UNIVERSITY OF SOUTHAMPTON

Social techniques for effective interactions in open cooperative systems

by

Maíra Ribeiro Rodrigues

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

November 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Maíra Ribeiro Rodrigues

Distributed systems are becoming increasingly popular, both in academic and commercial communities, because of the functionality they offer for sharing resources among participants of these communities. As individual systems with different purposes and functionalities are developed, and as data of many different kinds are generated, the value to be gained from sharing services with others rather than just personal use, increases dramatically. This, however, is only achievable if participants of open systems cooperate with each other, to ensure the longevity of the system and the richness of available services, and to make decisions about the services they use to ensure that they are of sufficient levels of quality. Moreover, the properties of distributed systems such as openness, dynamism, heterogeneity and resource-bounded providers bring a number of challenges to designing computational entities that cooperate effectively and efficiently.

In particular, computational entities must deal with the diversity of available services, the possible resource limitations for service provision, and with finding providers willing to cooperate even in the absence of economic gains. This requires a means not only to provide *non-monetary incentives* for service providers, but also to account for the level of *quality of cooperations*, in terms of the quality of provided and received services. In support of this, entities must be capable of *selecting* among alternative interaction partners, since each will offer distinct properties, which may change due to the dynamism of the environment. With this in mind, our goal is to develop mechanisms to allow *effective cooperation* between agents operating in systems that are open, dynamic, heterogeneous, and cooperative. Such mechanisms are needed in the context of cooperative applications with services that are free of charge, such as those in bioinformatics. To achieve this, we propose a *framework for non-monetary cooperative interactions*, which provides non-monetary incentives for service provision and a means to analyse cooperations; an *evaluation method*, for evaluating dynamic services; a *provider selection mechanism*, for decision-making over service requests; and a *requester selection mechanism*, for decision-making over service provision.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

α,β,γ,θ	generic agents.
srv	a generic service.
prv	a generic provider.
req	a generic requester.
prt	an interaction partner.
c	the result measure variable.
b	the evaluation strictness.
r, r'	renouncement values.
s, s'	satisfaction values.
t, t'	debt values.
v, v'	credit values.
g	a value representing a gain.
l	a value representing a loss.
ι	a subjective influence.
tol	the tolerance threshold.
δ	the subjective influence intensity.
rc	a refusal condition.
it	a previous interaction.
stb	the stage balance of exchange values.
stb^n	the proportion of negative stage balances in previous interactions.
oab^n	the proportion of negative overall balances in previous interactions.
oab	the overall balance of exchange values.
v^{acc}	the credit accumulated over previous interactions.
t^{acc}	the debt accumulated over previous interactions.
w_{srv,a_i}	the weight of attribute a_i in the overall evaluation of srv.
avs	the average satisfaction in previous interactions.
$totalt$	the total debt of a provider with a requester.
$totalv$	the total credit of a requester with a provider.
ι_{total}	the total influence applied over an objective value.
$Aeval_{srv,a_i}(c_{a_i})$	the evaluation function for attribute a_i of srv with result measure c_{a_i}.
$Seval(srv)$	the overall evaluation for srv.
$Peval(\alpha, srv)$	the overall evaluation for a provider α of srv.

$S[prt]$	the evaluation of a provider prt.
$A_{srv}^{prv}, A_{srv}^{req}$	the set of attributes for srv when evaluated by prv or req.
$PSE_{srv,\alpha}$	the set of evaluations $Seval(srv)$ of a provider α.
EV	the set of exchange values in the provision stage.
EV'	the set of exchange values in the reciprocation stage.
I	the set of influences.
P	the set of candidate providers.
P_o	the sequence of candidate providers ordered by some criteria.
P_{cop}	the sequence of providers more likely to cooperate.
P_{suc}	the sequence of providers more likely to yield successful interactions.
Q	the set of requesters.
Q_o,Q_m	the sequence of requesters ordered by some criteria.
Q_a	the set of requests that the provider needs to perform.
D_{on}	the set of agents that the provider depends on for any needed service.
D_{bd}	the set of agents that depend on the requester for a needed service.
RC	the set of refusal conditions.
IT	the set of previous interactions.

Acknowledgements

I first would like to thank Prof. Michael Luck, my supervisor, for all his guidance, patience, and dedication throughout my PhD. His constant support and careful corrections were indispensable during the formative and final stages of this research. I am grateful to CAPES, an agency of the Brazilian Ministry of Education, for sponsoring my PhD studies, and to Dr. Rafael H. Bordini and Dr. Antonio Carlos da Rocha Costa for giving me the first incentives to pursue a PhD abroad.

Many researchers have assisted in this work, and my special acknowledgements go to Dr. Antonio Carlos da Rocha Costa (UCPel, Brazil) for his valuable comments regarding Piaget's theory of exchange values; and Prof. David O'Connor, from the Centre for Proteomic Research at the University of Southampton, and Dr. Judit Nagy, from Imperial College London, for their help in providing proteomics expert knowledge and data. I am also indebted to my research colleagues Gopal Ramchurn, Luke Teacy, Felipe Meneguzzi, and Raj Dash, for their help and discussions during the course of my PhD.

Completing a PhD (specially abroad) is a challenging and demanding life experience, which could not have been successful without the emotional support of family and friends. I gratefully acknowledge my friends Alinne Veiga, Denise Cysneiros, Elen Lima and Solange Correa, for all their support and encouragement, particularly in the last year of my PhD.

