaluated in terms of agent utility. In a multi–issue negotiation round an agent can either generate a new offer conceding in its utility (i.e., using a concession strategy), or it can select a new offer with the same utility (i.e., using a trading–off strategy in case of dependent issues). In this latter case, these offers belong to the same agent utility curve known as an indifference curve.

The i-th SP utility is evaluated in terms of its own offer $\mathbf{x}_i$. In this work we consider evaluation functions that are non–linear. Moreover, the considered evaluation functions are continuous, strictly convex and strictly monotonically increasing in each of the issues.

In general, the utility of an offer $\mathbf{x}_i$ at round t is evaluated as follows:

$$u_i(\mathbf{x}_i, t) = \begin{cases} 0 & if \quad t = T_MAX \text{ and not } (\mathbf{A}) \\ v_i(\mathbf{x}_i) & if \quad t \leq T_MAX \text{ and } (\mathbf{A}) \end{cases} \tag{3.2}$$

where, $v_i(\mathbf{x}_i)$ is the evaluation function, $\mathbf{A}$ is an agreement and T_{MAX} is the deadline.

Here, we explicitly model a collaborative approach among different providers of different services to obtain a win-win opportunity. To enhance the possibility to reach an agreement, each agent may choose the issue values corresponding to a benefit for the other agents on its indifference curve. Indeed, while keeping the same value of utility, the agent choose to collaborate in order to find an alternative that is better for the others, by trading–off among values. Competition remains among providers of the same service, and it occurs at the concession step.

In a multi-issue negotiation round an agent can either generate a new offer conceding in its utility (i.e., using a concession strategy), or it can select a new offer with the same utility (i.e., using a trading-off strategy in case of dependent issue). In this latter case, these offers belong to the same agent utility curve known as an indifference curve.

We focused on the collaborative part of the negotiation, i.e., when agents make trade–off, without considering any concession strategy. In particular, we started from the

trading–off strategy proposed in [73] for multi–agent multi–issue negotiation, called the *orthogonal bidding strategy* that was adopted when multiple agents negotiate to distribute units of resources among them. The strategy relies on the possibility of each agent involved in the negotiation to evaluate a so called *reference point* introduced in [69], taking into account the bids of all the other agents involved in the negotiation. Of course, in multi–agent negotiation a reference point cannot be directly computed by applying a one–to–one agent interaction, as in [36]. The same happens in service composition since a single agent offer cannot be used to determine another agent offer because issues are partitioned among more than two agents.

A reference point of an agent, calculated according to the offers of the other agents, as in [73], allows the agent to select, step by step, a new offer on its indifference curve as the point that minimizes the Euclidean distance between the curve and the reference point. Practically, the reference point of an agent represents the desired bid in order to reach an agreement, keeping fixed all the other agents bids. Note that in their approach at each step only one agent can send an offer, while the other offers should be kept fixed, so reference points have to be computed one at a time.

In our reference market–based scenario, it is likely that for each AS in the AW more than one SP may issue offers. For this reason, we adopt the heuristic method proposed in [3] to select at each round a set of agents (one for each AS) providing a set of promising offers at that round, by assuming that the issues that are negotiated upon are additive (so the workflow structure is not relevant for their composition). The method consists in evaluating the utility of each offer, and in selecting the most promising set of offers, one for each AS, with respect to the global constraints, by considering global constraints as upper bounds for each issue of the composition. So, a promising combination of offers $\mathcal{B} = (\mathbf{b}_1^t, \ldots, \mathbf{b}_n^t)$, one for each AS, is obtained.

Definition 2. *A selected offer $\boldsymbol{b}_k^t$ at round t for the AS_k is the one that maximizes the following equation:*

$$\sum_{j=1}^{m} \frac{\underset{\forall x_{i,j}^t \in AS_k}{max}(x_{i,j}^t) - x_{i,j}^t}{\sum_{k=1}^{n} \underset{\forall x_{i,j}^t \in AS_k}{max}(x_{i,j}^t) - \sum_{k=1}^{n} \underset{\forall x_{i,j}^t \in AS_k}{min}(x_{i,j}^t)} \tag{3.3}$$

where, $max(x_{i,j}^t)$ is the maximum $x_{i,j}^t$ issue value offered by the agent i for the issue j of all the available offers for the AS_k at time t (i.e., $\forall x_{i,j}^t \in AS_k$), while $min(x_{i,j}^t)$ is the

corresponding minimum $x_{i,j}^t$ issue value. Equation 3.3 estimates how good an offer is, by evaluating the QoS values with respect to both the ones offered by the other SPs of the same service, by taking as a reference the maximum offered value for that issue, and the QoS values of a possible combination of offers. In fact, the numerator gives an indication of how good the value of each QoS parameter is with respect to the QoS value offered by other SPs of the same AS (local evaluation), and it is then related to the possible aggregated values of the same issue for all the ASs (global evaluation).

Differently from the work of [73], the offers and the SC constraints are private information, so it is not feasible for each SP to compute its own reference point. For these reasons, in our approach, reference points for each AS are calculated by the SC, as a sort of counteroffer, at the coordination step relying on the offers selected for the most promising combination at a given round. In addition, reference points are sent to all SPs providing the same AS, so involving them again in the negotiation even though not selected. So, the SC plays the role of a sort of mediator since it is the only one that has the necessary information to compute reference points.

A reference point is defined as follows:

Definition 3. *The **reference point** for the SPs corresponding to an AS_i and to m issues at round t is:*

$$\boldsymbol{r}_i^t = (C_1 - \sum_{k \in N-i} b_{k,1}^t, \ldots, C_m - \sum_{k \in N-i} b_{k,m}^t) \tag{3.4}$$

where $\mathbf{b}_k^t$ is the last bid of agent $k \in N - i$ selected for the considered combination at that round.

In [73], the authors proved that a set of offers $(\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t)$ is an agreement at round t iff each reference point $\mathbf{r}_i$ for each agent i Pareto dominates the bid of the agent it is calculated for, i.e $r_{i,j}^t \geq x_{i,j}^t$.

In our approach a reference point, calculated according to the Definition 3, is assumed to be the reference point for the entire set of available SPs for each AS at a given round. In this way, all SPs available for each AS are able to negotiate at the successive round by formulating offers based on the value of the reference point, so to avoid discharging offers that may become more promising at successive rounds.

We show through simulation example that agents using the orthogonal bidding strategy with the reference points of Definition 3 can reach an agreement, if it exists. Never-

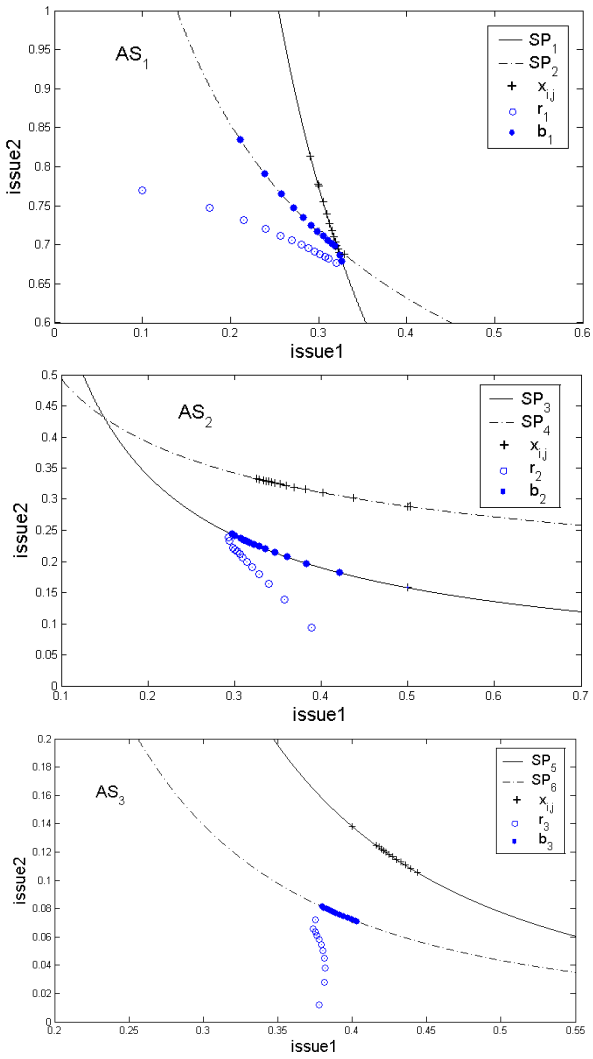

Figure 3.7: Negotiation evolution for an AW with 3 ASs, and 6 SPs.

theless, no formal proof of the convergence of the reference points towards the agreement is provided in this work.

In Figure 3.7 is shown the evolution of a negotiation execution for the considered experimental setting is shown. In particular, we plotted, for each AS, all SPs issue offers (crosses in the figure) that approach the reference point (empty circles in the figure) for that AS. The best offers selected at each round (filled circles in the figure), one for each AS, are used to compute the reference points for the successive round. The negotiation ends successfully with the set of offers respectively sent by SP_1, SP_3, and SP_6 converging to the Pareto optimal agreement.

In Figure 3.8, a different negotiation execution is reported for two provider agents of

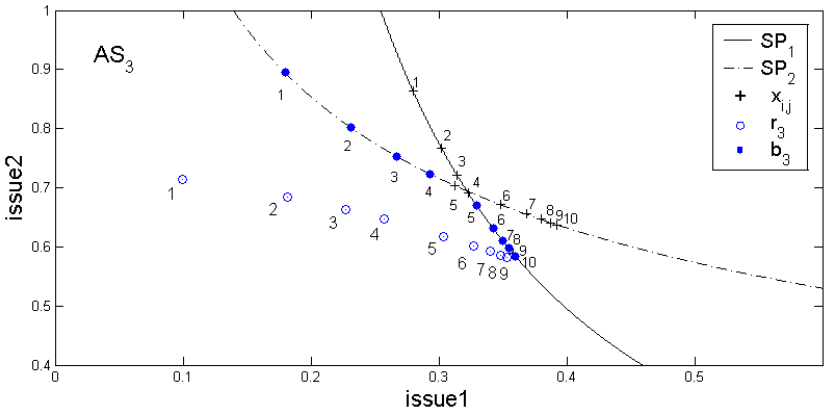

Figure 3.8: Offers evolution of the SPs for AS_3.

AS_3, indicating the reference points computed at each round, the corresponding offers respectively sent by the two agents, and the selected offers at each round. As shown, from round 1 to round 4, the offers sent by SP_2 are selected as the most promising ones, while from round 5 to round 10, the offers selected as the most promising ones are those sent by SP_1. The negotiation ends with an agreement including the offer sent by SP_1 at round 10. The possibility to negotiate at each round with all available providers for a given AS, allowed to achieve a Pareto optimal agreement with an agent that would have been discarded since it was not promising at the beginning of the negotiation. Hence, a reference point computed considering a set of single selected offers at a given round, allows to select a different set of offers at a successive round.

It could happen that an offer for an AS included in a Pareto optimal agreement may be provided by two different SPs, if the indifference curves intersect: in such a case just one of the SPs is randomly selected since the selected agent is not relevant for the Pareto optimality of the agreement.

3.4.2 Optimality in Concurrent Negotiations

Following the approach proposed in [73], the same results concerning Pareto optimal agreements are obtained, when calculating one reference point for each AS at a time within the same round. When the number of ASs increases, it is undesirable that an SP for a given AS waits for the offers of the others SPs of the remaining ASs to get its reference point since reference points are computed one at a time. This is even more crucial in an open market of services since the time spent in negotiation may prevent its

use in this scenario. To avoid this, reference points referred to a given round t should be computed relying only on the offers available at the previous round as follows:

Definition 4. *The **timed reference point** for the SP_i corresponding to an AS_i at round $t+1$ is:*

$$\bar{\boldsymbol{r}}_i^{t+1} = (C_1 - \sum_{k \in N-i} x_{k,1}^t, \ldots, C_m - \sum_{k \in N-i} x_{k,m}^t) \tag{3.5}$$

where, for simplicity there is one SP agent for each AS.

Unfortunately, with this definition of reference point, the convergence of the orthogonal bidding strategy is not guaranteed, but it can diverge and lead to an oscillatory behavior. This is due to the fact that reference points are concurrently computed at round t, and used by the SPs to formulate bids at round $t+1$. This prevents the adjustment of bids for each AS, step by step, within the same round that is a prerequisite for the convergence to the agreement. In fact, by concurrently computing the reference points for all AS at a given round, all the SPs adjust their offers at the successive round assuming that the other offers are kept fixed, without considering that the other offers are adjusting at the same time. This may cause that SPs tend to adjust their offers more than necessary. On the other hand, considering the offers at the previous round when computing reference points, is the only way to concurrently negotiate with the SPs for all ASs, so avoiding that the deadlines for each round depend on the number of ASs.

To keep the convergence of the orthogonal bidding strategy, while keeping the possibility to concurrently compute reference points, it is necessary to provide SPs with reference points that allow for different adjustments of bids, in terms of different "weights" that depend on the issue values of the offers with respect to their aggregated values. For this reason, we introduce a new reference point, named the *weighted reference point* ($\hat{\mathbf{r}}_i^t$) as follows:

Definition 5. *The **weighted reference point** for the SP_i corresponding to an AS_i at round $t+1$ is $\hat{\boldsymbol{r}}_i^{t+1} = (\hat{r}_{i,1}^{t+1}, \ldots, \hat{r}_{i,m}^{t+1})$, with $\hat{r}_{i,j}^{t+1}$ defined as follows:*

$$\hat{r}_{i,j}^{t+1} = \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t} \cdot \bar{r}_{i,j}^{t+1} = \omega_{i,j}^t \cdot \bar{r}_{i,j}^{t+1} \tag{3.6}$$

where $\omega_{i,j}^t$ is the weight of the issue value at time t compared to the aggregated value of all the bids for that issue, and $\bar{r}_{i,j}^{t+1}$ is the timed reference point of Definition 4.

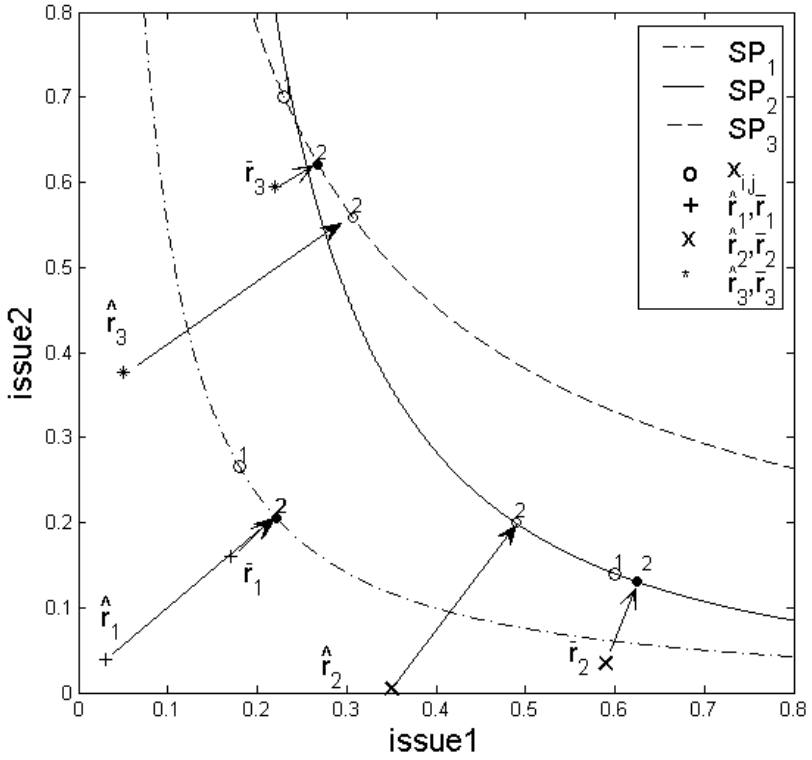

Figure 3.9: Execution of 2 rounds of negotiation with $\hat{\mathbf{r}}_i^t$ and $\bar{\mathbf{r}}_i^t$.

In Figure 3.9, the behavior of a negotiation in the first two rounds is reported showing reference points and offers in the case of weighted and timed reference points with the same initial configuration. As shown, for SP_1 the $\hat{\mathbf{r}}_1^t$ value corresponds to a scaled version towards the origin of $\bar{\mathbf{r}}_1^t$, since the relative weights of the two issues are comparable in the overall agreement. Instead, for SP_2 and SP_3 the weighted reference points lead to different new bids (number 2) with respect to the case of timed reference points.

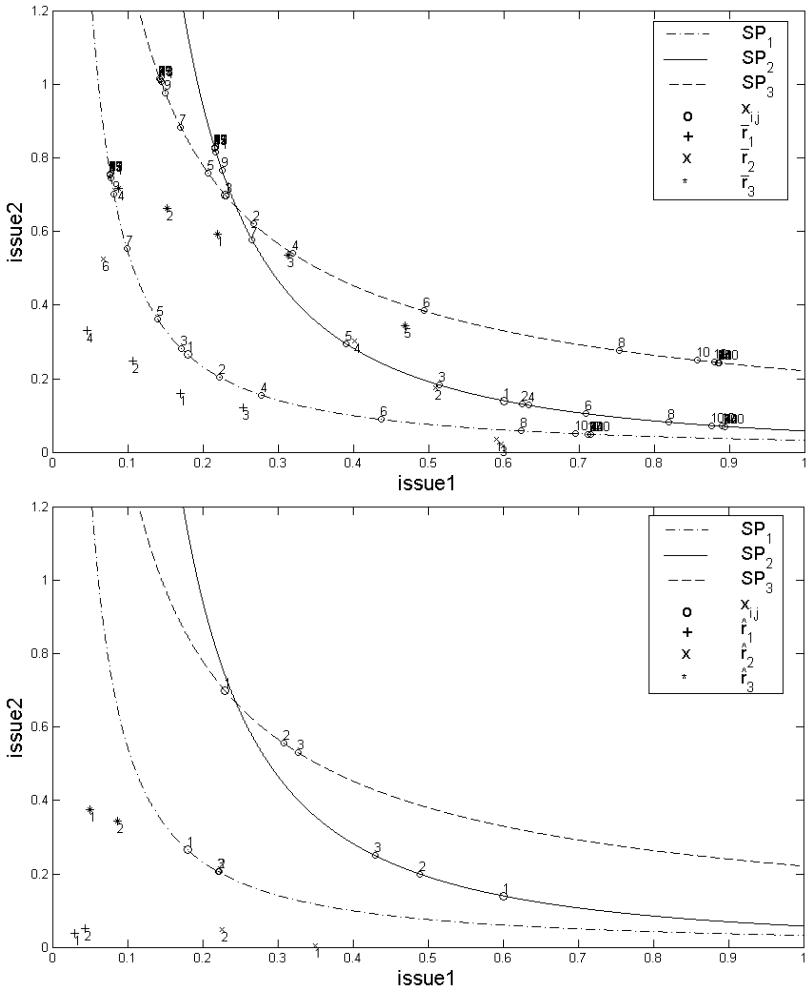

Figure 3.10: $\bar{\mathbf{r}}_i^t$ (top) and $\hat{\mathbf{r}}_i^t$ (bottom) convergence.