Especially, I am most grateful to my family, who have followed attentively my journey from the other side of the Atlantic. I thank my brother and sister, whom I am very proud of, for all their love and attention. Most importantly, I don't have words to thank my parents, for their immeasurable love and support, that made the ocean between us feel like a drop of water, and that kept my head up, and my heart fulfilled. Thank you for taking me by the hand, and dancing with me in the middle of the fog.

To my parents, the stars that light my way.

(Para meus pais, as estrelas que iluminam meu caminho.)

Chapter 1

Introduction

1.1 From Centralised to Distributed Systems

The shift from monolithic standalone systems to systems comprising multiple computers has brought about new possibilities for applications in which the key issues are distribution, interaction, and cooperation. In this view, the power of the machine is realised not through individual computational power, but through the combination of different capabilities, services and resources in a broader distributed system. Such distributed systems (Coulouris *et al*, 2001) in which resources and participants are located in geographically different locations but interconnect, so that they can interact with each other, underlie the power of next generation computing.

While initially distributed systems were used for particular kinds of specialist application, recent years have seen a huge growth in the different kinds of use to which they are now put. Indeed, because they are so pervasive, distributed systems are often no longer distinguished from centralised systems. However, because of the functionality they offer, they are becoming increasingly popular, both in commercial and academic communities, for sharing resources among members of these communities (Chin *et al*, 2002; Little, 2003; Wroe *et al*, 2004; De Roure and Hendler, 2004).

More specifically, while many individuals or organisations previously generated data, or developed tools, for their own use, it is now possible for such tools and data to be made accessible to others as *services* in a distributed system (Foster, 2005). As individual systems with different purposes and functionalities are developed, and as data of many different kinds (ranging from scientific publications, newspapers, experimental results, and so on) are generated, the value to be gained from sharing with others rather than just personal use, increases dramatically. In this context, individuals from different organisations, in different geographical locations, can use distributed systems to discover new computational resources that complement or extend their own, in an effort to achieve goals that they could not otherwise achieve.

Many efforts have been made to develop the necessary infrastructures to support such a vision of distributed systems as a cooperative interconnection of people, computational resources, and organisations. Such infrastructures are generally concerned with the interoperability and access of services provided by different components of the distributed system. In this sense, services are modular applications with a common interface, so that they can be used in geographically distributed locations without knowledge of their implementation (Foster *et al*, 2001; Curbera *et al*, 2003). This *service-oriented* approach is ideal for *open* environments, since it abstracts away specific details of individually created data and tools, and allows the dynamic formation of new services from the composition of existing services. Indeed, open systems are characterised by continual change in the number and nature of the participating entities, and thus are distinguished from the more traditional *closed* systems, which have a fixed set of participants that usually operate under the same authority (for example, participants in a distributed system of a private organisation must follow the set of rules and protocols of that organisation) (Coulouris *et al*, 2001).

As participants of open distributed systems can join and leave the system at any time, such systems gain a *dynamic* character. At the same time, the variety of (either software or hardware) components and participants that form such systems make them highly *heterogeneous*. This is because different hardware configurations and software functionalities cause available services to have distinct properties, and individual viewpoints of participants cause them to have distinct preferences, goals and motivations.

The complexity of the computational entities that operate in open distributed systems depends on their functionality and on the characteristics of the system. As distributed systems become more complex, with large numbers of individual connected entities, of different types, in different configurations and adhering to different standards, and with new entities joining and leaving, the entities involved need to be endowed with abilities that go beyond merely sending, receiving and performing service requests. In particular, it has been claimed that computational entities need to be flexible and autonomous (Foster *et al*, 2004; Huhns and Singh, 2005): flexibility is required to cope with changes in the environment; and autonomy is required in order to take decisions about which of many possible courses of action is best, and to consider individual aims and objectives.

Given these characteristics, the computational entities that make up such distributed systems can be considered to be *agents* interacting in a multi-agent system. Agents are independent entities, distributed across a system, and capable of communicating, making decisions and cooperating (Wooldridge, 2002; Luck *et al*, 2005). They are flexible and autonomous, and typically interact with others in an effort to solve problems that they would not be able to solve alone. Although in general agents can provide the required underpinning technology for the computational entities that operate in distributed systems, there are some characteristics of domain applications and of open systems that demand particular mechanisms not always considered in the generic agent application.

Perhaps one of the most notable application domains of open and dynamic distributed systems is that of *bioinformatics* (Campbell and Heyer, 2002), which is particularly interesting because of its largely cooperative nature (Stein, 2002). Bioinformatics applications are characterised by a vast, heterogeneous, and constantly changing amount of (mostly free of charge) interrelated biological data and services and, as we will explain through this chapter, these characteristics have a significant impact on the possibilities for interaction and cooperation. In light of this, we view bioinformatics as an ideal domain in which to explore cooperative distributed systems, and will use it as the key motivating example, and domain of application, in this thesis.

1.2 Distributed Systems in Bioinformatics

The domain of bioinformatics is characterised by the application of computer technology to the management and analysis of biological data, and includes tasks like gathering, storing, analysing and merging information related to genes and proteins of living organisms (Feitelson and Treinin, 2002; Kim, 2002; Cohen, 2004). Research projects in this domain are those responsible for the sequencing of the genomes[1] of many organisms, including the human genome (The Wellcome Trust, 2001).

Biological data resulting from genome sequencing projects are very interrelated. Even though each organism has its own set of genes and proteins, the evolutionary process of species has resulted in some organisms having similar genomes (such as humans and chimpanzees, for example, which have 98% identical DNA). In addition, a gene can *code for* different proteins with similar functions but existing in different species (The Wellcome Trust, 2001), so that genes found in one organism can help in the study of genes from different organisms.