In Figure 3.10, a complete negotiation execution is shown when reference points are computed respectively according to the Definition 4 (see Figure 3.10 top) and Definition 5 (see Figure 3.10 bottom), starting from the same configuration of SPs and ASs. In the first case, the negotiation does not converge to an agreement in 100 rounds, while in the second case such agreement is reached very quickly. These experiments suggest that when considering weighted reference points they converge to an agreement and, if it exists, it can be found through a *weighted orthogonal bidding strategy*. Hence, reference points can be concurrently computed.

With the weighted reference point of Definition 5, it is not possible to prove the convergence of each reference point to the corresponding bid in the agreement. Nevertheless, it is possible to prove the following Lemma.

Lemma 1. *A set of offers* $\mathbf{x}^t = (\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t)$ *is an agreement at round* $t \iff$ *the weighted reference point for the* SP_i *corresponding to an* AS_i, $\hat{r}_i^{t+1}$ *Pareto dominates its weighted bid at the previous round, i.e.* $\hat{r}_{i,j}^{t+1} \geq \omega_{i,j}^t x_{i,j}^t$ *with* $j \in M$.

Proof. Assuming that $\mathbf{x}^t$ is an agreement, then $\sum_{i=1}^n x_{k,j}^t \leq C_j$, hence $\frac{C_j}{\sum_{i=1}^n x_{k,j}^t} \geq 1$. Substituting Definition 4 in Definition 5, it follows that:

$\hat{r}_{i,j}^{t+1} = \omega_{i,j}^t \cdot r_{i,j}^{t+1} = \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t} \cdot (C_j - \sum_{k=1}^n x_{k,j}^t + x_{i,j}^t) = x_{i,j}^t \cdot (\frac{C_j}{\sum_{k=1}^n x_{k,j}^t} - 1 + \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t})$

From the agreement condition, the first term in the parenthesis is greater than one, hence the following inequality is obtained if this term is substituted with 1.

$\hat{r}_{i,j}^{t+1} = x_{i,j}^t \cdot (\frac{C_j}{\sum_{k=1}^n x_{k,j}^t} - 1 + \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t}) \geq x_{i,j}^t \cdot \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t} \geq x_{i,j}^t \omega_{i,j}^t$.

If $\hat{r}_{i,j}^{t+1} \geq \omega_{i,j}^t \cdot x_{i,j}^t$ then $\omega_{i,j}^t \cdot \bar{r}_{i,j}^{t+1} \geq \omega_{i,j}^t \cdot x_{i,j}^t$ by definition of weighted reference point. Given the definition of ω and timed reference point, the inequality can be rewritten as:

$\frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t} \cdot (C_j - \sum_{k=1}^n x_{k,j}^t + x_{i,j}^t) = x_{i,j}^t \cdot (\frac{C_j}{\sum_{k=1}^n x_{k,j}^t} - 1 + \frac{x_{i,j}^t}{\sum_{k=1}^n x_{k,j}^t}) \geq \omega_{i,j}^t \cdot x_{i,j}^t$.

Hence, $\frac{C_j}{\sum_{k=1}^n x_{k,j}^t} - 1 + \omega_{i,j}^t \geq \omega_{i,j}^t$, and $\frac{C_j}{\sum_{k=1}^n x_{k,j}^t} \geq 1$, that is an agreement.

□

The validity of the Lemma allows to conclude that, when trading–off among possible offers with the same utility, the weighted orthogonal bidding strategy leads to an agreement.

3.4.3 Related work on multi-issue negotiation

Several approaches have been proposed in the literature for selecting services characterized by QoS attributes for SBAs [46]. Typically, these works rely on static approaches assuming that QoS parameters of each service do not change during the selection process (they are predetermined), and they focus on optimality and performances of the provisioning selection method. Such approaches do not take into account the possibility to dynamically change the provided QoS values during the selection process that represents the basic motivation for the use of negotiation in this work.

There are approaches that propose negotiation mechanisms to select services according the QoS values [54, 62]. In most of these approaches negotiation occurs among a specific provider selected according to some criteria, that negotiates the values of the QoS parameters it provides with a service requester. Hence, the negotiation process is one–to–one between the service requester and the selected service provider [80, 20]. Other approaches use negotiation as a mechanism to dynamically select the appropriate service provider

whose provided service best matches the requester agent QoS requirements [27], as in our case. But usually negotiation is carried out for each required service independently from the others. Attempts to propose a coordinated negotiation with all the providers of the different required services in a composition have been proposed, as in [75], but they introduce a Negotiation Coordinator that instructs the negotiation of the single component services by decomposing end–to–end QoS into local QoS requirements. On the contrary, in the present work, a coordinated negotiation mechanism is proposed, where negotiations occur concurrently with all providers of the different required services in the composition. Coordination occurs at the end of each negotiation step when the aggregated QoS values, offered by different SPs, are collectively evaluated to decide whether to accept or not a set of offers, so to take into account the dependencies among different negotiation processes.

Moreover, the service composition domain requires that the negotiation have to proceed as a package–deal multi–issue negotiation. While single–issue negotiation is widely studied in literature, multi–issue negotiation is less mature [36]. Typically, multi issue–negotiation approaches con be classified as mediated or not mediated ones. Most of the not mediated approaches rely on bilateral interactions [21, 5]. In this framework a variety of searching methods are proposed in literature, as for example, similarity criteria based search [20], or decentralized search [36]. In this thesis, we deal with the problem of multi–issue negotiations where the component issue values are provided by multiple agents, and thus a requester agent is negotiating with multiple trading partners. In multi–issue, multi–agent negotiation literature, it is often assumed that there is an unbiased mediator who collects the agents preferences and propose offers to the trading agents [36, 57, 31, 18]. In this work, the SC agent plays a sort of mediator role. In [36], the authors propose a Pareto optimal mediating protocol where, at each step, the mediator provides a negotiation baseline and the agents propose base offers on this line. In [57], the authors use one–to–many multiple negotiations with a coordinator able to change the strategies of a specific negotiation thread. In [31], the authors proposed a protocol for multi–issue negotiation with non linear utility functions and complex preference spaces. They propose a simulated annealing–based approach as a way to regulate agent decision making, along with the use of a mediator.

In this work, we do not use concession strategies, but we only focus on the trading–off. Trading–off to find optimal solution in bilateral multi–issue negotiation was addressed in [36, 81, 20]. In particular, in [36, 81] an alternating projection strategy was proposed,

with reference points evaluated with respect to the last offer of the other agent. In [82] such strategy was extended to the multi–agent case, by evaluating reference points as a mean sum of all the offers at each step. Differently from our case, in [82] an agreement corresponds to a single point in the negotiation space, and weights are the same for all the agents. In [20], the authors used the notion of fuzzy similarity to approximate the preference weights of the negotiation opponent in order to select the most similar offer to the last received offer in a pool of generated offers by random trade–offs. In [18], the authors present a constraint proposal method to generate Pareto–frontier of a multi–issue negotiation corresponding to a given reference point. In practice, the mediator adjusts a hyperplane going through a predetermined reference point until the agents most preferred alternatives on the hyperplane coincide. By choosing reference points on the line connecting the agent global optima, Pareto optimal points are produced, and the mediator's problem has a solution when the number of issues is either two or any odd number greater than two [35]. In [73], the authors present an automated multi–agent multi–issue negotiation for allocation of unit resources, similar to our case. The proposed bidding strategy requires that at each round the agents make bids in a sequential order. This requirement is not feasible in a service composition scenario because it will drastically increase the overall negotiation time. Moreover, in [36] for the case of a two agent negotiation, and in [73], it was shown through simulations that agents always reach an agreement by using their respective bidding strategy, even though no formal proof of convergence is given. In [81] for the two agent case, such convergence is proven.

Generally, a buyer gets more desirable negotiation outcomes when it negotiates concurrently with all the sellers in competitive situations in which there are information uncertainty and deadlines [47, 48]. A model of concurrent negotiation was addressed in [5], where agents are allowed to make counter–proposal without having received proposals from all its trading partners. In [47, 48], the multiple negotiation threads still happen in the same negotiation round, as in our case, but the heuristic methods used by the negotiation coordinator strongly depend on history information about trading partners and negotiation environment. In our dynamic market based scenario, past information is not always relevant to drive negotiation.

3.5 Experimentation on negotiation for service composition

We report several experiments in order to evaluate the impact of the number of SPs and ASs on the negotiation progress. In particular, in subsection 3.5.1, we carried out a set of experiments to extract useful information on the trends of the adopted negotiation mechanism in a set of predefined configurations characterized by negotiation parameters affecting the negotiation protocol that are the number of allowed negotiation rounds, the number of ASs composing the required application, and the number of SPs involved in the interaction. Moreover, we carried out a set of experiments reported in subsection 3.5.2 to numerically evaluate negotiation trends in the case of multiplicative and additive QoS issue in service composition, comparing negotiation trends in the case in which the composition problem is the provision of a single composed service.

3.5.1 One-issue Negotiation

In this subsection, we report the experiments were carried out to numerically validate some hypothesis concerning specific aspects of the negotiation, with the main objective to individuate negotiation parameters values for which it is possible to foresee the trend of negotiation in advance. The efficiency of the negotiation mechanism is evaluated with respect to two performance measures: negotiation outcome (i.e., the utility of the solution and/or the negotiation success rate), and communication costs. For the time being, the negotiation communication cost is evaluated with a simple measure of the number of bits communicated [49] or, equivalently, the number of ex-changed messages, assuming a fixed cost per transmitted messages. In this research, evaluation of communication costs does not consider the time actually spent in negotiation including both the time necessary to process the exchanged messages, and the time necessary for the messages to travel from the sender to the receiver to actually exchange the messages. This is because we are interested in determining the negotiation parameters affecting the communication cost, and not to define metrics to measure it. In multi-agent systems literature, communication cost is evaluated also in terms of the achieved coordination [11], and to decide when to communicate [56], or when to change the coordination mechanism/protocol. Here, the communication cost of the negotiation process is also evaluated as the communication

overhead with respect to the success of the negotiation process resulting in the composition of the required SBA. In principle, the collected information may be used by the SC to avoid, when possible, useless expensive negotiations.

3.5.1.1 Compositor and Providers Utility Functions and Strategies

Here, we introduce the decision making algorithm for the SC evaluation of proposals and for the strategies to generate an offer by the SPs, proposed in [16]. The SC receives service offers at each negotiation round. It checked that there is at least one offer for each AS and it evaluates if the global constraints specified by the user are met. In literature different mechanism to evaluate the constraints satisfaction are proposed. Here, the SC evaluates, at each negotiation round, the received offers (both for the same AS, and for the different ASs) by solving a Linear Programming problem formulated as in [3]. In the case of a single additive QoS parameter (e.g. "price"), the SC evaluates the utility of the jth SP for the ith AS as follows:

$$U_{SC}(o_{i,j}) = \frac{max_i(price_{i,j}) - price_{i,j}}{\sum_{i=1}^{m} max_i(price_{i,j}) - \sum_{i=1}^{m} min_i(price_{i,j})} \tag{3.7}$$

Equation 3.7 provides a way to evaluate the utility for the SC of one offer with respect to both the ones offered by the other service providers for the same service, and to the entire workflow. It gives an indication of how good the value of each offered QoS parameter is with respect to other values of QoS offered by other SPs of the same abstract service i (local evaluation) by taking as a reference the maximum offered value for that parameter. Denominator of Equation 3.7 gives an indication of how good the value of each parameter is with respect to the possible aggregate values of the same parameter for all the abstract services (global evaluation). It should be remarked that the SC does not formulate counteroffers for each AS since a counteroffer for one SP will depend on the offers received from the SPs of the remaining ASs. Furthermore, the SC's utility is related to the distance between the QoS preferences, expressed at the time of the request is issued, and the aggregated QoS values obtained by combining the best offers (for each AS) evaluated according to Equation 3.7. It is normalized so that it is 1 in case of negotiation success and 0 for failures. In the case of price SC's utility is expressed as follows.

$$U_{SC} = 1 - \frac{\sum_{i=1}^{m} price_{i,s} - reqPrice}{reqPrice} \tag{3.8}$$

where, $price_{i,s}$ is the price offered for the ith AS by the selected s PS, and so $\sum_{i=1}^{m} price_{i,s}$ is the aggregated value for the price. $reqPrice$ is the user requested price for the AW.

On the contrary, SPs apply their own strategies to formulate their offers. These strategies are modeled as a set of functions that are both time and resource dependent [32], and they take into account both the *computational load* of the provider, and the *computational cost* of the provided service. The computational load of the provider accounts for the number of requests it agreed to fulfill, i.e., the amount of service implementations it will deliver, while the computational cost of the service represents a measure of the complexity of the provided service, i.e., the more complex the service is the higher its expected cost is. SPs strategies to concede in utility are modelled, as in [16], as Gaussian distributions. This assumption models the scenario where services providing the same functionality have the same "market price" corresponding to the maximum utility for the SP providing that service. At each negotiation round, the SP generates, following its provision strategy, a new value of utility corresponding to a new offer to be sent to the SC by comparing the new offer with the previously generated one to decide whether to submit it or not. This is a variation of the standard negotiation mechanisms where the utility of a new generated offer is compared with the last one generated by the other negotiation partner.

3.5.1.2 Parameter Tuning

Before carrying out the experiments, it is necessary to evaluate the impact of the values of the k_i parameter on the negotiation process for different configurations, so to find k_i values for which negotiation is required to find feasible solutions, i.e. composition of services whose QoS values are within the preferences specified in the user's request. Initially, negotiation is required when all the values of k_i are greater than one. In order to consider services with a different degree of complexity, we randomly distribute k_i values in the interval $[2/3, 5/2]$ for the services composing the SBA. The lower limit of $2/3$ is to simulate ASs with simple features (i.e., with a low $bestPrice_i$), while the upper limit of $5/2$ is to simulate the ones with more complex features. For each SP, the associated σ_i value is randomly selected in the interval $[bestPrice_i * 0.25, bestPrice_i * 0.5]$. The lower bound ($resValue_i = bestPrice_i - bestPrice_i * 0.5$) is for SPs willing to concede till half of their $bestPrice_i$, while the upper bound ($resValue_i = bestPrice_i - bestPrice_i * 0.25$) is for SPs less willing to concede. The considered setups for the experiments vary the

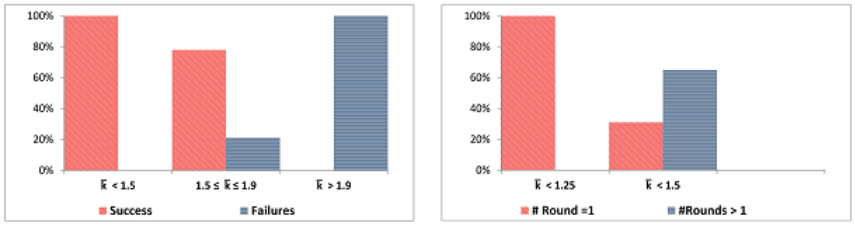

Figure 3.11: Percentage of failures and successes varying the k value (left). Number of rounds for successes varying the k value (right).

number of ASs (i.e., 2, 5, 10), and the number of SPs for each AS (from 2 to 32). For each configuration 50 executions were carried out. The average success rate on the total number of executions, for all the considered configurations, is 80%. This indicates that the interval $[2/3, 5/2]$ of k_i is reasonable with respect to the user's request. Then, the impact of the mean value of the k_i parameter for each single run is evaluated to determine a reasonable value to be set for all the other experiments. In order to do so, the obtained results are split into two classes: executions that led to successes and executions that led to failures. Within each class, the mean value of k_i is calculated over 50 executions for each considered configuration. The obtained results show that for average values of k_i greater than 1.9, the failure rate is 100%, while for average values of k_i less than 1.5, the success rate is 100% (see Figure 3.11-left). More specifically, for mean values of k_i less than 1.25, there is no need to negotiate since the values of offers sent at the first negotiation round already satisfy the user request (see Figure 3.11-right). While, mean values of k_i in the interval [1.25, 1.9] corresponds to users' requests comparable to the market prices of the services composing the AW, but that still need more than one round of negotiation to reach an agreement. Hence, for the following experiments we consider mean k_i values in such interval.

3.5.1.3 Experimental Settings

The configurations considered for the following experiments set different deadlines at 1, 10, 20, 30, 40, and 50 negotiation rounds, and for each deadline the number of SPs at 2, 4, 8, 16, 32, with a fixed configuration for the AW with 5 ASs. Let us highlight that for a deadline of 1 round, the protocol is based on m concurrent Contract Net Protocols (CNP). We also considered a configuration where the negotiation occurs only with one SP for each AS, that is the best SP in terms of $U_{SC}(O_X)$, where O_x are the offers received at

	$bestPrice_i$(\$)				
# ASs	1	2	3	4	5
5	648	540	351	270	216

Table 3.1: Distribution of services costs for 5 ASs.

round 1. In this case, the deadline varies from 10 to 50 rounds. The user requested price is 1350\$ (reqPrice). For each AS a default price $bestPrice_i$ is set, and the corresponding SPs will send as initial offer a price randomly extracted in the neighborhood of $bestPrice_i$ $[bestPrice_i - 5\% * bestPrice_i, bestPrice_i]$. This is because, even though the market price of services corresponding to the same AS is the same, to model a real market of services the variability of the first offered price is introduced. $bestPrice_i$ values for the considered ASs are shown in Figure 3.1.