The domain of bioinformatics can be characterised as follows.

- There is a *great variety of tools and data* that has been generated by both large and small scale organisations, but many of these have similar functionality. For example, the European Bioinformatics Institute (EBI)[2] hosts more than 70 bioinformatics tools (of which 17 are alternatives for analysing DNA sequences) and 60 databases, while the Brazilian LNCC laboratory[3] hosts 6 databases and 1 tool for analysing DNA sequences of bacteria.

- Bioinformatics tools and data developed or generated by different individuals or organisations are typically *heterogeneous*, mostly regarding the quality of the data or results, even though they may have the same functionality.

[1]The genome is the total DNA content of a cell.
[2]http://www.ebi.ac.uk
[3]http://www.labinfo.lncc.br/main.php

- Bioinformatics is a very *dynamic* domain. Although the genomes of many organisms have already been identified, most of their function remains unknown, and bioinformatics continues to see an increase in the data being generated and services being developed for more specific areas like proteomics[4] and drug discovery. Indeed, as new information is generated, and data that was previously unknown gains an identified biological function, existing databases and tools are updated.

- Most of the tasks in bioinformatics involve processing large amounts of data (for example, humans have more than 30,000 genes, and the entire human genome requires more than 3 gigabytes of computer storage (The Wellcome Trust, 2001)). These tasks thus typically require significant amounts of processing time and power.

- Research in bioinformatics, as in other disciplines, is in many cases a cooperative activity. There are many providers of bioinformatics services with altruistic goals of contributing to the dissemination of knowledge (such as public biological databases NCBI[5], EMBOSS[6], etc), and also some with economic goals of earning money by charging for services (such as commercial search engine tools ProteinLynx and Phenyx). However, current proposals for computer systems that support the creation of global bioinformatics communities generally adopt cooperative approaches (Stein *et al*, 2001; Stein, 2002; Overbeek *et al*, 2004; Ellisman *et al*, 2004; Gao *et al*, 2005). In particular, these global communities are composed not only of large research centres (like those providing the services mentioned above) but also of small research groups and individual researchers who are willing to exchange services on a cooperative basis with the aim of improving individual and global results and discoveries.

Some characteristics, such as dynamism and heterogeneity, are common to open distributed systems in general. However, the large variety of interrelated data and tools found in the bioinformatics domain and their high computational demand, together with the cooperative character of service request and provision, are particularly exemplified in this domain, due to the dramatic explosion of interest and research, and the critical significance of its results. This raises several interesting challenges. While the domain has evolved over several years, and generally operates on a cooperative basis with many individual scientists and organisations participating, it is still emerging. Indeed, as the technology matures, and as the rewards to be gained from the research increase, there is a recognised need to move away from an *ad hoc* approach to one in which cooperation is supported and encouraged, especially in light of the lack of formal payment systems for services provided and received. We consider several key issues below.

[4]Proteomics is the study of the collection of all proteins in a cell, tissue, or organisms in a particular time (Campbell and Heyer, 2002).

[5]http://www.ncbi.nlm.nih.gov/Entrez/index.html

[6]http://emboss.sourceforge.net

1. The need to *incentivise* service provision among participants is a more general challenge for distributed services (Foster, 2005). Altruism cannot be relied upon to guarantee service provision, since some service providers (or their organisations) may have conflicting individual goals and thus may not be willing to cooperate with others. Moreover, since services are not always paid for, as with the majority of services in bioinformatics, but since service providers are (or must be considered as) self-interested entities, some non-monetary incentive for cooperation is important. Since providing a service always incurs some cost and requesting a service has no (significant) cost, it is easier for participants joining an open system to request services than to provide services, and an incentive is necessary to maintain a rich source of services and resources.

2. Once cooperation is motivated and there are more participants in the open system providing services (as individuals or organisations), service users may find a variety of available services. This *service variety* is beneficial for participants in open systems, since it is more likely that they will find the required services for achieving their goals or to improve the results of their problem-solving tasks. However, service variety also gives rise to the problem that, when a participant is faced with several alternatives for a required service, it needs a means to select among them.

3. The existence of *heterogeneous* data and tools suggests a concern with finding those with particular properties or levels of quality when selecting among alternatives (for example, finding the service with highest accuracy, with greater availability, or with faster results).

4. When a system is dynamic, not only may the existing set of services for a particular task change, but the properties of those services may also change. In addition, in the former case, the entrance of new service users may cause more demands on a particular service, making the system more competitive, while new providers may cause the opposite effect. This suggests that, to cope with such changes, any decision that participants make over their interactions with other participants of the open system must be performed *dynamically.*

5. A common issue for distributed systems with many participants, as observed by Huhns and Singh (2005), is that it is likely that useful services will become overloaded, especially if computational service requesters can generate many more requests than a human participant. This is particularly relevant for distributed bioinformatics applications, since most tasks in bioinformatics involve the processing of large amounts of data, and thus require significant computational time and power. Consequently, service providers may need to *limit the number of services* they provide when resources are scarce, in order to avoid being overloaded by tasks to perform.

Developing computational entities that operate efficiently in open systems in which the above issues arise is challenging, since it involves taking into account many different aspects of interactions. For example, computational entities must be capable of differentiating possible interaction partners dynamically, and of choosing among possible interactions (also dynamically) by taking into account the *properties of others* and their own *limitations* and *preferences*. Finally, all decisions relating to interactions must be taken in a cooperative context, since they may influence existing and potential cooperations. As argued above, such characteristics suggest a need for autonomous, flexible and cooperative behaviour, and these entities can therefore be seen as agents.

1.3 Agent-based Systems

The agent-oriented paradigm has been advocated as a natural way to design and implement dynamic and heterogeneous distributed systems (Foster *et al*, 2004; Luck *et al*, 2005; Huhns *et al*, 2005). This is because agents can be viewed as independent modules that are distributed over a system, and can interact with each other in order to achieve individual or global goals. More importantly, agents are *autonomous* (Luck and d'Inverno, 2001). They are capable of making decisions and solving problems, both individually and in cooperation with others, in order to perform tasks, overcoming individual limitations and achieving more complex goals than might otherwise be achieved. Such autonomous and cooperative behaviour provides flexibility to cope with changes first in the system in which agents operate, and second in the properties of the agents with which they interact.

Moreover, agents are capable of coordinating their tasks, and negotiating with others to reach agreements over different goals. This is important in open systems in which participants may need to form partnerships or coalitions despite possibly conflicting interests and objectives. For example, agents may form a partnership in which each agent provides a service that the partner is not able to perform, so that each can achieve its individual goals. Agents can also form coalitions in which each agent provides a service which is required to achieve a common goal, so that the result is more efficient than if they were operating alone.

To design cooperative distributed systems, such as those needed in the bioinformatics domain, and not assuming that all participants have benevolent behaviour, it is necessary to incentivise participants towards cooperation. Such incentives can be of an economic nature, such as any monetary gain, or of a social nature, such as reciprocal and cooperative relationships. Since agents are seen as both rational and social entities, they are capable of economic and social behaviour, the former involving decision-making strategies to maximise the expected utility of the agent (Parsons and Wooldridge, 2002), and the latter involving the incorporation of social concepts, like norms and commitments (Castelfranchi, 1998), as part of their reasoning process, thus facilitating the identification and maintenance of cooperation and reciprocity.

Agent-based systems can also be used to simulate the behaviour of distributed systems when these are designed as comprising multiple interacting autonomous agents with their own aims and preferences (Dawid *et al*, 2001; Balmer *et al*, 2004; Gilbert and Troitzsch, 2005). This allows the study of possible emergent system-level behaviour and the identification of characteristics of individual behaviour that can lead to desirable emergent properties.

In summary, we argue that designing and implementing systems of the kind described above as collections of autonomous agents that have flexible behaviour is an appropriate way to cope with dynamism and heterogeneity, while cooperative behaviour is needed for maintaining the service variety that makes such open distributed systems so valuable for many domains.

1.4 Research Goals

As described above, agent-based systems are a natural way of designing and implementing distributed systems that are dynamic and heterogeneous, and in which the individual components are autonomous, cooperative and flexible. However, the specific issues related to the target domain of bioinformatics, described in Section 1.2, make it difficult to design agents that cooperate efficiently, since agents must deal with the diversity of available services, the possible resource limitations for service provision, and with finding providers willing to cooperate even in the absence of economic gains.

In particular, such agents must be self-interested, reflecting the real world, as opposed to benevolent, even though they might manifest benevolent behaviour. We assume that agents are motivated to cooperate with others such that the load of service requests is shared among different service providers with a balance between service request and provision in the system. This ensures the longevity of the system through fairness, where agents (or service providers) continue to participate because they gain and lose in broadly equal measure, and don't simply incur costs without benefit. In this context, incentives for service provision are also important because the more services and information are available, the greater potential benefit available to participants from accessing them.

Moreover, since such systems are open and dynamic, interactions must be flexible with participants stopping or starting different interactions at any time. This copes with participants joining and leaving the system and allows them to terminate interactions if they are not satisfactory, or not desired (for example, if the quality of a service provider decreases or if performing a service for another participant will exhaust local resources). In support of this, participants must therefore be capable of selecting among alternative interaction partners, since each will offer distinct properties, which may also change due to the dynamism of the environment.

The key aim of this thesis, therefore, is to develop mechanisms to allow *effective cooperation* between agents operating in systems that are open, dynamic, heterogeneous, and cooperative. Such mechanisms are needed in the context of cooperative applications with services that are free of charge, such as those in bioinformatics, which provides additional challenges in developing agents with self-interested cooperative behaviour. More specifically, we can break this broad aim down into the following particular research goals.

1. **Find agents who will cooperate**. Agents need to find, from among alternative partners, those that are more likely to provide services either now or in the near future. Given the restrictions on cooperative behaviour imposed by free services, self-interested behaviour, and resource limitations, finding a cooperation partner could take a long time. To avoid this requires particular actions from service providers and requesters, as follows.
   (a) **Incentivise self-interested providers when services are free of charge**. Requesters must make use of non-monetary incentives to motivate providers to cooperate, even in the absence of any economic gain. This is important to ensure the longevity of the system and the richness of services and resources that brings greater potential benefit for participants of the system.
   (b) **Choose requesters that bring future benefits in terms of potential cooperation**. Providers must engage in interactions that improve their chances of finding cooperation partners in the future but, at the same time, they must limit service provision when resources are scarce. Such a strategic limitation of service provision contributes to effective cooperation since denying a request in a cooperative environment may affect future interactions, and some instances of cooperation may bring more benefits than others.
2. **Select cooperation partners with the best properties based on individual agent perspectives**. It is likely that agents operating in open systems will find services being provided with different levels of quality, and that agents have different perspectives over the quality of services received.
3. **Instantiate proposed cooperation mechanisms for systems in the bioinformatics domain**. Due to the large variety and demand of interrelated services in bioinformatics, and its cooperative character, using services and resources, and maintaining their richness in open bioinformatics environments depends on effective cooperation among their participants. However, such domain-specific properties present a challenge to developing agents that operate efficiently. Therefore, tailoring our proposed mechanisms to bioinformatics provides a case study for agents in open cooperative systems more generally.