The σ value randomly varies for each SP in the range $[0, bestPrice_i/2]$, so with respect to the previous parameter tuning experiments, we also include SPs with the maximum computational load that are not willing to concede (i.e. $\sigma = 0$). This will increase the number of possible failures in the negotiation process. In all the experiments 100 runs were performed for each of the described configurations.

3.5.1.4 Numerical Evaluation of Hypothesis

In this paragraph we report the numerical results obtained for the different configurations to validate a number of hypothesis concerning the trends of the negotiation that could support the decision-making mechanism of the SC.

Hypothesis-1 *It is not worth negotiating with one SP for each AS.*

The first set of experiments was conducted in order to evaluate the percentage of negotiation successes when negotiation takes place with only 1 SP for each AS. The configurations considered for the experiments vary the number of rounds (at 10, 20, 30, 40) with a fixed number of ASs (at 5), and with a fixed number of SPs (at 8) for each AS. At the first negotiation round the "best" SP is selected by evaluated the best-received offer, for each AS. In Figure 3.2 the percentage of negotiation successes is reported, and the results show that selecting the best SP at round 1 results in a low success rate. In fact, only the 12% of success rate is obtained with 30 or 40 negotiation rounds allowed, so in this case it is not worth negotiating at all.

	1 SP for each AS			
# Rounds	10	20	30	40
% Successes	1	6	12	12

Table 3.2: Success rate varying the number of rounds with only the "best" SP for each AS.

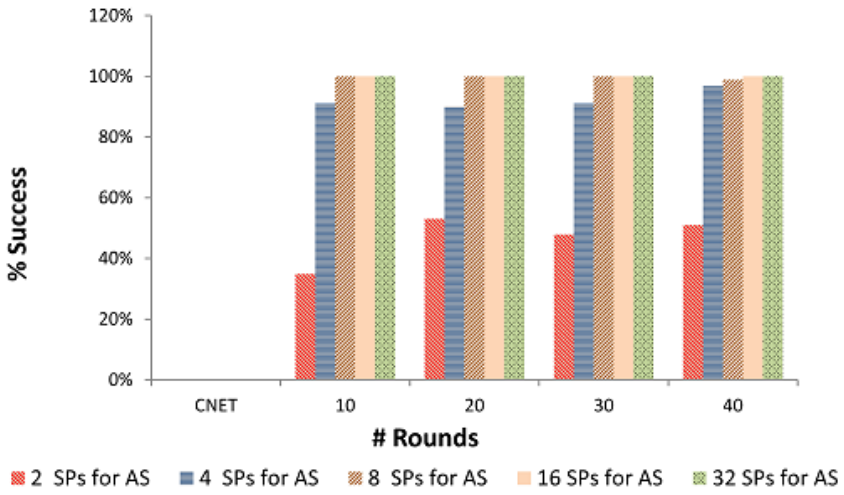

Figure 3.12: Success rate varying the number of rounds and the number of SPs for each AS.

Hypothesis-2 *The percentage of successes increases by increasing the number of negotiation rounds and the number of SPs for each AS.*

We evaluated the percentage of negotiation successes varying the number of rounds (at 10, 20, 30, 40), and the number of SPs (at 2, 4, 8, 16, 32) for each AS (fixing the number of ASs at 5). In Figure 3.12 we plotted the percentage of negotiation successes in the case negotiation occurs with all the available SPs for a fixed number of negotiation rounds. As expected, for a deadline of 1 round (simple CNP protocol) we have 100% of failures since the specific settings of the tests require a negotiation phase to find a feasible solution.

By increasing the number of rounds, a better success rate than the one shown in Table 3.2 is obtained by just adding another provider for each AS. In fact with 2 SPs for each ASs, 50% of successes are obtained with 30 or 40 allowed negotiation rounds, as shown in Figure 3.13. Scaling up the number of SPs from 4 to 32 the success rate increases from 90% to 100% just after 10 rounds. These results support the choice to negotiate with all the available SPs, since the increase in the communication cost coming from negotiating with more SPs is partially compensated by an in-crease in the negotiation success rate.

Hypothesis-3 *Negotiation with all the SPs for a fixed number of rounds introduces a considerable overhead in communication.*

In order to evaluate the computational cost of the protocol due to communication occurring between the SC and the available SPs, we calculated the maximum number of exchanged messages for different configurations. The original formulation of CNP requires *O(n)* messages, where n denotes the number of participants. In the considered version of the protocol, as described in [16], at each negotiation round the SC sends $m * n$ call for proposals, receives at the most $m * n$ possible offers, and it sends back at most m accept and/or $m * n$ reject messages. This means that for each round the cost of communication in terms of exchanged messages is at most $3 * n * m$ (i.e., $O(n * m)$). The maximum numbers of exchanged messages are reported in Table 3.3 for the same configurations as the previous experiments.

	1 SP for each AS					
# Rounds	1	2	4	8	16	32
1	15	30	60	120	240	480
10	150	300	600	1200	2400	4800
20	300	600	1200	2400	4800	9600
30	450	900	1800	3600	7200	14400
40	600	1200	2400	4800	9600	19200

Table 3.3: Maximum number of messages varying the number of SPs and the deadline.

A comparative evaluation of Table 3.3 and in Figure 3.12 shows that, provides information on the trade-off between communication costs and success rate. In particular, from configurations with 2400 exchanged messages to configurations with 9600 no variation in success rate is obtained (that is stable at 100%). This means that from 2400 onward there is only a communication over-head without any gain in the success rate. A first conclusion of this evaluation is that negotiating with 16 SPs for each AS with respect to negotiating with 8 SPs will not change the success rate, but it will only require, hypothetically, more exchanged messages. Hence, in this case, selecting a sub-set of available providers reduces the cost of communication without affecting the success rate. Moreover, as already showed in Figure 3.2, such selection cannot be made only evaluating current offers because promising providers may change their concession strategy during the interaction. So, it is possible to randomly select the SPs to negotiate with.

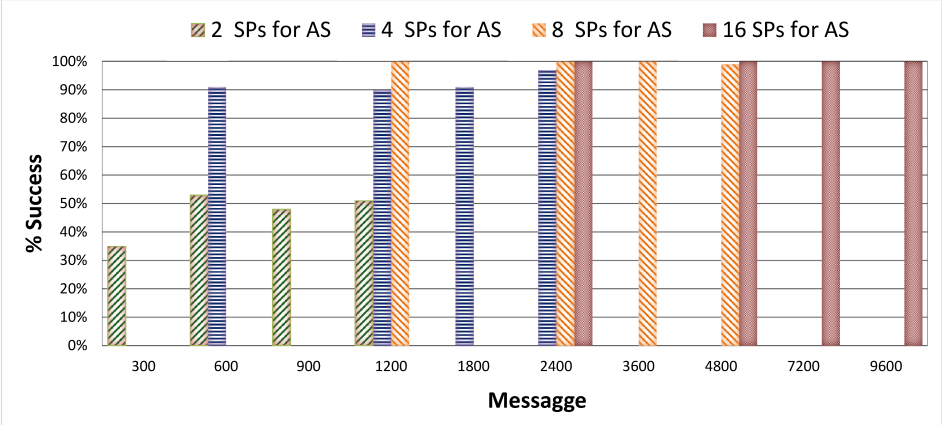

Figure 3.13: Success rate evaluated with respect to the number of messages.

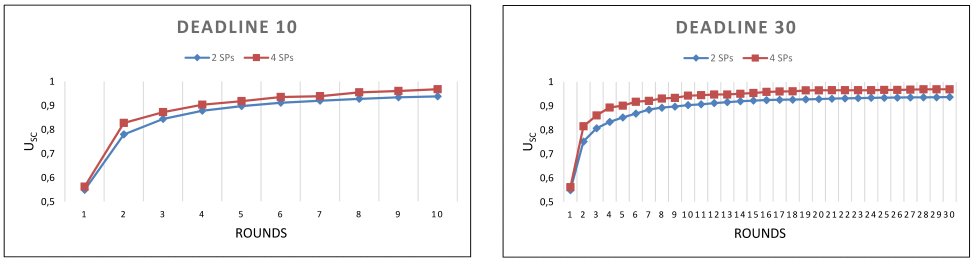

Figure 3.14: The SC utility for different configurations in case of negotiation failures.

Hypothesis-4 *Utility variation for the SC may suggest when to stop negotiating.*

As stated in Hypothesis 3, with a fixed number of negotiation rounds, communication overhead is introduced when the number of negotiation rounds increases (in the case the negotiation takes place for all the fixed number of rounds). Hence, it would be useful to extract information on the negotiation trends to decide when to stop negotiating. At this purpose we evaluated the variation of the SC global utility in case of failures. In particular, the number of SPs for each AS is 2 and 4, and the deadline is set to 10, 20, 30 and 40 rounds. Note that by increasing the number of SPs to more than 4 SPs, only successes are obtained from round 10th onward. In Figure 3.14 the SC utility is re-ported for the cases of deadline set to 10 and 30 varying the number of available SPs and the deadline of the negotiation. Trends for cases failures at 20 and 40 are similar. Results show that the SC utility varies very little by increasing the number of the negotiation rounds, so making proceeding with negotiation expensive without any benefit in the SC utility.

		Round Range		
		10/20	20/30	30/40
# SPs	2	0.0223	0.0082	0.0042
	4	0.0221	0.0043	0.0083
Average		0.0222	0.0063	0.0062
STD		0.0002	0.0028	0.0029

Table 3.4: SC utility variation in case of failures.

	$bestPrice_i$(\$)									
# ASs	1	2	3	4	5	6	7	8	9	10
2	1350	675	-	-	-	-	-	-	-	-
5	648	540	351	270	216	-	-	-	-	-
10	351	324	297	270	230	175	135	108	81	54

Table 3.5: Distribution of services costs varying the number of ASs.

In the Figure 3.14 the SC utility variation in case of failures is reported in order to allow the SC to dynamically stop the negotiation according to its trend, i.e., according to whether and in which measure its utility is varying. The variation is calculated as a difference between the value of the SC utility respectively at rounds 10 and 20 (for the first column), at rounds 20 and 30 (for the second column), at rounds 30 and 40 (for the third column). The variation is evaluated varying the number of SPs (2 and 4). The average SC utility variation with the corresponding standard deviation are summarized in Figure 3.14. Results show that in the configurations with the number of SPs equal to 2 and 4, by increasing the number of rounds the SC utility variation is less than 1% after 20 rounds, so indicating that keeping on negotiating is not likely to lead to a success.

3.5.1.5 Experimental Evaluation

In this paragraph we present the results of the experiments carried out to evaluate the communication costs of the negotiation process. In particular, we report the number of messages actually exchanged to reach a success, and not the ones theoretically calculated to reach the negotiation deadline, as in the previous experiments. The considered configurations are the same as the previous experiments, but including, for some of them, three different AWs composed respectively of 2, 5 or 10 ASs. Distributions of market prices ($bestPrice_i$) for the different ASs are shown in the Figure 3.5. Furthermore, 500 runs for each configuration are carried out.

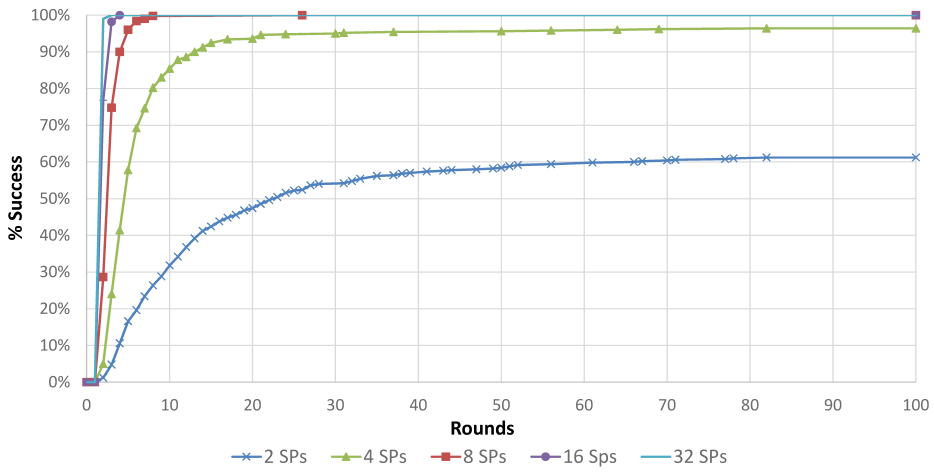

Figure 3.15: Success rate and number of rounds varying the number of SPs.

Hypothesis-5 *It is worth to increase the number of SPs if for the SC it is sufficient only to meet the global requirements.*

As stated in Hypothesis 2, by increasing the number of SPs and negotiation rounds the success rate increases, and also the number of combinations that satisfy the global requirements. This can be useful when the SC tries to maximize its own utility. Nevertheless, as stated in Hypothesis 3, it increases also the communication overhead, since the theoretical maximum number of exchanged messages increases. Here, we evaluate the success rate with respect to the number of rounds (see Figure 3.15), and with respect to the actual number of exchanged messages (see Figure 3.16), varying the number of SPs (at 2, 4, 8, 16, 32) in the case of 5 ASs, and fixing the number of negotiation rounds at 100, by keeping on negotiating for all the allowed rounds. The obtained experimental results show that, in the case of a relevant number of SPs, long negotiation mechanisms are not necessary in order to reach a success, i.e. if the SC wants to select just one combination of offers that meet the global requirement, so stopping negotiation as soon as the first feasible combination of offers is found, without any optimization. In fact, as shown in Figure 3.15 and in Figure 3.16, with number of SPs is greater than or equal to 8 a feasible combination is found quickly, so requiring few negotiation rounds, and hence less exchanged messages (no more than 1000 messages). Of course, in the case the best combination, in terms of the SC utility, has to be selected, negotiation should proceed for the allowed number of rounds so more feasible combinations can be found to optimize the SC utility.

Hypothesis-6 Trading off is possible.

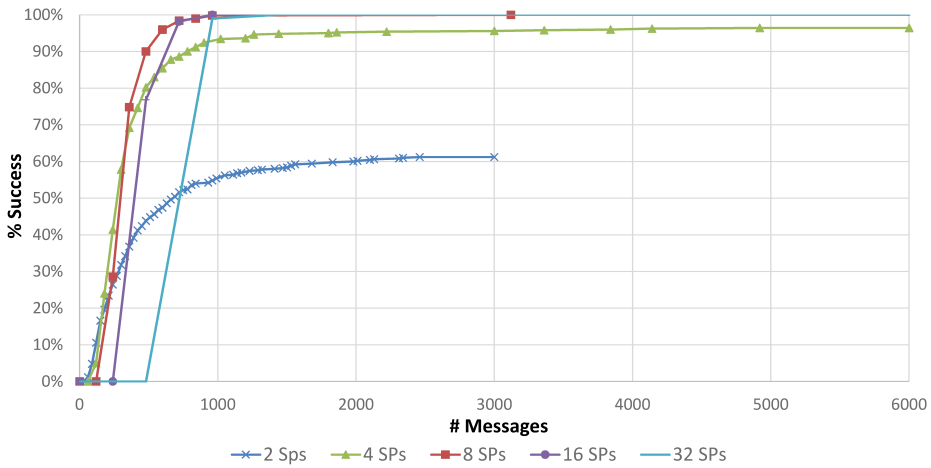

Figure 3.16: Success rate and number of messages (cut at 6000) varying the number of SPs.

We carried out experiments to evaluate the number of exchanged messages to reach a success (i.e. a feasible solution if any), and the percentage of successes by varying the number of ASs (to 2, 5, and 10), and the number of SPs for each AS (to 2, 4, 8, 16, 32) with a negotiation deadline at 100 rounds. In particular, we report the ratio between the average number of messages actually ex-changed and the percentage of success representing a measure of a "tradeoff" between the negotiation communication cost and the probability of succeeding in the negotiation. The smaller this value is, the better the configuration is for the negotiation. Figure 3.17 shows that a configuration with 8 SPs for the different number of ASs considered in the experiments, is an acceptable tradeoff between the negotiation communication cost and benefits (i.e. the percentage of successes). This result gives an indication on a reasonable number of SPs to negotiate with, so making it possible to limit the communication overhead in case the number of available SPs for each AS is high. Note that a high rate of failures resulting in keeping on negotiating until the deadline (so increasing considerably the number of exchanged messages) for the cases with 2 and 4 SPs is responsible for the high values of the standard deviation in the results.

Hypothesis 7 Increasing the number of ASs is not the same as increasing the number of SPs.

According to Hypothesis 2, by increasing the number of SPs, the percentage of successes increases. This is because in the stochastic market we modeled in this work, an increase in the number of negotiators corresponds to an increase in the variability of the market,

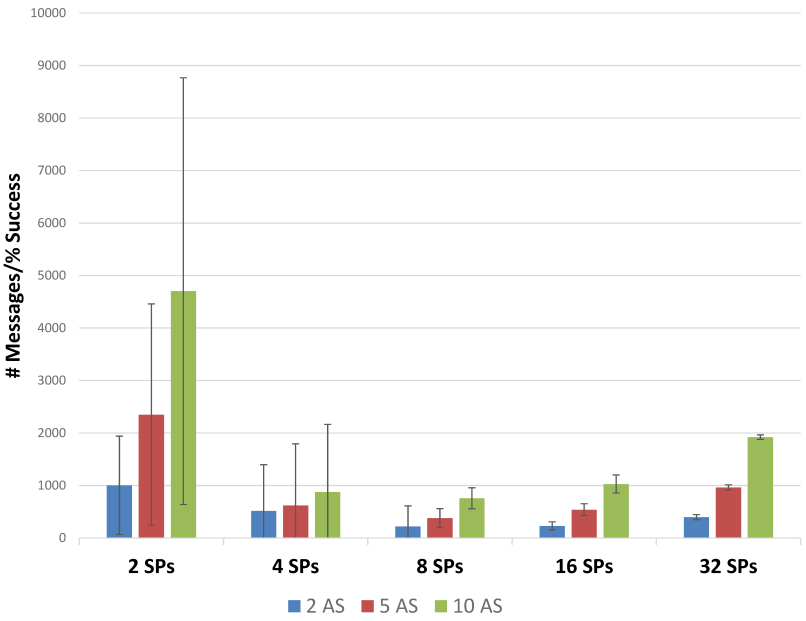

Figure 3.17: Tradeoff varying number of ASs and number of SPs.

that statistically results in improving the possibility of a negotiation success, and hence in a higher percentage of negotiation successes. In principle, the number of SPs involved in the negotiation increases also if the number of ASs increases. So, it could be concluded that increasing the number of ASs, may result in improving the negotiation success rate. However, this is not true in the case of service composition since the increase in the number of ASs consists in a more complex composition. This is confirmed by the results reported in Table 3.6 where the percentage of successes is evaluated with respect to the number of ASs, the average number of negotiation rounds necessary to reach a negotiation success (together with the associated standard deviation), and the average number of exchanged messages (together with the associated standard deviation). Recall that high values for standard deviation are due to the number of failures occurring during negotiation. As shown in Table 3.6, with the same total number of SPs, different percentage of successes are obtained for different AW configurations. In fact, considering the same number of SPs involved in the negotiation (e.g. 20), the success percentage is 62% in the case of 10 ASs and 2 SPs for each AS, and it is 96% in the case of 5 ASs and 4 SPs for each AS.

The experimental results showed that, in cases of negotiation failures, after a certain number of rounds no gain is obtained in the success rate. Hence, this information could be used to predict the negotiation outcome in order to decide to stop negotiation without

# SPs	# AS	% Success	# Round		# Message	
			Average	ST. Dev.	Average	ST. Dev.
2	2	58%	48	45	580	541
	5	61%	48	43	1438	1291
	10	62%	49	42	2925	2528
4	2	87%	19	32	452	768
	5	96%	10	19	599	1130
	10	99%	7	11	871	1276
8	2	99%	5	8	220	387
	5	100%	3.2	1.5	380	179
	10	100%	3.1	0.8	757	200
16	2	100%	2.4	0.8	231	77
	5	100%	2.2	0.5	540	113
	10	100%	2.1	0.3	1029	171
32	2	100%	2.0	0.3	397	49
	5	100%	2.0	0.1	965	48
	10	100%	2.0	0.04	1922	43

Table 3.6: Experiments summary.

any loss in terms of consumer's utility (limiting the cost of negotiation in terms of its length). Of course, these results are related to the considered configurations, and also to the specific strategies adopted for the SPs.

3.5.2 Composition vs. Orchestrated Negotiation

Here, we carried out another set of experiments to numerically evaluate the trends of negotiation in the case of multiplicative and additive QoS issue in service composition. We also compare the trends of negotiation when the composition problem is represented in terms of the provision of a single virtually composed service.

The attributes of the composed service have been composed, according to the aggregation function, that is the composition of utility functions of the components issues, and are offered following this function.

The aim of this numerical analysis is to show that the mathematical equivalence of the Gaussian distributions is also reflected in the obtained experimental negotiation trends. These trends are measured in terms of the:

- percentage of successes obtained during the negotiation (%succ),

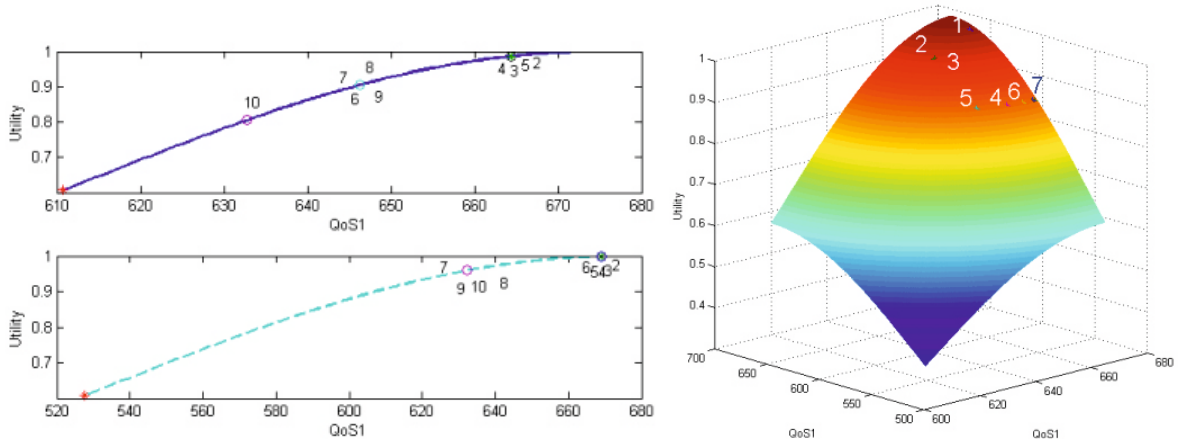

Figure 3.18: Probability distributions for a single–issue (reliability) negotiation with two SPs (left) and an example of the aggregated probabilities (right).

- number of negotiation rounds necessary to reach that success percentage (#rounds),
- number of the corresponding exchanged messages (#mess).

These negotiation parameters are computed for an SBA request composed of 2 ASs, by varying the number of SPs (from 1 to 4) and the number of allowed negotiation rounds (from 10 to 30). The reported values are mediated on a number of 100 experiments for each considered configuration.

We first considered the case of multiplicative QoS values, such as the reliability value of the complete application that is given by the multiplication of the reliability values of its component services. For simplicity, in this case, we considered only 1 SP for each AS. The SPs probability distributions, for one negotiation run, are plotted in Figure 3.18 (left), together with the offers generated at different negotiation rounds. The trends of the negotiation are compared with the same problem represented by considering a single provider whose utility function is the product of the utility functions of the two providers (one for each AS), so representing the distribution of the composed QoS values offered for the virtually composed service. An example of such distribution is reported in Figure 3.18 (right), together with the composed offers generated at different negotiation rounds. As discussed in Subsection 3.3.3 , in multi–issue negotiation trade–off among issues is possible. In fact, as shown in Figure 3.18 (right), in the considered negotiation run, an agreement is reached at round 7, with the last offer lying on the same indifference curve of the previous three ones.

reliability						
	% succ		# rounds		# mess	
	1 SP - 2 AS	1 SP_c	1 SP - 2 AS	1 SP_c	1 SP - 2 AS	1 SP_c
10 round	0.83 ± 0.40	0.88 ± 0.33	5.1	4.6	10	9
20 round	0.84 ± 0.37	0.97 ± 0.17	7.6	5.6	15	11
30 round	0.91 ± 0.29	0.99 ± 0.10	8.5	5.2	17	10

Table 3.7: Negotiation trends values for multiplicative issues.

As shown in Table 3.7, the percentages of successes for the two problem configurations are comparable within the error, with better values in the case of the multi–issue case, as expected. This behavior may be due to the fact that the adopted concession strategy for the composed offers (i.e., a marginal concession and a trade–off) is not the same as the one adopted for the two individual offers. In addition, in the multi–issue case, the number of offers with the same probability to be offered increases, so leading to an increased probability of success that becomes more evident by increasing the number of negotiation rounds. Accordingly, also the number of rounds necessary to reach success is lower in the multi–issue case. Finally, since in the latter case negotiation occurs only with one SP, the number of exchanged messages is lower that the single–issue negotiation.

In the second set of experiments, we considered the case of additive QoS values, such as the price value of the complete application that is given by the sum of the price values of its component services. Also in this case, we evaluate if the problem of finding n services that, once combined, provide an overall QoS acceptable value, is the same as the problem of finding 1 service providing n QoS values composing an overall acceptable QoS value. Hence, the trends of the single–issue negotiation are compared with the ones of the same problem represented by considering a single service provided by SPs whose utility functions are the convolutions of the utility functions of two providers (one for each AS). In order to have the same negotiation configuration in both cases, as reported in Subsection 3.3.3, we have to consider all the possible combinations of the two component QoS values, independently provided for the case of two services. So when SPs vary from 1 to 4 for each AS, the number of SPs in the single service case, varies from 1^2 to 4^2, where each SP has an utility function obtained by the convolution of the two utility functions of two SPs of the component services. Since the convolution of two mono–dimensional Gaussian function is still a mono–dimensional Gaussian function, the utility function used to generate the virtually composed QoS values is mono–dimensional.

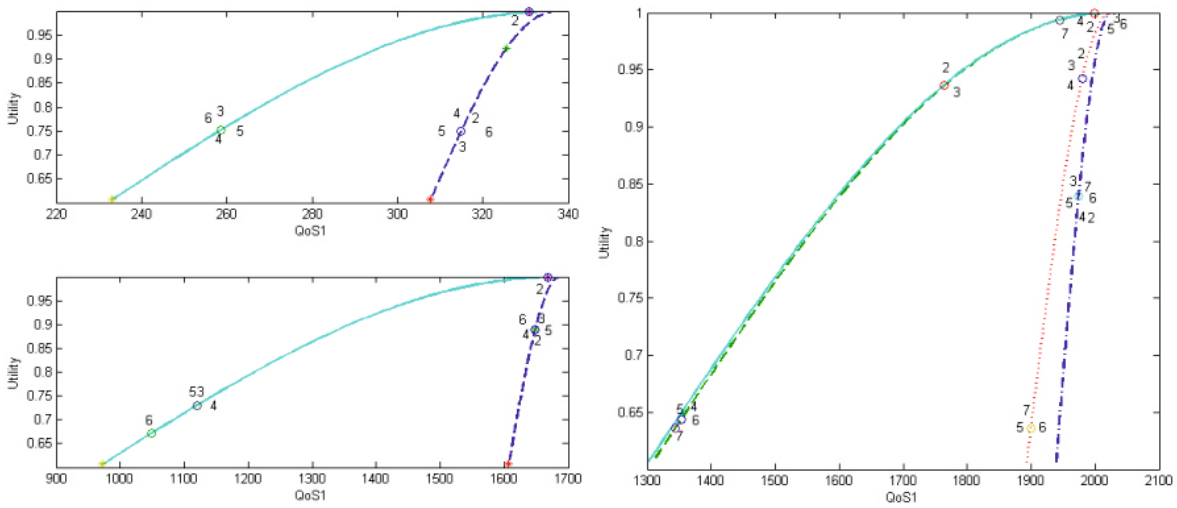

Figure 3.19: Probability distributions for a single–issue (price) negotiation with two SPs (left) and an example of the 4 aggregated probabilities (right).

The probability distributions of two SPs (one for each AS), and the corresponding convoluted distribution together with the offers generated in both cases at different rounds, for a negotiation run, are reported respectively in Figure 3.19 (left) and (right). As shown in Table 3.8, the negotiation trends in terms of percentage of successes are comparable within the error. This is confirmed by the number of negotiation rounds for both experiments that are the same. It should be noted that the end–to–end constraints used in these experiments is set in such a way that it is difficult to reach an agreement also by increasing the number of SPs. But with this setting, a high number of negotiation failures occurs so causing a big fluctuation of the collected results, and hence an increased value of the standard deviation. Nevertheless, the number of messages exchanged during negotiation increases exponentially with the number of SPs available for each AS. In fact, in order for the two problem representations to be equivalent the number of SPs for the composed QoS increases exponentially with the number of ASs.

We showed that by modelling non linear utility functions, as well as concession strategies through the use of normal distributions, allows to deal with computational tractability requirements need by real market of services, due to their properties in scaling up in dimensions and convolutions. More specifically, with the adopted strategies, negotiation in SBAs is shown to be inherently multi–issue even in the case of a single–issue negotiation, where the multiple dimensions are given by the requirement of composing the QoS provided by the different component services to meet the end–to–end constraint. In fact,

price						
	% success		# rounds		# messages	
	1 SP - 2 AS	1 SP_c	1 SP - 2 AS	1 SP_c	1 SP - 2 AS	1 SP_c
10 round	0.13 ± 0.34	0.10 ± 0.30	9.6	9.6	19	19
20 round	0.21 ± 0.41	0.11 ± 0.31	17.6	18.7	35	37
30 round	0.23 ± 0.42	0.15 ± 0.36	25.7	27.3	51	55
	2 SP - 2 AS	4 SP_c	2 SP - 2 AS	4 SP_c	2 SP - 2 AS	4 SP_c
10 round	0.34 ± 0.48	0.24 ± 0.43	8.5	8.9	34	71
20 round	0.38 ± 0.50	0.31 ± 0.46	14.7	16.2	59	129
30 round	0.43 ± 0.50	0.36 ± 0.44	21.5	25.0	86	200
	3 SP - 2 AS	9 SP_c	3 SP - 2 AS	9 SP_c	3 SP - 2 AS	9 SP_c
10 round	0.43 ± 0.50	0.35 ± 0.48	8.0	8.4	48	152
20 round	0.65 ± 0.48	0.54 ± 0.50	12.0	12.4	72	223
30 round	0.70 ± 0.46	0.61 ± 0.50	15.8	18.6	95	335
	4 SP - 2 AS	16 SP_c	4 SP - 2 AS	16 SP_c	4 SP - 2 AS	16 SP_c
10 round	0.61 ± 0.49	0.57 ± 0.50	7.1	6.7	57	215
20 round	0.74 ± 0.44	0.60 ± 0.49	10.6	11.7	85	374
30 round	0.78 ± 0.42	0.64 ± 0.48	14.5	15.2	112	488

Table 3.8: Negotiation trends values for additive issues.

Gaussian–based utility functions used to model providers of n different services with a single QoS negotiating for reaching a successful composed QoS value, can be mapped to Gaussian–based utility functions that model providers of a single service with an already composed QoS value. The expected behavior of the negotiation trends in different but equivalent representations of a service composition problem, is supported by the reported numerical analysis.

In conclusion, the adoption of Gaussian–based strategies in negotiation for service composition allows to use the same negotiation mechanism in terms of protocols, strategies and utilities for both single issue and multi–issue negotiation, and they are easy to implement. Furthermore, the negotiation in service composition can be represented by different problem configurations that are equivalents due to the properties of the adopted Gaussian functions. These configurations can be used to carry out service composition simulations, and to evaluate which negotiation configurations to use in order to improve negotiation outcomes.

Chapter 4

Negotiation for Decision Making

In this chapter, we present the smart parking application developed using a negotiation mechanism between software agents, allowing to automate the selection of parking spaces according to the user preferences and taking into account the city needs.

The application has been developed in the context of the research project S^2-Move a PON funded by the MIUR on Smart City and Social Innovation. The goal of this project is to provide ICT-based solutions for the monitoring and the real-time management of urban mobility, providing information services for motorists interactive through the integration of ICT and the exchange of information between individual users. For this reason, it was developed an intelligent platform that allows data collection, analysis and processing, providing intelligent and efficient solutions to the citizens. In this scenario, the user becomes part of the platform because it provides information that is at the base of the operation of the services made available by platform, in order to return solutions for "easy and sustainable" mobility.

The type of information collected is heterogeneous, retrieved from different devices (e.g. smartphones and tablets) and from hardware systems embedded directly on board the cars. The use of such systems on private vehicles and public utilities (e.g. public transport, vehicles of the police or the health service) allows to gather updated information and the status of urban mobility.

The services developed on platform of project, that are provided on through the web, mobile apps and embedded devices on-board, are the follows:

- **Traffic Monitoring** that estimates the traffic in the city, managing the information provided by cars monitored;

- **Fleet Management** that allows the monitoring and management of vehicle fleets;
- **Warning Management** that allows users to report faults and warnings present in the urban area, such as the presence of traffic, potholes and disadvantageous conditions for mobility.
- **Smart Parking System** provides a service to visualize, choose and book a parking spot.

  In particular, the research carried out for developing Smart Parking Services service will be described in detail in the following sections.

4.1 Smart Parking Application Scenario

Urban transportation is considered a relevant investigation area for the innovation of Smart Cities since it may contribute to increase the quality of life of city-dwellers, to enhance the efficiency and competitiveness of the city economy, and to move towards the sustainability of cities by improving resource efficiency and meeting emission reduction targets. The main themes addressed in urban transportation for Smart Cities are:

- *Cooperative Intelligent Transport Systems and Services (C-ITS)*, based on the principle that all cooperative parties (i.e. ITS stations, vehicles, road side units) exchange information between each other, so enabling up-to-date traffic information, improved road safety and traffic efficiency.
- *Enabling Seamless Multi-modality for End Users*, based on the possibility to combine public transport with other motorized and non-motorized modes as well as with new concepts of vehicle ownership.
- *Smart Organization of Traffic Flows and Logistics* that involves multi-agency interaction, linking individual mobility with public transport services.

In this framework, one of the challenging problems linked to the above themes, is parking in urban areas. It is widely recognized that drivers searching for a parking space in wide urban areas waste time and fuel, so increasing traffic congestion and air pollution [55]. It is not always possible to address the problem by creating more parking spaces, but rather "intelligent" parking facilities are necessary.

The use of advanced technologies, including vehicle sensors, wireless communications, and data analytic, is the base for the efficient allocation, monitoring, and management of smart parking solutions for future Smart Cities in order to improve urban mobility strategies. Most of the research projects concerning smart parking systems focus on ways to collect and publish live parking information to drivers so they can be informed of available parking spaces near the destination they require. At the same time, many companies are developing electronic parking systems allowing for a wide variety of available payment methods in conjunction with the dissemination of parking availability information. Nevertheless, they lack of intelligent features allowing not only to advise motorists of available car parks in multiple zones, but more importantly to help them in making decisions on where to park.

Mechanisms to manage the relationship between supply and demand are necessary to provide user-oriented automatic parking services that take into account both drivers preferences, and parking vendors requirements together with social benefits for the city, such as a reduction of traffic in city centres by limiting parking in that area [64].

Usually, parking applications provide users with available parking spaces among which to select the preferred one according to their own preferences, if possible. In the Smart Cities of the future, users should be equipped with applications able to carry out this selection automatically, and more importantly, to take into account different requirements for a parking space based on user profiles (e.g. business, tourist, generic) that may have different preferences on parking attributes. Furthermore, in order to help refining the selection process, additional information may be used (that could come from other sources of information), such as unavailability of public transportation at the required time, the necessity to reach different locations once the car has been parked, the possibility to find other attractions in the area, and so on.