Although there are existing mechanisms in the literature to achieve parts of these goals individually, as will be discussed further in this thesis, there are limitations in dealing with the specific characteristics of open systems that we focus on here, and which can be found in systems in the bioinformatics domain. We claim that by addressing these goals, we contribute to the development of self-interested agents that combine flexible and autonomous behaviour with cooperative behaviour, in order to use and share free services efficiently in open cooperative applications.

1.5 Research Contributions

The research goals defined in the previous section point to two different types of solutions. First, we need to develop decision-making mechanisms that allow agents to decide whether to cooperate with others and with whom to cooperate in the constantly changing open environment. Second, we need to provide the elements on which this decision-making is based, including the incentives for cooperation, the properties of cooperations and of other agents in the open system, and the cooperative relationships among agents.

In this context, the key contribution of this thesis is a set of interrelated mechanisms necessary for supporting and performing decision-making over interactions between agents in open cooperative systems. Such mechanisms are dynamic, and thus cope with changes in the environment that might affect existing cooperations, such as changes in the properties of services, in the set of participants in the system and the services they provide and request. Moreover, all mechanisms support the view that agents are self-interested, as opposed to benevolent, preventing participants from having to rely on benevolence to receive services from others, and contributing to longevity through a fair system of incentives for service provision. In a general sense, we argue that the latter also contributes to the development of open systems that are rich in the number and variety of available services, and in which participants take efficient decisions over the interactions in which they engage.

In carrying out this work, several distinct and specific contributions have been made. We briefly outline these below, but provide a more substantial discussion of our contributions at the end of the thesis.

1. We develop a *framework for non-monetary interactions* among self-interested agents, based on Piaget's theory of exchange values (Piaget, 1973), in which the motivation to cooperate comes from acquiring, accumulating, and spending (non-monetary) credits and debts that result directly from interactions. Such a framework includes a *computational model* of Piaget's exchange values, which defines how exchange values are accumulated and spent by interacting agents, provides the

basic incentive for non-monetary cooperations, and provides a means for agents to compare the quality of the services provided and received in a cooperation. This work is presented in Chapter 6.

2. We describe an *evaluation method* to analyse the outcome of dynamic services, in order to provide a guide for agents in future decision-making over alternative interaction partners. We consider the application of the evaluation method to the bioinformatics domain, in particular to evaluate services used in protein identification experiments. Such a mechanism is valuable for agents operating in open systems in which services are provided with different levels of quality, so that accurate information about those services can contribute to more efficient decision-making over interactions. This work, which is presented in Chapter 5, has been published as follows:
   M. R. Rodrigues and M. Luck. Evaluating dynamic services in bioinformatics. In M. Klusch, M. Rovatsos, and T. Payne, editors, *Cooperative Information Agents X*, volume 4149 of *Lecture Notes in Artificial Intelligence*, pages 183–197. Springer-Verlag, 2006.

3. We describe a *partner selection mechanism* for dynamic cooperative applications with free services, using a social interaction model to allow agents to find available partners quicker despite the resource limitations of service providers, and changes in the services that participants provide and need. Such a partner selection mechanism is suitable for agents competing for available providers in a cooperative environment (such as those in which providers must limit service provision due to resource constraints). In particular, it copes with changes in the environment that affect existing cooperations by using a dynamic model of interactions and decision-making (as opposed to other mechanisms that use static models of interactions (Sichman *et al*, 1994; David *et al*, 2001)). This work, presented in Chapters 7 and 8, has been published as:
   M. R. Rodrigues and M. Luck. Analysing partner selection through exchange values. In J. Sichman and L. Antunes, editors, *Multi-Agent-Based Simulation VI*, volume 3891 of *Lecture Notes in Artificial Intelligence*, pages 24–40. Springer-Verlag, 2006.

4. We describe a *model for cooperative interactions* among self-interested agents and *partner selection strategies* that use this model to choose among possible cooperations. The model for cooperative interactions uses non-monetary incentives for service provision, and provides a means for agents to analyse the cooperations in which they engage in terms of the quality of the services provided and received in reciprocation. By using non-monetary incentives for service provision, the model supports cooperative behaviour among self-interested agents in the context of cooperative applications with free services. By providing dynamic information about cooperations, this model supports flexible decision-making in open cooperative en-

vironments. This work, presented in Chapters 6, 7 and 8, has been published as: M. R. Rodrigues and M. Luck. Cooperative interactions: An exchange values model. In *Proceedings of the Coordination, Organization, Institutions and Norms in Agent Systems Workshop at the Seventeenth European Conference on Artificial Intelligence*, pages 63–70, 2006.

1.6 Thesis Structure

We start this thesis by presenting an overview of agent technology and the mechanisms it uses to address the problems inherent to distributed cooperative systems. In particular, we describe existing approaches that deal with some of the research goals stated in the previous section and their limitations in Chapter 2. In Chapter 3, we introduce the application domain of bioinformatics and give examples of existing bioinformatics systems for managing computer-based experiments. Here, we also describe a specific bioinformatics application which we take as our problem scenario, and identify the target problems we aim to address and our proposed solutions in more detail. A method for assessing different properties of services through dynamic evaluation, and its application to bioinformatics services are shown in Chapter 5. A computational framework to incentivise non-monetary cooperative interactions based on a social theory is presented in Chapter 6. Next, a mechanism for selection of alternative service providers is presented in Chapter 7, together with strategies for selection according to different criteria. Similarly, a mechanism for an agent to selecting among incoming requests is introduced in Chapter 8, together with different selection strategies. Both selection mechanisms use the proposed evaluation method and computational framework for non-monetary cooperative interactions through strategies, since each provide a different criteria for selecting interaction partners which are captured by distinct strategies. Finally, the different strategies that implement each selection mechanism are compared through an experimental testbed, in Chapter 9, in order to identify specific advantages and disadvantages of each strategy to the specific purposes of each selection mechanism. Since those strategies use information on the evaluation method and on the computational framework for non-monetary cooperative interactions, their properties are also tested. Conclusions and the summary of research contributions are presented in Chapter 10.

Chapter 2

Background Review

2.1 Introduction

Computational entities operating in open distributed systems typically have bounded resources and capacities, and depend on others to solve problems in order to achieve their goals. Even when they are capable of achieving their goals alone (because they can perform all the tasks involved), there may be other participants that can perform the same tasks but with better quality or in less time. To overcome individual limitations or dependencies that prevent the achievement of their goals, or to optimise their behaviour, these computational entities must work together and *cooperate*.

The problem of cooperation among (autonomous) computational entities that operate in distributed systems has long been studied in multi-agent systems research, e.g., (Wooldridge and Jennings, 1999; Sen and Dutta, 2002; Tate, 2006). In a multi-agent system, agents are capable of communicating and interacting with each other in a flexible and autonomous manner, in order to meet their design objectives (Luck *et al*, 2005). These characteristics make the agent paradigm suitable for the design of cooperative computational entities in open distributed systems (Foster *et al*, 2004; Huhns *et al*, 2005). Moreover, agents can be viewed as independent modules that are distributed across a system, a well-known principle for avoiding the complexity of managing large systems.

However, since agents in open systems may have different, and sometimes conflicting, interests and goals, cooperation is difficult to achieve. In particular, due to individual interests and resource limitations, agents may not always be willing to cooperate with others. Indeed, agents with different skills and preferences suggests that some cooperations will be more desirable than others, in the sense that some agents may perform services of better quality and faster, or be more reliable than others. Perhaps more important is that open systems allow agents to join and leave at any time, requiring cooperation to be flexible in the sense that agents might need to terminate existing cooperations and start new ones.

Effective cooperation among autonomous agents in open systems requires cooperative behaviour to be *motivated*, and cooperations to be *formed* among groups of agents in order to take advantage of individual diversity. In this context, in this chapter we review research on cooperative behaviour and the formation of cooperations, and analyse existing approaches in the multi-agent systems literature.

We start by introducing the notion of agents in Section 2.2. Then, we discuss existing approaches to coordinating agent behaviour, in Section 2.3, and to motivating and maintaining cooperative behaviour for achieving local and global goals, in Section 2.4. We then focus on how agents can form efficient cooperations, both as a group and as a pair of agents. To achieve this, agents first need a means to find cooperation partners, and then to solve possible conflicts in forming a cooperation. Therefore, in Section 2.5 we review alternative approaches to finding cooperation partners so that agents take advantage of relevant skills, or overcome dependencies. Here we focus in more detail on existing approaches to forming efficient cooperations among a pair of agents. Then, given that agents in open systems may have different interests, preferences and goals, in Section 2.6 we focus on how agents can reach agreements through negotiation to form a cooperation. Finally, in Section 2.7 we discuss the limitations of current approaches to effective cooperative behaviour in open distributed systems. Note that this chapter does not aim to provide an *extensive* review of the topics mentioned above, but to indicate relevant approaches to our problem, so that there is a better understanding of the problem and a guide to the kinds of solutions that we should seek.

2.2 Agents

In computer science, and more specifically in artificial intelligence (AI), an agent is seen as a computer system, situated in some environment, and capable of flexible autonomous action in order to meet its design objectives (Russell and Norvig, 1995; Luck *et al*, 2005). This view emphasises four attributes as essential requirements for agenthood: *autonomy* (the ability to act without intervention and have control over behaviour towards goal achievement), *reactivity* (the ability to react in a timely fashion to perceived conditions of the environment), *social ability* (the ability to interact with other agents to facilitate problem-solving) and *pro-activeness* (the ability to take the initiative to act and make decisions).

In addition to the basic properties listed above, further complementary attributes identified for agents include *intelligence*, *believable personality*, *mobility*, *adaptability*, and *rationality* (Luck and d'Inverno, 2001). These attributes have more or less importance according to the applications for which agents are used. For example, an interface agent should manifest believable personality (Bates, 1994) to attract the user's interest and attention, and information agents might offer mobility (Glitho *et al*, 2002), to be able to search for information in remote sources, more than believable personality.

Although there is some divergence of opinion about the agent concept and its essential attributes, there seems to be a general consensus that *autonomy* is the central point (Wooldridge, 1999). An autonomous agent should be able to act without any intervention (of humans or other agents), and to have full control over its internal state and its behaviour in order to achieve its goals. A more elaborate concept of autonomy can be found in (Luck and d'Inverno, 2001), in which it is strongly related to goal generation, goal adoption and motivation. Specially, Luck and d'Inverno (2001) define autonomous agents to be those agents that generate their own goals from motivations (as opposed to goals being generated by the user or by other agents).