Another problem of parking in big cities is the fragmentation of public and private parking providers, each one adopting their own technology to collect occupancy data that, as such, cannot be easily shared among different owners or made accessible by user-friendly applications. In order to provide motorists with smart parking applications, the first step would be to encourage public and private parking providers to share their data and to build smart parking software applications that coordinate individual parking solutions for end users without involving them in the fragmentation of parking owners. At the same time, individual parking owners should be made aware of the benefits of providing

such a global parking provision showing them that the coordinated provision of parking solutions still guarantees their individual income and fair competition by better exploiting the parking spaces offered in a city. Furthermore, a coordinated parking system allows to gather information to dynamically change the price of the offered parking spaces according to market-based evaluations based on the flow of user requests and the occupancy of the car parks in a given time interval (e.g., the price could decrease according to the occupancy of the parking, or to the time requested by the user), their geographical location, and so on.

In this context, we investigate the possibility to use software agent negotiation to address some of the challenges concerning smart parking and mobility pricing strategies. In particular, we propose an automated negotiation mechanism among a software agent that models a Parking Manager (PM) responsible for providing parking spaces, and a software agent User Agent (UA) acting on behalf of a motorist searching for a parking space in the specific city area. Negotiation is used in order to accommodate both users and providers needs that are different and, more importantly, conflicting. In fact, the Parking Manager has the objective to sell parking spaces to make a profit, but to prevent, as much as possible, motorists to park in the specific city area, while users would prefer to save as much money as possible, but to park close to the city center location they require. The allocation of the parking space is the result of a negotiation process between the Parking Manager and the user having their own private utility functions respectively to make a parking space offer, and to evaluate whether to accept a received offer. Different user profiles may be modeled by using different utility functions to evaluate parking offers.

It is assumed that car park owners (that can be both public and private) agree to subscribe to a Coordinated Parking System by making it available a given number of parking spaces managed by a Parking Manager agent (PM). It is responsible for their coordinated reselling to provide a better distribution of vehicles in the managed car parks. Its objective is to sell parking spaces to make a profit, but also to prevent, as much as possible, motorists to park in the red zones e.g city center, so improving the city life by decreasing the circulation of cars in the high traffic areas. Motorists are modeled as User Agents interacting with the PM to submit requests for parking spaces specifying their own preferences on where to park, but also trying to pay as little as possible. Automated negotiation between the PM and the UA is used to find a parking space allocation that accommodates their needs up to a certain extent, i.e. by finding an acceptable compromise

for the involved negotiators.

The length of the negotiation process could prevent its use in real-world scenarios, so we adopt a flexible negotiation mechanism, proposed in [15], that allows to dynamically set the negotiation duration

4.1.1 The Negotiation Model

The adopted negotiation mechanism, reported in [15], is used in the present work as a bi-lateral negotiation whose protocol is based on a Contract Net Iterated Protocol, and it may be iterated for a variable number of times until a deadline is reached or the negotiation is successful. Each iteration is referred to as a negotiation *round*, and the deadline is the number of allowed rounds.

According to the protocol, at the first negotiation round the UA submits its request for a parking space specifying the preferred location area in the city area, and the requested time interval. The PM replies sending an offer for a parking space, waiting for an acceptance or rejection from the UA. If the offer is accepted the negotiation ends successfully, otherwise a new round is started, if allowed by the protocol. The PM will send as many offers as the number of allowed rounds, that of course cannot be greater than the number of available car parks.

In the proposed negotiation, utility functions are used to model the different needs of the PM and the UA: the PM uses the value of the utility function to decide which offer to send, while the UA uses the utility function to evaluate whether to accept or to reject the received offer. The issues for the PM are the car park availability (`availability`) and its distance from the center of the red zone (`distance_from_red_zones_center`) area to avoid selected by PM, while the issues for the UA are the parking space price (`price`), the distance of the car park from the requested location (`GPS_distance`), and the same distance evaluated in terms of travel time from the requested location (`time_distance`). So, the utility functions for the PM and the UA have the following domains:

$$U_{PM}(offer_{PM}(k)) : \texttt{availability} \times \texttt{distance\_from\_red\_zone\_center} \rightarrow [0,1]$$
$$U_{UA}(offer_{PM}(k)) : \texttt{price} \times \texttt{GPS\_distance} \times \texttt{time\_distance} \rightarrow [0,1]$$

where, the co-domain $[0,1]$ indicates that the functions are normalized.

Utility functions are modeled as linear functions (as will be explained in the following

subsections) resulting from the weighted sum of the considered issues. Different *weights* can be associated with the considered parking attributes, so modeling the different importance of the attributes for different classes of users, and even for different Parking Managers. It should be noted that an offer proposed by the PM in a negotiation round cannot be considered available in the successive rounds once rejected by the UA, since it may be allocated to a different user, or its price may change according to the number of requests.

4.1.2 The User Agent request

It is assumed that motorists interact with a Coordinated Parking System, by submitting a request for a parking space to the Car Park Server through several devices (e.g. Tablet, Smart-Phone, PDA or PC), using a city map to select the area where he/she would like to park, and an interface to indicate his/her parking preferences. The Parking Manager is responsible for processing the request: it queries an internal database to retrieve information on the available car parks, and it relies on specific applications to extract car park availability when the request is processed, and to collect relevant information on city regulations, or on events that may affect public transportation.

Each car park is characterized by the following parameters:

$$\texttt{car\_park= <park\_id, park\_GPS\_location, ref\_price\_unit, park\_capacity, sector>}$$

where `park_id` is the unique identifier of the car park, `park_GPS_location` is its GPS location, `ref_price_unit` is the default time unit price for a parking space, `park_capacity` is the total number of parking spaces in the car park, and `sector` represents the geographical location of the car park with respect to the `red zone`. A `sector` identifies a ring and its value is an integer.

The distribution of sectors starting from the center of a red zone is shown, in Fig. 4.1 and it is used to model the relation of the price offered for a parking space on the distance between the car park and the centre of the red zone.

A User Agent request, `park_req`, is composed of values referred to the parking space attributes that are relevant for the user to decide where to park:

Figure 4.1: Representation of city sectors.

$$\texttt{park\_req= <id\_req, dest\_GPS\_location, start\_time, end\_time>}$$

where `id_req` is the unique identifier of the user request, `dest_GPS_location` represents the GPS location of the destination the user wants to reach, the interval (`end_time` - `start_time`) represents the time the user wants to park for.

With a static selection, the PM will select car parks considering only to meet the user requirements in terms of location, and available parking spaces for the required time interval. If there is no parking space meeting the requirements, a static mechanism will end up with no solutions for the driver request.

The proposed mechanism allows the evaluation of criteria that may not be explicitly expressed by the user, and that can influence both the selection of parking spaces offered by the PM, and the evaluation of the received offer. In fact, By using an automated negotiation mechanism for the selection of parking spaces, it is possible to propose offers that do not strictly meet the user requirements, but that are a result of an evaluation of the available parking spaces against parking space attributes that are relevant to the PM, and whose values may depend on dynamic information, take into account time-dependent information such as car park occupancy, or special events, on the lack of public transportation in a given time windows and so on. On the other hand, a received offer is evaluated by the UA against parking space attributes that are relevant to the UA and whose importance may vary for different users.

4.1.3 The Parking Manager Model

As described, the proposed negotiation mechanism is not based on the exchange of offers and counteroffers, since UA may only accept or reject offers. So, the PM may compute the set of offers it will propose during negotiation, at the first round. The set of possible offers is computed by selecting first a set of car parks that meet the following requirements:

- the distance (referred to as `park_GPS_distance`) of the car park location (`park_GPS_location`) from the destination (`dest_GPS_location`) set by the user, is within a given distance (referred to as the `location_tolerance`),

- the car park have spaces available for the time interval specified by the user at the time `t` the request is issued.

The `location_tolerance` is set by the PM in such a way to include also car parks that are not in a red zone, and consequently they may be far from the `dest_GPS_location` specified by the user, since the PM tries to prevent users from parking in the red zones and to maximize the occupancy of car parks not located in a red zones.

In order to push users to park outside the red zones and in car parks with more parking spaces available, a dynamic cost model is associated to the Coordinated Parking System. The PM calculates the unit price to offer for a parking space by considering that car parks located in the red zones are more expensive. In fact, according to the distribution reported in Figure 4.1, the area around the red zone is divided in *sectors*, that account for the distance between a car park and the specific red zone. The price associated to parking spaces depends on the sector the corresponding car park belongs to, so the farther the car park is from the red zone, the cheaper it is. In addition, in order to incentivize the occupancy of less crowded car parks, a discount factor can be applied to each car park in accordance to its occupancy with respect to its total capacity.

The PM includes in the offer also the distance `GPS_distance` and time `time_distance` necessary to travel from car park offered to the destination. `time_distance` is evaluated as travel time by using public or other means of transportation (`time_distance`). Also information is assumed to be retrieved with the support of external services. So, an offer of the PM is:

$$\texttt{offer(k)} = < \texttt{park\_id}, \texttt{GPS\_distance}, \texttt{dest\_time\_distance}, \texttt{park\_price\_unit} >$$

Once the PM computes the set of possible offers, it needs to establish which one to offer at each negotiation round, i.e. it needs to establish its concession strategy during negotiation. In order to do so, the PM uses a private utility function to rank the selected car parks. The evaluation function used by the PM to compute the utility of each car parking ($offer_{PM}(k)$) is the following:

$$U_{PM}(offer_{PM}(k)) = \sum_{i=1}^{n}(\alpha_i * \frac{q_{i,k} - \min_j(q_{i,j})}{\max_j(q_{i,j}) - \min_j(q_{i,j})}) \tag{4.1}$$

where n is the number of issues the agent is evaluating, $q_{i,k}$ is the value of the i-th issue of the k-th car park, $\min_j(q_{i,j})$ and $\max_j(q_{i,j})$ are respectively the minimum and the maximum values of the i-th issue among all the car parks selected by the PM, and the constants α_i are weights associates to the different issues with $\sum_{i=1}^{n}\alpha_i = 1$. As previously described, the issues for the PM are:

$q_1 = \texttt{availability}$
$q_2 = \texttt{distance\_from\_red\_zone\_center}$

Once the set of offers is ordered according to the utility values of Equation 4.1, the PM sends as first offer the one with the highest utility value, and it concedes in utility offering, at each negotiation round, parking spaces with a monotonically decreasing value of its own utility. The PM will end the negotiation with a failure if all the car parks selected have been offered and not accepted. If an offer is accepted by the UA, then the negotiation ends successfully.

4.1.4 The User Agent Model

The evaluation function used by the UA to compute the utility of each offer proposed by PM is the following:

$$U_{UA}(offer_{PM}(k)) = \left[1 - \sum_{i=1}^{m} \beta_i * \frac{q_{i,k} - c_i}{h_i - c_i}\right] \tag{4.2}$$

where, m is the number of issues the agent is evaluating, $q_{i,k}$ the value i-th issue of the k-th offer, c_i is the preferred value over the i-th issue, h_i are constant values introduced for normalizing each term of the formula into the set [0,1], and β_i are weights associates to the different issues with $\sum_{i=1}^{m} \beta_i = 1$. The weights are used to model different type of drivers:

- **business**, i.e. drivers that consider very important the location of the parking space with respect to the destination they need to reach, also being available to spend more money to get it ($\beta_1 < \beta_2$),

- **tourist**, i.e. drivers that are available to choose a parking space not so close to their preferred destination, provided that they can save money ($\beta_1 > \beta_2$).

The UA strategy is to accept an offer if its utility value is above a *threshold value* (UA_{att}) representing a measure of its attitude to be flexible on its preferred values for the considered parking space attributes. Since the utility function is normalized, its values may range in the interval $[0, 1]$. It should be noted that at each negotiation iteration, the UA utility varies according to the received offer, so it is not monotonic as the PM one.

Moreover, we assume that the preferred c_i values are reasonable with respect to each considered issue, i. e. the preferred user values are not unreasonable in relation to the issue (this means that the user cannot ask for a parking space for free!). If $q_{i,k} - c_i < 0$ than the term is set to zero. As previously described, the issues for the UA are:

$q_1 =$ `price`
$q_2 =$ `GPS_distance`
$q_3 =$ `time_distance`

At each round, the UA calculates its utility for the received offer according to Equation 4.2, and it accepts it only if the utility value is greater than the predefined threshold. Otherwise, it rejects the offer and waits for another offer, or for a message of negotiation end.

4.2 A Social-Aware Parking Space Selection

We propose to find an allocation of parking spaces that is viable from a global social benefit point of view. In many decision-making situations in transportation, the competitive alternatives and their characteristics are reasonably well known in advance to the decision makers (driver). On the other hand, drivers usually discover different parking alternatives one by one in a temporal sequence. Clearly, this temporal sequence has a very strong influence on the driver's final decision about the parking space. In our research, the Parking Manager selects a set of car parks belonging to the Coordinated Parking System, but the temporal sequence in which they are offered during negotiation privileges first car parks meeting also city needs requirements.

The goal of the negotiation between the User Agent and the Parking Manager is to select a parking space that represents a viable compromise between the driver's and the city needs (represented by the Parking Manager preferences), so reaching a sort of *utilitarian social welfare*. The concept of social welfare, as studied in welfare economics, is an attempt to characterize the well-being of a society in relation to the welfare of its individual members [7]. The proposed negotiation mechanism relies on the utilitarian interpretation of the concept of social welfare in multi agent systems literature, i.e. whatever increases the average welfare of the agents of a society is considered beneficial for the society as well.

4.3 A Prototype Implementation

In order to provide Smart Cities with an intelligent smart parking solution to be integrated in a more complex Coordinated Parking System, we designed and implemented a web-based multi-agent application to automatically select parking spaces in reply to user's requests. The application was tested in a case study based on both real and simulated information, to assess the suitability of software agent negotiation in the context of intelligent parking. The architecture of the implemented prototype is shown in Figure 4.2 reporting its main components.

The negotiation module is implemented by using the JADE framework [9] to implement the UA and the PM, and relying on its communication primitives to implement the adopted negotiation protocol. JADE is an open source software framework for developing

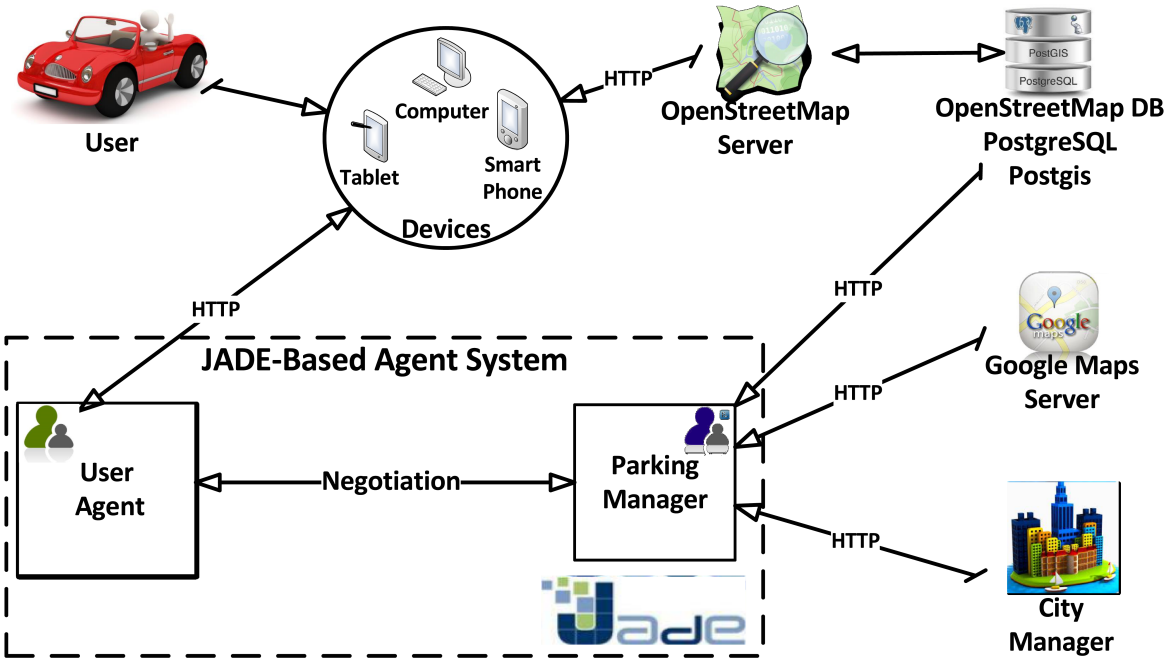

Figure 4.2: The prototype architecture of the smart parking application.

applications that implement agent and multi-agent systems. It is a Java based agent development environment providing libraries designed to support communication between agents in compliance with Foundation for Intelligent Physical Agents (FIPA) specifications. The multi-agent system is composed of the UAs and the PM. The PM is enveloped in an application server, more specifically on Apache Web Server extended with Tomcat, and it is able to communicate with external services and information sources:

- Google Map Server [51] to retrieve walking distance and travel time from a selected car park to the user's destination location,
- the Car Park Database to retrieve information on the available car parks,
- City Manager facilities to retrieve information regarding roads accessibility-related information.

The contents of the Car Park Database are retrieved from the OpenStreetMap application [29], that is implemented using PostgreSQL, an object-relational database management system, and PostGIS an open source software providing support for geographic objects to

the PostgreSQL database. The user's application also queries OpenStreetMap to obtain maps for the interface it interacts with.

4.4 Experimentation with Smart Parking Application

In order to assess how software agent negotiation can be used in the selection of parking spaces in an urban area, we carried out several experiments. In this section we reported some of this experiments considering the city of Naples as the target area. In subsection 4.4.1 we investigate the use of software agents negotiation allows to assign parking spaces in an automatic and intelligent manner by taking into account that users needs and parking vendors needs regarding efficient allocation of parking spaces, and city regulations. In particular in subsection 4.4.2 we evaluate the benefit in using negotiation to select parking solutions that represent a compromise between the needs of both the PM and the UA, comparing with the parking spaces chosen respectively by the PM and the UA in the case of complete information sharing among agents, and of no shared information. Finally, in subsection 4.4.3 we evaluate with different metrics the social benefit of the parking allocation in terms of both agents utilities, and allocation efficiency.

4.4.1 Smart parking allocation

In this subsection, we firstly report a set of experiments carried out in order to determine whether the negotiation is a viable approach in order to meet both users and parking managers requirements.

The experiments simulate 150 different queries made by users by selecting a destination on the interactive map of the city centre of Naples provided by the Coordinated Parking System, and associating to the destination the time interval which the user wants to park for. In this case, the city center represents the red zone. The destinations selected by the users are located in sectors 1 and 2 on the city map shown in Figure 4.3. For each query a negotiation run takes place. At the first negotiation round, the PM selects the car parks according to the query as reported in subsection 4.1.3. Parking identifiers and locations are extracted from the OpenStreetMap database of the city of Naples, while routing information (`dest_GPS_distance` and `dest_time_distance`) are retrieved through

Figure 4.3: Reference scenario.

the use of Google MAPs API. The occupancy of car parks is randomly generated for each negotiation run.

The weights in the utility functions are equally distributed among issues ($\alpha_i = 0.5$ and $\beta_i = 0.33$ for all i), while, for each issue i, h_i and c_i are dynamically set respectively to $max_j(q_{i,j})$ and $med_j(q_{i,j})$ (i.e., the maximum and the medium value for the current issue). The UA accepts an offer if its utility for that offer is greater than a threshold value set to 0.6 for the experiments.

4.4.1.1 Experimental Results

The first experimental results are summarized in Table 4.1 in case of successful negotiations. In particular, the table reports the maximum, the minimum and the mean value (with the standard deviation), obtained at the end of each negotiation run, of the number of selected car parks (# available parks), the number of negotiation rounds (# Rounds), the parking spaces available in the car park (Availability), the distance between the selected car park and the city center (Distance), the distance between the selected car park and the user's destination (Route), the parking space unit price (Price), the travel time to reach the destination from the car park (Time), the PM utility (PM Utility), and the UA utility (UA Utility).

The mean value of rounds (that is the number of offers sent by the PM) is very low with respect to the mean number of car parks selected by PM for the experiments. This means that the negotiation ends before the PM offers all the selected car parks, and the

	max_value	**min_value**	**mean_value**
# Available parks	14	10	11 ± 2
# Rounds	9	1	3.3 ± 2.5
Availability	237	1	110 ± 58
Distance (m)	7339	1948	3495 ± 360
Route (m)	4355	649	1105 ± 160
Price (u_p)	8.9	5.1	7.6 ± 0.3
Time (s)	3046	457	927 ± 211
PM Utility	0.97	0.03	0.62 ± 0.22
UA Utility	0.75	0.10	0.68 ± 0.06
PM Utility without Neg	–	–	0.35 ± 0.27
UA Utility without Neg	–	–	0.71 ± 0.04

Table 4.1: Experimental Data collected in 150 runs.

obtained mean utility values for the UA and PM show that the requirements of both parties can be met in a satisfactory way.

With the same settings we evaluated the PM and the UA mean value utilities obtained in the case the complete set of offers selected by the PM is known to the UA as well reported in the last two rows in Table 4.1. Figure 4.4 reports a graphical representation of the different utility values respectively for the PM and the UA on the interactive city map. In this case the UA would select the offer that maximizes its own utility (in the average 0.71), that corresponds to a low utility for the PM (in the average 0.35). As expected, in this way, the UA requirements are privileged with respect to the PM ones.

In Table 4.2 experimental results are reported for two negotiation runs with the same query, but varying the occupancy of the selected car parks. The Table reports the values of the issues of each offer for both the PM and UA, and their utilities. According to the negotiation mechanism, at each negotiation round, the PM selects the offer with the best utility value, among the remaining offers. The negotiation ends as soon as the UA utility for an offer is greater than its threshold value. As shown in Table 4.2, varying the occupancy of the selected car parks impacts the length of the negotiation (i.e., the number of rounds necessary to reach an agreement).

These experimental set, we show that an automated negotiation mechanism between the Parking Manager and motorists represented by User Agents, allows to find a compromise, through the use of utility functions for the involved negotiators that manage different needs to be dynamically evaluated, and help users in their decision making pro-

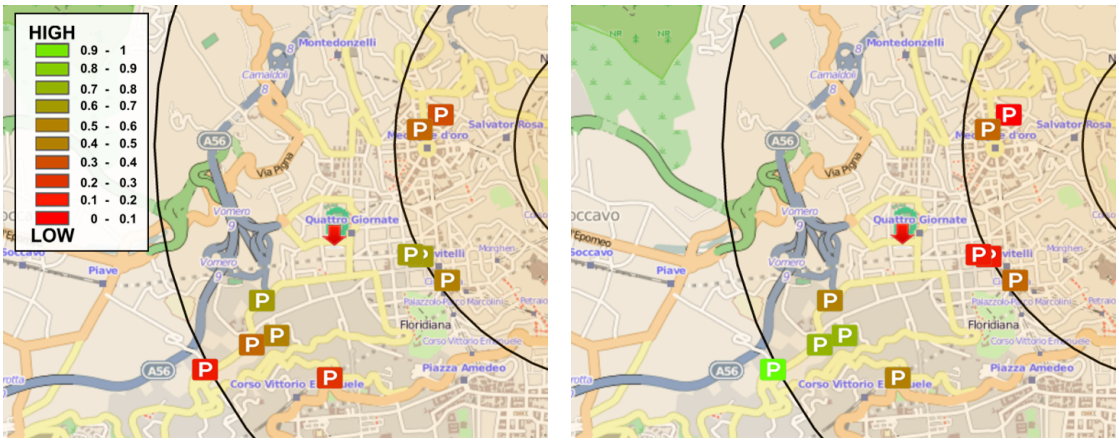

Figure 4.4: User Agent and Parking Manager Utilities.

# Rounds	ID	Availability	Distance	Price	Route	Time	PM Utility	UA Utility
1°	417856728	78	3530	7.78	1849	1676	0.77	0.19
2°	2204657189	27	4389	5.14	2151	1951	0.65	0.18
3°	1495201878	40	3719	7.30	1442	1110	0.59	0.45
4°	2245281153	87	2357	7.59	1030	720	0.58	0.62
# Rounds	ID	Availability	Distance	Price	Route	Time	PM Utility	UA Utility
1°	2204658556	171	3712	7.46	1126	848	0.72	0.53
2°	2239471042	237	2273	7.99	1263	1013	0.56	0.43
3°	2204657189	2	4389	5.82	2151	1951	0.50	0.11
4°	2204657190	7	3946	7.86	1525	1790	0.41	0.19
5°	1495201878	18	3719	7.52	1442	1110	0.40	0.39
6°	417856728	36	3530	7.92	1849	1676	0.40	0.18
7°	2245281149	138	2434	7.17	883	725	0.39	0.63

Table 4.2: Negotiation on a single query.

cess. The first experiments carried out shows that negotiation is a viable and promising approach since a solution is found before all the selected car parks are proposed to users. The second experimental result shows that car parks occupancy have an impact on the length of negotiation and further experiments will be carried out to find the relation between the occupancy percentage and the length of negotiation.

4.4.2 The Football Match

Another set of experiments was carried out to evaluate how software agent negotiation can be used in the selection of parking spaces in an urban area to push motorists to consider parking solutions that are not only biased on their preferences.

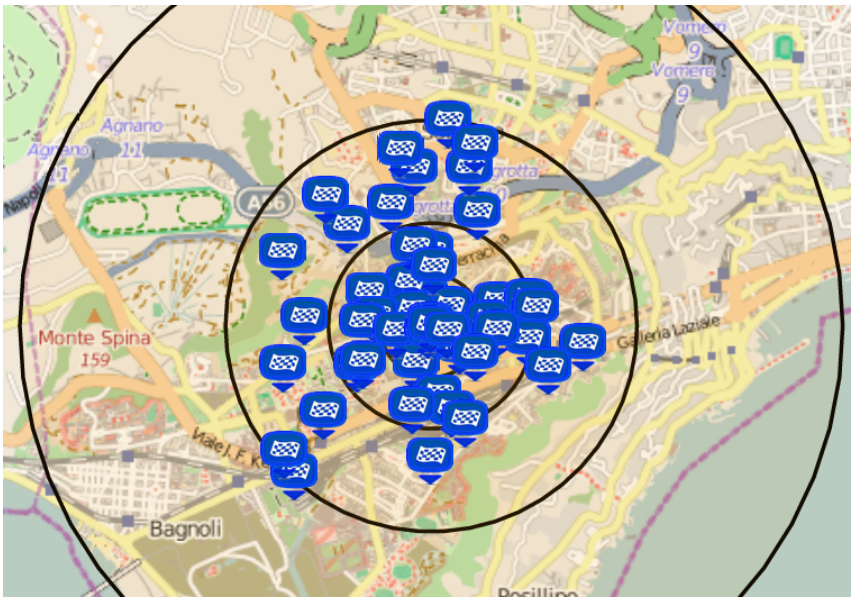

Figure 4.5: Queries distribution in sectors 0, 1 and 2.

The reference scenario consists of a set of users that make requests to park in different zones of Naples on the day a football match will take place. For this reason, it is considered beneficial for the city to make motorists to avoid the area around the football stadium for parking, in order to limit traffic congestion. So, the red zone is represented by the city sector (sector 0) centered in the location of the football stadium, with a radius set to $500m$ The rest of the city is split in sectors as well, starting from sector 0, with an exponential increased radius. The experiments simulate 60 queries (q_i) with destination locations distributed in sectors 0, 1 and 2 (20 queries in each sector), as shown in Figure 4.5. The sectors are determined with respect to the red zone (target t), and the destination locations are randomly generated. The threshold value for all users is set to 0.7 in all the experiments.

For each generated query the negotiation process between the UA and the PM takes place. In Table 4.3 we report, for each set of queries (respectively for sector 0, sector 1 and sector 2), the mean value together with the standard deviation of the following attributes of the parking space (s_i) selected after the negotiation: the UA and PM utilities (U_{UA} and U_{PM}), its distance from the red zone ($Dist(t)$), its walking distance ($Dist(q_i)$) and travel time distance with public means of transportation ($Time(q_i)$) from the query destination location, its offered price ($Price$), its position in the PM ranking ($Rank_{PM}$), and the social welfare value (SW), obtained as the sum of UA and PM utilities. The ranking

s_i	U_{UA}	U_{PM}	$Dist(t)$ m	$Dist(q_i)$ m	$Price$ $euro$	$Time(q_i)$ s	$Rank_{PM}$	SW
sector 0	0.75 ± 0.04	0.83 ± 0.22	1089 ± 255	866 ± 228	2.8 ± 0.4	269 ± 109	11 ± 10	1.59 ± 0.23
sector 1	0.75 ± 0.05	0.91 ± 0.09	1650 ± 304	1053 ± 167	2.2 ± 0.7	212 ± 77	3.8 ± 3.3	1.66 ± 0.11
sector 2	0.83 ± 0.04	0.93 ± 0.11	2399 ± 377	1145 ± 219	0.9 ± 0.8	213 ± 59	2.2 ± 2.1	1.76 ± 0.13
total	0.78 ± 0.07	0.89 ± 0.15	1713 ± 624	1021 ± 234	1.9 ± 1.0	232 ± 87	5.5 ± 7.1	1.67 ± 0.18

Table 4.3: Experimental data of park selection (s_i) w.r.t the queries q_i after the negotiation.

position of s_i corresponds to the number of the negotiation round at which the offer was sent by the PM, so representing the length of the negotiation (i.e., the number of rounds necessary to reach an agreement between the UA and the PM).

The obtained results show that the PM and UA utility values for the selected parking space increase when users' destination locations are far from the red zone. Furthermore, the negotiation length increases when users want to park in the red zone since it is more difficult to find a compromise. In fact, when users require destination locations far from the red zone the social welfare (last column of Table 4.3) increases since the needs of both the PM and the UA are easily satisfied. The distribution of the selected parking spaces for the considered queries is reported in Figure 4.6, showing that the parking spaces are selected in accordance with the objective to prevent motorists from parking in the red zone.

UA_{best}	U_{UA}	U_{PM}	$Dist(t)$ m	$Dist(q_i)$ m	$Price$ $euro$	$Time(q_i)$ s	$Rank_{PM}$	SW
sector 0	0.91 ± 0.06	0.21 ± 0.20	390 ± 163	339 ± 232	4.5 ± 1.0	101 ± 86	34 ± 5	1.12 ± 0.18
sector 1	0.98 ± 0.06	0.66 ± 0.17	1114 ± 217	473 ± 106	2.8 ± 0.4	115 ± 62	17 ± 6	1.64 ± 0.20
sector 2	0.99 ± 0.05	0.75 ± 0.14	1830 ± 463	616 ± 282	1.5 ± 1.0	114 ± 63	9 ± 6	1.74 ± 0.16
total	0.96 ± 0.06	0.54 ± 0.29	1112 ± 666	476 ± 244	2.9 ± 1.5	110 ± 70	20 ± 12	1.50 ± 0.33

Table 4.4: Data of UA best choices (UA_{best}) w.r.t. the queries q_i without negotiation.

PM_{best}	U_{UA}	U_{PM}	$Dist(t)$ m	$Dist(q_i)$ m	$Price$ $euro$	$Time(q_i)$ s	$Rank_{PM}$	SW
sector 0	0.44 ± 0.18	1 ± 0	1332 ± 153	1496 ± 362	2.5 ± 0.0	385 ± 109	1 ± 0	1.44 ± 0.18
sector 1	0.56 ± 0.21	1 ± 0	1875 ± 241	1656 ± 920	1.9 ± 0.9	240 ± 85	1 ± 0	1.56 ± 0.21
sector 2	0.57 ± 0.26	1 ± 0	2580 ± 350	2006 ± 1057	1.5 ± 1.0	114 ± 63	1 ± 0	1.57 ± 0.26
total	0.52 ± 0.22	1 ± 0	1929 ± 576	1720 ± 849	1.6 ± 1.0	294 ± 123	1 ± 0	1.52 ± 0.22

Table 4.5: Data for the PM best choices (PM_{best}) w.r.t. the queries q_i without negotiation.

In order to assess the benefit in using negotiation to find a compromise between the PM and the UA, we also evaluated the attribute values of parking spaces chosen respectively by the PM and the UA without negotiation. In this case, the PM and the UA select

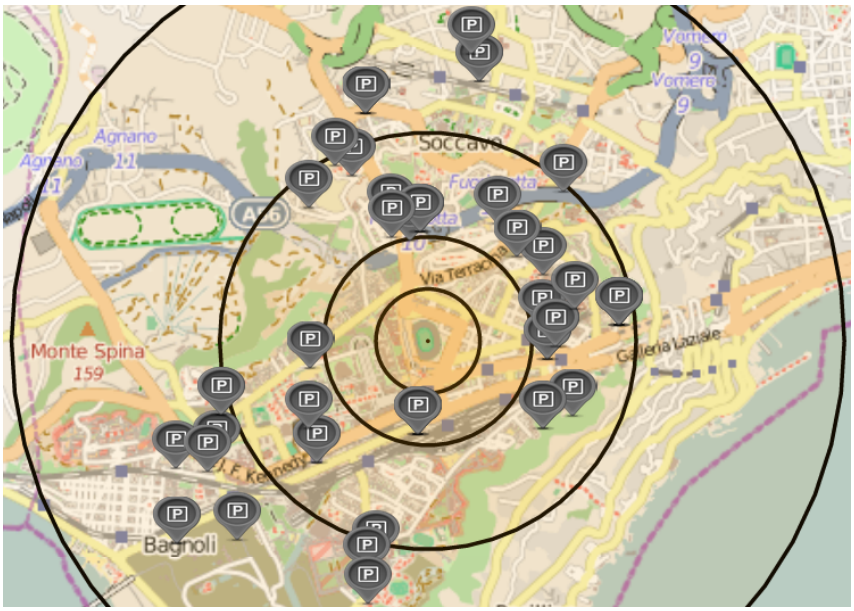

Figure 4.6: Selected parking spaces for the UA queries with negotiation.

respectively the best parking space that maximizes their own utility functions. Of course, in order for the PM and the UA to choose the best parking space, it is assumed that they both share the same information concerning the available parking spaces.

In Table 4.4 the same attributes described in Table 4.3 are reported for the best parking space for the UA (UA_{best}). As expected, in this case, the UA preferences are privileged while the PM utility value increases only for locations far from the red zone. Note that the ranking position of the selected parking space for the PM is in average 20, meaning that in case of negotiation such parking space would be offered to the UA only after 20 rounds, so requiring a longer (and hence more costly) negotiation with respect to the case reported in Table 4.3, where the average number of rounds is 5.5. Finally, the price of the UA_{best} is in the average higher than the price of the s_i because it corresponds to parking spaces nearer to the query locations and, hence, nearer, in average, to the red zone.

In Table 4.5 the same information as Table 4.4 is reported, but considering the best parking space for the PM (PM_{best}). In this case the PM preferences are privileged, while the UA utility value increases only for locations far from the red zone. Of course, the ranking value of the best parking space for the PM is 1, because it is normalized with respect to the max values of the parking attributes available for each query.

The average values of social welfare, reported in the last rows of Tables 4.4 and 4.5,

are lower than the one obtained with negotiation, since in these cases only the needs of one agent (respectively the UA and the PM) are taken into consideration. By the way, the average values of SW in the three tables are very close, but the values relative to each sector differ, so showing that negotiation is useful to improve social welfare when users want to park close to a red zone (i.e., for sector 0), while for sector 1 and 2 the social welfare is comparable.

The distribution of the best parking spaces for the UA, reported in Figure 4.7, is similar to the distribution of query locations, meaning that without negotiation users are not prevented from parking in the red zone. While the distribution of the best parking spaces for the PM, reported in Figure 4.8, is similar to the distribution of the parking spaces selected with negotiation since in this case the PM needs are considered.