Agents are normally situated in environments with other agents with which they interact to solve problems and to achieve goals. In open and dynamic environments, however, when agents have limited abilities, resources, and knowledge about the global system state, they must *cooperate* with each other and *coordinate* their actions in order to meet their individual goals and to form a coherent whole (in which the parts work in harmony) (Wooldridge, 2002). These are called *multi-agent systems.* When agents have individual interests or do not share common goals, they have to *negotiate* with each other to reach mutually beneficial agreements so that cooperation and coordination are possible. One can say that all of this is possible by means of *interaction*, through which agents can exchange tasks, information about the environment, and other agents' abilities and goals.

Interactions between agents in a multi-agent system are not pre-defined, arising as a result of the agents' autonomous decisions, and they allow simpler individual behaviours to be combined, through coordination, cooperation and negotiation, into more complex global system behaviour. Achieving coordination, cooperation and reaching agreements through negotiation are central problems for multi-agent systems research.

Within this space, a more specific problem is to find the most appropriate agents to interact with in order to solve some problem. These interaction partners can be agents with complementary abilities or information necessary to perform a particular task. *Partner selection* is the research area within multi-agent systems concerned with mechanisms and models for choosing which agent or agents to engage in interaction with (Sichman *et al*, 1994; Munroe *et al*, 2004).

The abilities to interact and to work as a group to achieve goals are characteristics found in both animal and human societies, and because multi-agent systems can also have these properties, they can be viewed as *artificial societies.* This makes possible not only the application of multi-agent systems in social simulation (Conte *et al*, 1997), but also the use of social concepts to improve multi-agent systems coordination and cooperation. Examples here include the use of social norms for regulating agents' interactions

(Wooldridge, 2002), and of social dependence (David *et al*, 2001) for selecting partners, for example.

The next sections consider coordination and cooperation in more detail, and describe existing approaches for these problems in the multi-agent systems literature.

2.3 Coordination

Agents operating in a society of agents need to *coordinate* their actions in order to solve their problems and to promote harmonic behaviour. Jennings (1996) defines coordination as the process by which an agent reasons about its local actions and the (anticipated) actions of others so that the community can act in a coherent manner, and argues that without coordination, the benefits of decentralised problem solving disappear.

Key approaches to coordinating agents' activities include: multi-agent planning (Durfee, 1999), commitments (Jennings, 1996), norms (Lopez y Lopez *et al*, 2005), and organisational structures (Dignum *et al*, 2002). These all share the idea of *constraining* agent behaviour in some way so that coordination is possible. For example, an agent's actions might be constrained by taking into account other agents' plans, commitments with other agents, and the rules established by an organisation, or by constraining the possible interactions in which an agent can participate. These approaches are described in more detail below.

2.3.1 Multi-agent Planning

With multi-agent planning, agents coordinate their activities by analysing other agents' plans. Planning is one of the most complex issues in multi-agent systems, since agents are distributed in the system, and an agent's plan must consider several constraints, including those over goals (such as missing resources, blocked sub-goals, etc), capabilities (such as limited skills to perform tasks), the environment (such as limited access to global information), as well as those constraints that others place on an agent's choices (since one agent can choose to use a resource that is also needed by another). Multi-agent planning can be achieved by having a central control agent to address plan interdependencies before action, as in (Georgeff, 1983). Although such an approach offers a fairly simple solution for distributed planning, it is susceptible to the failure of the central control agent, which is also a bottleneck for system performance. As an alternative, and based on the idea that agents do not need to have local access to all necessary information for constructing their plans, Durfee (1999) developed the *partial global planning* approach, in which agents interact with each other to communicate plans and goals in order to form expectations about the others' future behaviour and to adjust their own local planning.

2.3.2 Joint Commitments

As argued by Jennings (1996), commitments and conventions are the fundamental mechanisms for coordinating agent behaviour. If an agent *commits* itself to perform an action, this obligation constrains its decision about future actions, and enables other agents to make assumptions about the actions of this agent in the community. Commitments are associated with *conventions*, which describe the circumstances under which an agent should reconsider its commitments, and indicate the appropriate actions an agent should undertake to abandon a commitment.

In particular, when agents decide to cooperate in some activity, they must have a joint commitment towards a common goal and share social conventions, which specify how agents should behave with respect to other agents participating in the cooperative activity when their commitments change.

In short, commitments and conventions provide the necessary requirements for coordinated activity in dynamic environments, since commitments provide the support for predictable interactions, and conventions provide flexibility of behaviour (in that when conditions change, commitments should be revised and changed if appropriate).

2.3.3 Norms

Norms are established, expected patterns of behaviour. Human societies have numerous types of norms, which define expected behaviour in many situations. For example, in places such as banks and bus stops, people are expected to form queues by having new arrivals join the rear of the queue and wait for their turn; in restaurants, people are expected to follow etiquette norms during their meals. These are social norms, which are not enforced in any way, but establish a pattern of acceptable behaviour that helps individuals to self-regulate. If someone is driving a car, he or she is expected to follow the norms of the road and to have a driving licence, a legal norm defined by the state and enforced by severe penalties. In the same way that norms coordinate human behaviour, they can be used to coordinate agent behaviour, by establishing expected behaviour and by constraining individual freedom.