4.4.3 Computing the Social Benefit of a Parking Allocation Process

In the set of experiments reported, we have already shown that negotiation is a viable approach to push drivers to select parking spaces that are also beneficial from a city point of view, at the level of single parking requests. In this subsection, we consider a global parking allocation problem for a set of requests, to be fulfilled with fixed number of available parking spaces When considering a whole set of parking space requests, each one

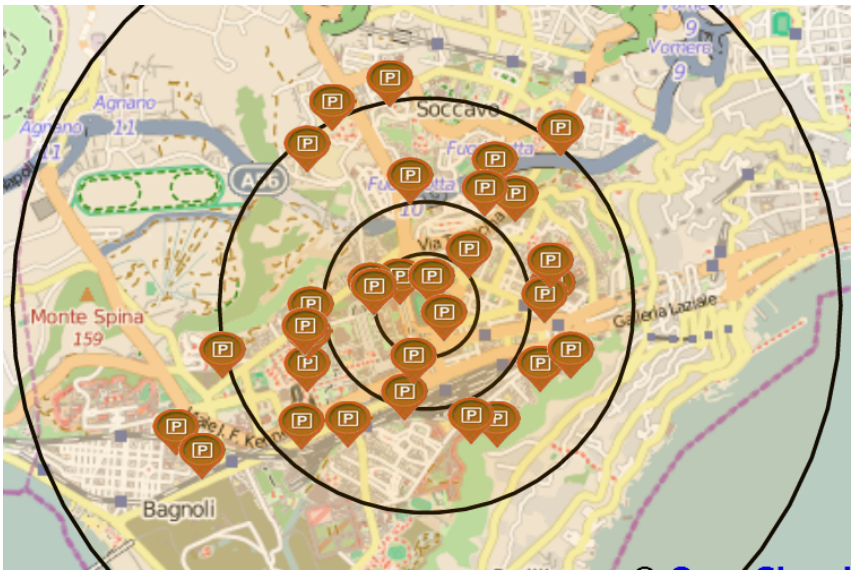

Figure 4.7: Distribution of UA_{best} parking space without negotiation.

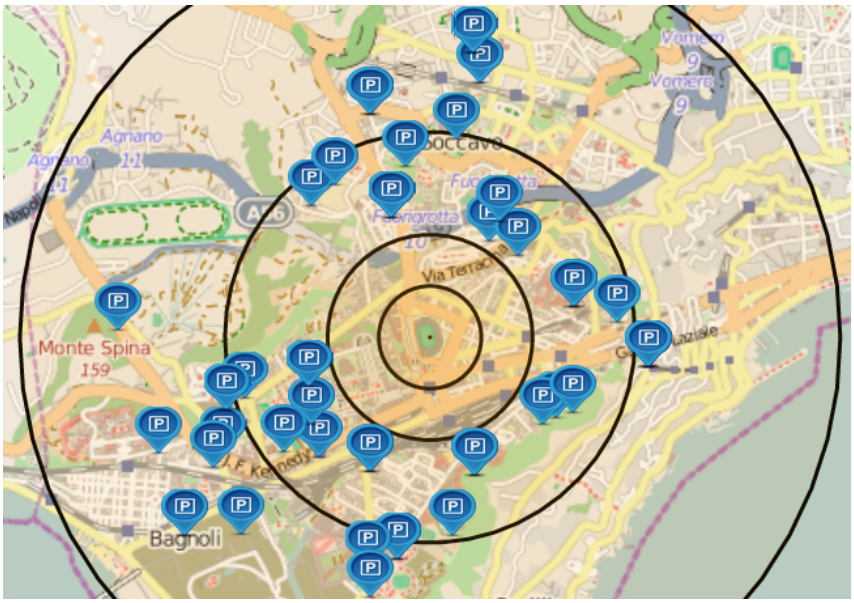

Figure 4.8: Distribution of PM_{best} parking space without negotiation.

processed through a negotiation process, the problem can be assimilated to a distributed indivisible resource allocation problem, where the selection of resources to be allocated for a specific request is carried out through a bilateral negotiation without considering the other requests. In our case, given a set of available resources $\mathcal{R}$ (i.e., parking spaces), and a set of driver agents UA, the overall process is to assign a single resource to each request (if available), in order to best match the UA request and, at the same time, to fulfill as many requests as possible. In resource allocation problems the *social welfare* is used as a metric to evaluate the efficient allocation of resources [19]. Hence, social welfare, computed for all requests, fulfilled or not, can be used also as a metric to evaluate an efficient allocation of parking spaces as follows.

In order to provide a measure of the social benefit of an allocation that takes into account different needs, the negotiation was evaluated in terms of the obtained social welfare of the global outcome of all negotiations occurring for the received parking requests. Different types of social welfare were evaluated by taking into that account:

- the distribution of parking spaces with respect to only drivers needs,
- the same distribution with respect to both drivers and city manager needs,
- the same distribution with respect to how the drivers and city needs are balanced.

Both the UA and the PM face a maximization problem when negotiating with each other, i.e. both agents try to maximize their individual utility.

Given a set of User Agents requesting a parking space, an optimal allocation of resources is the one that maximizes the social welfare of the driver agents, i.e., $SW_{UA} = \sum_{i \in \mathcal{UA}} U_i(x_{agr})$. Note that U_i depends only on the agent i and on the selected parking space (agr). Hence, the overall utility of a set of UAs corresponds to the sum of the individual utilities. In order to get a global utility value that does not depend on the cardinality of the set, a normalized version of the social welfare is used:

$$SW_{UA} = \frac{\sum_{i \in \mathcal{UA}} U_i(x_{agr})}{|\mathcal{UA}|} \tag{4.3}$$

Equation 4.3 accounts for the social welfare of driver agents and for the allocation problem in the sense that a high number of fulfilled requests with a high average utility will result in a high SW_{UA} value. However, in order to evaluate the social benefit of such an allocation of parking spaces, the social welfare should include also the utility of the PM. In fact, there could be two parking spaces that have the same utility for the UA, but one is more beneficial for the city welfare, i.e., it has a greater utility for the PM, so being a Pareto optimal solution with respect to the other one. For this reason, a social welfare evaluation should include the PM utility, so in our case a fair outcome of the negotiation is an agreement that maximizes a global social welfare (SW_+) obtained, for each negotiation, as the mean value of UA and PM utilities.

$$SW_+ = \frac{\sum_{i \in \mathcal{UA}} (U_i(x_{agr}) + U_{PM}(x_{agr}))/2}{|\mathcal{UA}|} \tag{4.4}$$

The adopted definition of social welfare does not account for situations with an inequality distribution of utilities among agents. In order to detect these situations the Nash Social Welfare definition [58] can be used defined as follows:

$$SW_* = \frac{\sum_{i \in \mathcal{UA}} (U_i(x_{agr}) \cdot U_{PM}(x_{agr}))}{|\mathcal{UA}|} \tag{4.5}$$

In the following subsections, we will use Equations 4.3,4.4, and 4.5 to evaluate the outcome of negotiation for the parking spaces allocation problem.

4.4.3.1 Negotiation Simulation

In order to assess if the proposed negotiation mechanism is able to push drivers to select a parking space beneficial for the different involved entities, an experimentation was carried out simulating a set of drivers' requests sequentially processed in a time window. The experiments are aimed to evaluate the percentage of the allocated parking spaces with respect to the number of processed requests, the available car parks, and the corresponding PM and UAs utilities. Currently, two UA profiles are considered:

- **strict**, i.e. drivers who are quite strict on their preferences, i.e. they are characterized by a high threshold value,
- **flexible**, i.e. drivers who are more flexible on their preferences, i.e. they are characterized by a low threshold value.

The evaluation is carried out against two baseline cases. In the first case, the availability and locations of all parking spaces is known to the UA, i.e., there is a complete knowledge setting. The UA selects the parking space (x_i) with the highest utility ($x_i = argmax(U_{UA}(x_j)), \forall j$), and it reserves it if this utility is above its threshold ($U_{UA}(x_i) > UA_{att}$). In the second case, the PM selects the parking space with the highest utility ($x_i = argmax(U_{PM}(x_j)), \forall j$) to offer, and the UA accepts it if its own utility for that offer is above the threshold ($U_{UA}(x_i) > UA_{att}$).

All requests specify a random destination that is located in the the first sector of the Historic Centre of Naples (Figure 4.9) with a radius of $500m$. The considered car parks are located in city sectors ranging within a radius of $5km$ from the city center and none located in the first sector that is assumed to be a pedestrian area.

The requests are issued by four different types of users as follows:

- **Flexible business**: $UA_{att} = 0.5$, $\beta_1 = 0.3$, and $\beta_2 = 0.7$;
- **Strict business**: $UA_{att} = 0.7$, $\beta_1 = 0.3$, and $\beta_2 = 0.7$;
- **Flexible tourist**: $UA_{att} = 0.5$, $\beta_1 = 0.7$, and $\beta_2 = 0.3$;
- **Strict tourist**: $UA_{att} = 0.7$, $\beta_1 = 0.7$, and $\beta_2 = 0.3$.

The PM instead has the same preferences on the attributes included in its utility function, i.e. $\alpha_1 = \alpha_2 = 0.5$ The number of processed requests is 50 or 100, and the number of

Figure 4.9: A representation of Historic Centre of Naples split in sectors.

total available parking spaces is 100 equally distributed over 20 car parks. The requests are processed one by one, and if a request is satisfied the corresponding assigned parking space is reserved, and it is not available for the other requests. If a request is not satisfied it is discarded and not processed anymore. We recall that the deadline of a negotiation may vary for each request according to the number of car parks with available places for that request.

4.4.3.2 Experimental Results

In Table 4.6, the overall UAs and PM utility values (U_{UA} and U_{PM}) and the percentage of successful allocations ($\%all.$) are reported, normalized w.r.t. the number of requests, in the case of 50 and 100 requests. Such utilities are evaluated for the negotiation case (Negotiation), and for the two baseline cases, i.e., the selection on the best parking space respectively for the UA (UA–best), and the PM (PM–best).

The results show that with negotiation a better parking space allocation is obtained (94% and 91%), with an increased overall utility for the UAs (0.68 and 0.67). Furthermore, the results confirm that with negotiation also the PM utility increases, so potentially finding an allocation that is more beneficial for the city as well (0.64 and 0.55). As

	50 req./100 spaces			100 req./100 spaces		
	U_{UA}	U_{PM}	$\%all.$	U_{UA}	U_{PM}	$\%all.$
Negotiation	0.68	0.64	94%	0.67	0.55	91%
UA–best	0.66	0.38	86%	0.60	0.38	79%
PM–best	0.31	0.37	46%	0.32	0.39	48%

Table 4.6: UAs and PM utilities in different settings.

expected, when privileging only the PM needs (PM–best) the PM utility does not increase, compared to the negotiation case because of the high number of failures in the allocation process for both 50 and 100 requests (respectively 46% and 48%).

50 Req./100 spaces						
	SW_{UA}	$max(SW_{UA})$	SW_+	$max(SW_+)$	SW_*	$max(SW_*)$
Negotiation	0.68	0.76	0.67	0.73	0.44	0.53
UA–best	0.66	0.66	0.52	0.71	0.29	0.57
PM–best	0.31	0.36	0.34	0.35	0.25	0.26

100 Req./100 spaces						
	SW_{UA}	$max(SW_{UA})$	SW_+	$max(SW_+)$	SW_*	$max(SW_*)$
Negotiation	0.67	0.72	0.64	0.69	0.40	0.47
UA best	0.60	0.60	0.49	0.65	0.29	0.51
PM best	0.32	0.38	0.36	0.37	0.27	0.28

Table 4.7: SW_{UA}, SW_+, and SW_* values for 50 and 100 requests.

In Table 4.7 the social welfare values (SW_{UA}, SW_+, and SW_*), evaluated respectively with Equation 4.3,4.4, and 4.5, are reported along with their corresponding optimal values ($max(SW_{UA})$, $max(SW_+)$, and $max(SW_*)$) for the cases of 50 and 100 requests. It should be noted that the definition of Equation 4.3 is exactly the overall UAs utility ($SW_{UA} = U_{UA}$).

As already highlighted in Table 4.6, a better overall utility for the UAs is obtained with negotiation, also compared with the UA–best baseline case. This unexpected result is due to the fact that negotiation leads to an increased percentage of parking spaces allocation, and hence, while the average value of the utilities is sub–optimal (i.e., it is less than $max(SW_{UA})$, $0.68 < 0.76$ and $0.67 < 0.72$), it is greater than the optimal value achieved in the case UA–best ($0.68 > 0.66$ and $0.67 > 0.60$). So even though this negotiation simulation does not lead to an optimal social welfare, it still improves the

social welfare with respect to the case of shared information.

When including the PM utility in the social welfare (SW_+), the values obtained with the negotiation are greater than both baseline cases. In addition, these values are now closer to their respective optimal values ($max(SW_+)$), i.e., the negotiation results in near optimal global outcomes.

Finally, negotiation allows for a better balancing of utilities among the involved agents, as showed by the values reported for SW_* ($0.44 > 0.29$ and $0.40 > 0.29$). Nevertheless, the values of $max(SW_*)$ with and without negotiation represent an opposite behaviour, apparently showing that utilities could be more balanced without negotiation. But this is not the case since the values of $max(SW_*)$ are not comparable with each other. In fact, the maximum values are considered only at local level for each selection step, but they do not represent the global maximum values for the overall selection process, that should instead be evaluated for all the possible permutations of allocations.

So, the results of the experiments carried out confirm that negotiation leads in average to better allocations and utilities for all the adopted measures when compared to experiments carried out without negotiation.

4.5 Related Works

Multi-agent negotiation has already been used in Intelligent Transportation System applications. In [1] cooperative agent negotiation is used to optimize traffic management relying on shared knowledge between drivers and network operators about routing preferences. In [41] a negotiation algorithm is designed for negotiating routes based on the calculation of routes utility, while in [13] agent negotiation is used for dynamic parking allocation, focusing on satisfying driver's preferences on prices and distances. Negotiation in smart parking application was used in [41] to determine the price of a car park. In our approach, the price of a car park is not negotiable, but it is dynamically set so to incentive users to select parking spaces located in specific urban areas. In our work a dynamic price mechanism is used as one of the factors influencing both the generation and the evaluation of parking offers.

Dynamic pricing mechanisms are being used in the context of parking applications. In [67] the authors presented, as in our case, a smart parking application that tries to find a trade–off between benefits of both drivers and parking providers. To balance the needs

of involved parties, they use a dynamic parking price mechanism as an incentive, as also used in [40], for the drivers to balance the convenience and cost in terms of parking price and the convenience in terms of parking distance from the user's destination. Differently from our approach, in [67] all the information is available and the parking selection is obtained as a maximization of drivers' utilities. Dynamic price mechanisms were also explored in [42], where the objective was to set up prices for available parking spaces in a such a way to propose the most efficient parking allocation, in terms of social welfare, intended as the total utility value of all agents for which a parking space is allocated. The social welfare in our approach is a result of a mediation of the conflicting needs of drivers and the city management.

We showed that negotiation improves the allocation of parking spaces when a complete set of requirement is processed. In our case the allocation problem is addressed but taking into account also other requirements (both of PM and UA) and not only the best exploitation of available parking spaces.

The optimal allocation of cars in car parks was also studied in [43], where the authors propose a semi–centralized approach for optimizing the parking space allocation, and improving the fairness among parking zones by balancing their occupancy–load. In this approach, parking coordinators are used to distribute the optimization allocation problem that is not manageable in a centralized way. In [26] the parking space allocation strategy, is also implemented as a global optimization problem, through the use of a Mixed Integer Linear Program. It is based on a user's objective function that combines proximity to destination and parking cost, while ensuring that the overall parking capacity is efficiently utilized. A set of requests are collected in a given time window, and they are processed by a software module producing an overall allocation that tries to optimize ad hoc function describing both driver–specific requirements, and system–wide objectives. In our case, the use of negotiation allows to model the parking space allocation problem not as a global optimization problem, but as the possibility to find a feasible compromise accommodating different needs. In our approach, we showed that a negotiation process is more effective, in terms of social welfare maximization, than a one–sided utility maximization.

Chapter 5

Conclusion

The research work of this thesis was carried out in the context of two PON funded by the MIUR on Smart City and Social Innovation, OR.C.HE.S.T.R.A. and S^2-Move, whose common goal is to promote the utilization of the modern Information and Communication Technologies in the cities, to develop smart platforms as decision support systems, that provide a set of technological solutions directed towards valorization and requalification of urban area. These platforms provide Service Based Applications (SBAs) aimed to satisfy users' requests by increasing the quality of life of city-dwellers.

It is well known Service Oriented Computing (SOC) is the enabling technology for development of SBAs, in fact promotes to assemble applications into loosely-coupled networks of services, where a service is any software entity developed on standard (Web Service), able to communicate through well-defined protocols that allow interoperability between different software systems. The SOC allows to integrate and develops different systems and functionality available through the network, in order to provide value-added functionality difficult to achieve with a single monolithic application.

In this context, we investigated the possibility of using advanced software mechanisms that enable the delivery of dynamic and adaptive Service-Based Applications (SBA), in highly dynamic environments and variable as those of the Smart City, also taking into account non-functional aspects of such applications, known as Quality of Services (QoS). Indeed, the evolution of the Internet from a vision document-centric to that of service-centric, led to a characterization of services that includes not only the functionality they offer, but also the ways in which the service is provided (e.g. cost, response time, reliability, reputation and so on), and their value may differ from one service to another.

The reference scenario in this work consists in a Smart City modeled as a dynamic market of services in which users and suppliers of services must interact. The user will issue a request for an SBA specifying the functionality of each service component, their functional dependence constraints, and the value(s) of the quality attribute(s) they want the application to provide. The suppliers provide in an independent and autonomous manner their services that, once aggregated, satisfy the requirements of users both from a functional point of view and from what concerns the overall QoS values of the application (end-to-end requirements). In this scenario, we assume that the QoS attribute values for the same service may change in time according to dynamic circumstances as: service provision, according to market trends, provision strategies, and so on.

Therefore, the research activities carried out are focused on the study of innovative methods for modelling and managing, the development of applications based on composition of services. Since software agents provide an appropriate paradigm to design scalable and open systems as the one based on SOC, we propose to use multi-agent paradigm to model services providers and users, and to adopt automated negotiation as a tool for modeling the interactions between providers and users of services. Automated negotiation was adopted both for the dynamic composition of services that meets specific requirements of QoS, and as a mechanism to develop a decision support tool for smart mobility enabling the achievement of an agreement between the users and providers of services in the presence of conflicting interests.The negotiation mechanism proposed is based on Iterated Contract-Net Interaction Protocol ICNIP, and two main roles are identified: initiator of the negotiation that cannot build a counter-proposal and may only accept or reject an offer; providers that proposes the most convenient offer for itself, willing to concede in order to be selected.

For what concerning the first aspect of this research work, it is assumed that service providers (Service Providers - SPs) are modeled as software agents and a Service Compositor agent (SC), is on agent acting on behalf of a user. We propose to use automated negotiation to select a set of SPs that provide services whose QoS values, once aggregated, fulfill the users' requirements expressed as global constraints on the Services Based Application's QoS. In this context, we proposed to model Service Providers strategies as Gaussian distribution to represent their stochastic behaviour in a zero-intelligent market, for both single-issue and multi-issue negotiation cases, i.e. to negotiate with one or more values of QoS respectively. In particular, Gaussian distribution is used to model mono-

tonic concession strategies for Service Providers, to map values of QoS in utility value, and to know the probability distribution of the offers. We showed that, the scaling properties of the Gaussian distributions allow to model single-issue and multi-issue negotiations in a uniform way. Hence, it was shown that the complexity of negotiation for SBAs depends on the number of issues and of services composing the SBA.

In the single-issue case, we analysed the negotiation trends varying the number of Service Providers involved, and negotiation rounds. The experiments carried out showed that it is worth to negotiate with all available SPs in order to increase the probability of successful negotiation, in fact in a market of services it is not possible to assume that a promising provider will keep on sending promising offers because a less promising provider may change its strategy in the meantime. In our approach, the increase in communication costs is partially compensated by the fact that, as shown in the experiments, by increasing the number of SPs the success rate of the negotiation increases. So, the overhead due to the communication cost is partially compensated by a decrease in the negotiation length, i.e. its overall computational cost. Furthermore, the analysis of the negotiation trends, in different configurations, allows to gather information that can be used to evaluate the possibility to stop negotiation if the outcome of the negotiation does not improve, by iterating the negotiation. This is a useful feature when adopting computationally expensive mechanisms like negotiation in service-based application settings since it is possible to limit the negotiation length according to its trends.

Finally, we extended the Orthogonal bidding strategy in order to be applied to our negotiation model for service composition. This extension is based on concept of *weighted reference point* and the use of an heuristic for the select the more promising set of offer at each round with all available providers, allows to achieve a near Pareto optimal agreement with a provider agent that would have been discarded according to the adopted heuristics since it was not promising at the beginning of the negotiation. Hence, a reference point computed considering a set of selected offers at a given round, allows to select a different set of offers at a successive round.

Moreover, we show that an agreement near Pareto optimal can be computed, if it exists. The reference point so defined allow to negotiate concurrently with all available service providers, this aspect is important when adopting negotiation for service composition since it avoids making the length of negotiation depending on the number of functionality requested by user. This is even more crucial when the considered reference

scenario for service composition is an open market of services since the time spent in negotiation may prevent its use in these settings.

As a second approach, the automated negotiation mechanism proposed, was also used to develop decision support tool for smart mobility, allowing to achieve an agreement among the users and service providers in the presence of conflicting interests. In this field of research, a Smart Parking application, including the decision tool (a prototype application for automatic selection of parking in urban car park) has been developed. The application allows to automatically provide a parking space the result of negotiation between agents software agents. So, we model the parking allocation as a multi–agent negotiation process to find an agreement between different and sometimes conflicting needs. Negotiation occurs among User Agents acting on behalf of drivers requesting to reserve a parking space that satisfies their own criteria, and a Parking Manager Agent acting on behalf of a city authority that tries to allocate parking spaces by accommodating city needs, respectively managing the allocation of parking available, and improving the circulation of cars in the city.

In the experiments carried out with Smart Parking application, we simulated the mechanisms of negotiation in case of a single request, and with multiple requests of parking spaces.

We showed that negotiation is a viable approach to push drivers to select parking spaces that are also beneficial from a city point of view, at the level of single parking requests. Moreover, we showed that also when considering the global parking allocation problem for a set of requests, negotiation leads to better utilities for both the UAs and the PM, together with an improved percentage of fulfilled parking requests.

In order to provide a measure of the social benefit of an allocation that takes into account different needs, the negotiation was evaluated in terms of the obtained social welfare of the global outcome of all negotiations occurring for the received parking requests. Different types of social welfare were evaluated by taking into account: the distribution of parking spaces with respect to only drivers needs, the same distribution with respect to both drivers and city manager needs, and finally the same distribution with respect to how the drivers and city needs are balanced. The results of the experiments carried out confirmed that negotiation leads in average to better allocations and utilities for all the adopted measures when compared to experiments carried out without negotiation.

In conclusion, the work carried out in this research shows that the use of the multi-

agent paradigm and of automated negotiation for Service Based Applications in the context of Smart Cities is a promising approach: allows to model a "smart" behaviour for the providers of SBAs; makes it possible the automate the composition of services based on specific user requirements of QoS, taking into account the variability of a market of services in terms of QoS; allows the management of conflicting interests among providers and users of services in applications of Smart Mobility.

Bibliography

[1] Jeffrey L Adler and Victor J Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10:433 – 454, 2002.

[2] Md. M. Akbar, M. S. Rahman, M. Kaykobad, E. G. Manning, and G. C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Journal of Compututers and Operations Research*, 33(5):1259–1273, 2006.

[3] Mohammad Alrifai and Thomas Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *Proceedings of the 18th Int. Conf. on World Wide Web*, WWW '09, pages 881–890, New York, NY, USA, 2009. ACM.

[4] B. An, V. Lesser, and K. Sim. Strategic agents for multi-resource negotiation. *Autonomous Agents and Multi-Agent Systems*, 23(1):114–153, 2011.

[5] Bo An, Kwang Mong Sim, Liang Gui Tang, Shuang Qing Li, and Dai Jie Cheng. Continuous-time negotiation mechanism for software agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1261–1272, 2006.

[6] Danilo Ardagna and Barbara Pernici. Adaptive service composition in flexible processes. *IEEE Trans. on Software Eng.*, 33(6):369–384, 2007.

[7] K.J. Arrow, A. Sen, and K. Suzumura. *Handbook of Social Choice & Welfare*. Handbooks in Economics. Elsevier Science, 2010.

[8] Tim Baarslag and Koen V. Hindriks. Accepting optimally in automated negotiation with incomplete information. In *Proceedings of AAMAS*, AAMAS '13, pages 715–722, 2013.

[9] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade: A software framework for developing multi-agent applications. lessons learned. *Inf. Softw. Technol.*, 50(1-2):10–21, January 2008.

[10] Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, and Ralf Steinmetz. Heuristics for qos-aware web service composition. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, pages 72–82. IEEE Computer Society, 2006.

[11] Rachel A Bourne, Karen Shoop, and Nicholas R Jennings. Dynamic evaluation of coordination mechanisms for autonomous agents. In *Progress in Artificial Intelligence*, pages 155–168. Springer, 2001.

[12] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1069–1075. ACM, 2005.

[13] Shuo-Yan Chou, Shih-Wei Lin, and Chien-Chang Li. Dynamic parking negotiation and guidance using an agent-based platform. *Expert Syst. Appl.*, 35(3):805–817, October 2008.

[14] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93, 2002.

[15] Claudia Di Napoli, Dario Di Nocera, and Silvia Rossi. Evaluating negotiation cost for qos-aware service composition. In *Proceedings of the 14th Workshop "From Objects to Agents" co-located with the 13th Conference of the Italian Association for Artificial Intelligence (AI*IA 2013)*, volume 1099 of *WOA '13*, pages 54–59. CEUR workshop proceedings, 2013.

[16] Claudia Di Napoli, Paolo Pisa, and Silvia Rossi. Towards a dynamic negotiation mechanism for qos-aware service markets. In *Trends in Practical Applications of Agents and Multiagent Systems*, volume 221 of *AISC*, pages 9–16. Springer, 2013.

[17] Claudio Di Napoli, Dario Di Nocera, and Silvia Rossi. Computing pareto optimal agreements in multi-issue negotiation for service composition (extended abstract). In *Proceedings of the 14th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS'15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[18] Harri Ehtamo, Raimo P. Hamalainen, Pirja Heiskanen, Jeffrey Teich, Markku Verkama, and Stanley Zionts. Generating pareto solutions in a two-party setting: Constraint proposal methods. *Management Science*, 45(12):1697–1709, 1999.

[19] Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources. *J. Artif. Intell. Res. (JAIR)*, 25:315–348, 2006.

[20] P. Faratin, C. Sierra, and N.R. Jennings. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205 – 237, 2002. International Conference on MultiAgent Systems 2000.

[21] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24:3–4, 1998.

[22] Peyman Faratin, Carlos Sierra, Nick R Jennings, and PHIL Buckle. Designing responsive and deliberative automated negotiators. In *Proc. AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, pages 12–18, 1999.

[23] S. Shaheen Fatima, Michael Wooldridge, and Nicholas R. Jennings. On optimal agendas for package deal negotiation. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1083–1084. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[24] FIPA. Fipa iterated contract net interaction protocol specification, 2001.

[25] Ian Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 8–15. IEEE Computer Society, 2004.

[26] Yanfeng Geng and C.G. Cassandras. New smart parking system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139, 2013.

[27] Henner Gimpel, Heiko Ludwig, Asit Dan, and Bob Kearney. Panda: Specifying policies for automated negotiations of service contracts. In *Service-Oriented Computing - ICSOC 2003*, volume 2910 of *Lecture Notes in Computer Science*, pages 287–302. Springer Berlin Heidelberg, 2003.

[28] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.

[29] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[30] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *The Journal of the Operational Research Society*, 55(12):1323–1332, 2004.

[31] Takayuki Ito, Hiromitsu Hattori, and Mark Klein. Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In Manuela M. Veloso, editor, *IJCAI*, pages 1347–1352, 2007.

[32] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *Int. Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[33] Nicholas R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001.

[34] Nicholas R Jennings, Simon Parsons, P Norriega, and Carles Sierra. On augumentation-based negotiation. *Int. Workshop on Multi-Agent Systems*, 1998.

[35] Mitri Kitti and Harri Ehtamo. Analysis of the constraint proposal method for two-party negotiations. *European journal of operational research*, 181(2):817–827, 2007.

[36] Guoming Lai and Katia Sycara. A generic framework for automated multi-attribute negotiation. *Group Decision and Negotiation*, 2009.

[37] KangChan Lee, JongHong Jeon, WonSeok Lee, Seong-Ho Jeong, and Sang-Won Park. Qos for web services: Requirements and possible approaches. *W3C working group note*, 25:1–9, 2003.

[38] Huan Liu, Farong Zhong, Bang Ouyang, and Jiajie Wu. An approach for qos-aware web service composition based on improved genetic algorithm. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, pages 123–128, 2010.

[39] Alessio R. Lomuscio, Michael Wooldridge, and Nicholas R. Jennings. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, 12(1):31–56, 2003.

[40] Wang Longfei and Chen Hong. Coorporative parking negotiation and guidance based on intelligent agents. In *International Conference on Computational Intelligence and Natural Computing*, volume 2, pages 76–79, 2009.

[41] Wang Longfei, Chen Hong, and Li Yang. Integrating mobile agent with multi-agent system for intelligent parking negotiation and guidance. In *4th IEEE Conference on Industrial Electronics and Applications*, pages 1704–1707, 2009.

[42] Reshef Meir, Yiling Chen, and Michal Feldman. Efficient parking allocation as online bipartite matching with posted prices. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 303–310, 2013.

[43] Naourez Mejri, Mouna Ayari, and Farouk Kamoun. An efficient cooperative parking slot assignment solution. In *UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 119–125. IARIA, 2013.

[44] Daniel Menasce. Qos issues in web services. *Internet Computing, IEEE*, 6(6):72–75, 2002.

[45] Daniel Menasce. Composing web services: A qos view. *Internet Computing, IEEE*, 8(6):88–90, 2004.

[46] M. Moghaddam and J. G. Davis. Service selection in web service composition: A comparative review of existing approaches. In Athman Bouguettaya, Quan Z. Sheng, and Florian Daniel, editors, *Web Services Foundations*, pages 321–346. Springer New York, 2014.

[47] Thuc Duong Nguyen and N. R. Jennings. Concurrent bi-lateral negotiation in agent systems. In Danielle C. Martin, editor, *4th DEXA Workshop on e-Negotiations*, pages 839–844. IEEE, 2003.

[48] Thuc Duong Nguyen and N. R. Jennings. A heuristic model of concurrent bi-lateral negotiations in incomplete information settings. In *International Joint Conferences on Artificial Intelligence*, pages 1467–1469, 2003.

[49] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1):192–224, 2006.

[50] OASIS. Web services business process execution language version 2.0, April 2007.

[51] Bing Pan, JohnC. Crotts, and Brian Muller. Developing web-based tourist information tools using google map. In Marianna Sigala, Luisa Mich, and Jamie Murphy, editors, *Information and Communication Technologies in Tourism 2007*, pages 503–512. Springer Vienna, 2007.

[52] Mike P Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 3–12. IEEE, 2003.

[53] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, 2007.

[54] Shamimabi Paurobally, Valentina Tamma, and Michael Wooldrdige. A framework for web service negotiation. *ACM Trans. Auton. Adapt. Syst.*, 2(4), November 2007.

[55] E. Polycarpou, L. Lambrinos, and E. Protopapadakis. Smart parking solutions for urban areas. In *2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, 2013.

[56] David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, pages 389–423, 2002.

[57] Iyad Rahwan, Ryszard Kowalczyk, and Ha Hai Pham. Intelligent agents for automated one-to-many e-commerce negotiation. volume 24, pages 197–204, Los Alamitos, CA, USA, January 2002. IEEE Computer Society Press.

[58] Sara Ramezani and Ulle Endriss. Nash social welfare in multiagent resource allocation. In Esther David, Enrico Gerding, David Sarne, and Onn Shehory, editors, *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, volume 59 of *Lecture Notes in Business Information Processing*, pages 117–131. Springer Berlin Heidelberg, 2010.

[59] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, MA, USA, 1994.

[60] Silvia Rossi, Dario Di Nocera, and Claudia Di Napoli. Normal distributions and multi-issue negotiation for service composition. In Javier Bajo Perez, Juan M. Corchado RodrÃguez, Philippe Mathieu, Andrew Campbell, Alfonso Ortega, Emmanuel Adam, Elena M. Navarro, Sebastian Ahrndt, MarÃa N. Moreno, and Vicente JuliÃ¡n, editors, *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, volume 293 of *Advances in Intelligent Systems and Computing*, pages 1–8. Springer International Publishing, 2014.

[61] Quan Z. Sheng, Xiaoqiang Qiao, Athanasios V. Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280(0):218 – 238, 2014.

[62] F. Siala and K. Ghedira. A multi-agent selection of web service providers driven by composite qos. In *Proc. of 2011 IEEE Symposium on Computers and Communications (ISCC)*, pages 55–60. IEEE, 2011.

[63] Reid. G. Smith. The contract net protocol: High–level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, 29(12):1104–1113, 1980.

[64] Dušan Teodorović and Panta Lučić. Intelligent parking systems. *European Journal of Operational Research*, 175(3):1666–1681, 2006.

[65] Quoc Bao Vo, Lin Padgham, and Lawrence Cavedon. Negotiating flexible agreements by combining distributive and integrative negotiation. *Intelligent Decision Technologies*, 1(1):33–47, 2007.

[66] W3C. Web services description language (wsdl) version 2.0 part 0: Primer, June 2007.

[67] Hongwei Wang and Wenbo He. A reservation-based smart parking system. In *IEEE Conference on Computer Communications (INFOCOM WKSHPS)*, pages 690–695, 2011.

[68] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F Ferguson. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005.

[69] AndrzejP. Wierzbicki. The use of reference objectives in multiobjective optimization. In GÃ$\frac{1}{4}$nter Fandel and Tomas Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 468–486. Springer Berlin Heidelberg, 1980.

[70] Colin R Williams, Valentin Robu, Enrico H Gerding, and Nicholas R Jennings. Iamhaggler2011: A gaussian process regression based negotiation agent. In *Complex Automated Negotiations: Theories, Models, and Software Competitions*, pages 209–212. Springer, 2013.

[71] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[72] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.

[73] M. Wu, M. de Weerdt, and H. La Poutré. Efficient methods for multi-agent multi-issue negotiation: Allocating resources. In Jung-Jin Yang, Makoto Yokoo, Takayuki Ito, Zhi Jin, and Paul Scerri, editors, *Principles of Practice in Multi-Agent Systems*,

volume 5925 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin Heidelberg, 2009.

[74] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The michigan internet auctionbot: a configurable auction server for human and software agents. In *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, pages 301–308, New York, NY, USA, 1998. ACM.

[75] Jun Yan, Ryszard Kowalczyk, Jian Lin, Mohan B. Chhetri, Suk Keong Goh, and Jianying Zhang. Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, 23(6):748 – 759, 2007.

[76] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, 2005.

[77] T. Yu and K. Lin. Service selection algorithms for web services with end-to-end qos constraints. In *Proceedings of the IEEE International Conference on E-Commerce Technology*, pages 129–136. IEEE Computer Society, 2004.

[78] T. Yu, Y. Zhang, and K. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1), 2007.

[79] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 411–421, New York, NY, USA, 2003. ACM.

[80] Liangzhao Zeng, Boualem. Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, may 2004.

[81] Ronghuo Zheng, N. Chakraborty, Tinglong Dai, K. Sycara, and M. Lewis. Automated bilateral multiple-issue negotiation with no information about opponent. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 520–527, 2013.

[82] Ronghuo Zheng, Nilanjan Chakraborty, Tinglong Dai, and Katia Sycara. Multiagent negotiation on multiple issues with incomplete information: Extended abstract. In

Proceedings of the 2013 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '13, pages 1279–1280, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.