The implementation of norms can be achieved by using pre-designed rules that are hardwired into agents. For dynamic and open systems, however, this approach is not suitable if changes in the environment can require the creation of new rules or the modification or deletion of existing rules. In addition, an agent that joins a new community in an open system should be able to adopt the norms of that community. Indeed, Conte *et al* (1998) argue that, to cope with dynamism and openness, agents must be able to recognise a norm as a normative structure, and to decide whether to adopt a norm, a process that is known as *norm acceptance*. In addition to deciding upon norm acceptance and adoption, Dignum (1999) argues that agents in dynamic environments should also have the autonomy to decide for themselves whether to violate a norm (for example, when an agent faces a dishonest partner and wants to break the conventions to protect its interests). Flexible normative structures can thus help agents to cope with dynamic and open environments for effective coordination through norms. Lopez y Lopez *et al* (2005) describe a normative framework for open societies to cope with heterogeneity and diversity of interests among their members. This is achieved with a model of norms, a model of normative multi-agent systems and a model of normative autonomous agents.

2.3.4 Organisational structures

Organisational structures can be used in multi-agent systems, as in human societies, to regulate behaviour and to specify interaction patterns for promoting coordination (Dignum and Dignum, 2001). By constraining the possible interactions between agents, an organisation can enforce global order and goals. In dynamic and open systems, modeling organisations requires the organisation structure and rules to be independent from the agents' internal structure, so that they are able to join new organisations and internalise their structure and rules.

Dignum *et al* (2002) present a framework for designing organisations in which possible interactions are independent of the internal design of the agent, and organisational characteristics are integrated with agent goals such that their autonomy is preserved. The framework assumes that a set of roles and norms is designed for an organisation, representing its goals and structure, and that roles are fulfilled by agents which behave according to their roles, and commit to the goals they are expected to achieve. In this framework, agents commit to roles by means of contracts.

2.4 Cooperation

Cooperation is the process through which agents choose to work together to achieve a common goal (Wooldridge and Jennings, 1999). In an ideal world, in which all agents are benevolent and willing to help others, cooperation is straightforward. In most real and open systems in which agents have individual interests, this assumption is rarely valid, however, and mechanisms or models to motivate cooperative behaviour are required.

Some theories behind cooperation are inspired by economics, where the major contribution comes from game theory (Wellman, 1995; Parsons and Jennings, 1996). Although this provides a good mathematical and practical framework, it is mostly directed towards applications in which agents get some economical gain from a cooperation. It also limits the agents' motivations to utility measures and does not consider other aspects of cooperative behaviour in society (for example, taking into account reciprocation, values, and social norms) (Castelfranchi and Conte, 1998).

Alternatives to economical approaches to cooperation include those inspired by evolutionary processes (Riolo *et al*, 2001), and those in which cooperation is based on *reciprocity.* In the latter approach, there are different ways to achieve reciprocity among autonomous agents. For example, social incentives can be provided for agents to cooperate with others (Glass and Grosz, 2000), such as to increase the priority of cooperative agents when they need to compete with others to execute some task. Also, agents can use expectations of future interactions as an incentive to reciprocate (Sen *et al*, 2003; Banerjee *et al*, 2005). Reciprocity can also be enforced by norms and organisational structures (Dignum and Dignum, 2003). These different approaches to cooperation are described next.

2.4.1 Evolutionary Approaches to Cooperation

Evolutionary approaches use cooperation to evolve from interactions among self-interested agents. The basic idea is that agents interact with each other using different strategies to choose whether to cooperate. Each combination of the choices of the agents results in an interaction payoff for the agents. Evolution occurs by replicating successful strategies (that is, those resulting in higher payoffs) in the agent population. Agents therefore have an incentive to cooperate if cooperative strategies are successful. The challenge is then to develop cooperative strategies capable of outperforming non-cooperative strategies.

In (Hales and Arteconi, 2006), a dynamic algorithm is presented, which executes in individual entities in a distributed system. It uses social *tag*, a mark attached to an entity that is visible to others (Hales and Edmonds, 2005), and simple evolutionary techniques to achieve cooperative behaviour among entities, without explicit reciprocity. Each entity has a tag and a strategy (which here can be cooperate or defect), and entities with identical tags are seen as forming a group. The algorithm consists of choosing a pair of entities within the system with similar tags, and comparing the performance of the original entity with the chosen one. If the chosen entity has a higher performance, the original entity copies its strategy. The algorithm also makes random changes to an entity's strategy with low probability (in order to mimic the process of evolutionary mutation). The fundamental idea is that cooperative groups are formed, and these outperform non-cooperative groups, and thus pass their cooperative strategy on to other entities in the system. This algorithm was applied to nodes in a simulated peer-to-peer system and was shown to maintain high levels of cooperation among nodes. A similar approach that studies the establishment of cooperation without reciprocity, through tags, is presented in (Riolo *et al*, 2001).

A distinct approach appears in (Feldman *et al*, 2004), in which cooperative strategies are based on reciprocity. Here agents use their history of other agents' actions to decide whether to cooperate. This decision is based on a *generosity* measure for the partner agent, which represents the benefit that the partner agent has provided in relation to the benefit it has consumed. Agents interact in pairs and use their strategies to decide whether to cooperate. Then, agents replace their strategies with the most successful one and apply a mutation to that strategy with a low probability. This approach achieves high levels of cooperation when applied to computational entities in an open system.

2.4.2 Social Incentives and Reciprocity

Alternatives to evolutionary approaches, in which cooperative behaviour is achieved through repeated interactions, include consideration of social incentives, norms, and expected reciprocity in the agents' decision-making over interactions.

Glass and Grosz (2000) present a decision-making model in which monetary factors and social non-monetary utility are weighted to reconcile the agent's social intention with individual interests when participating in collaborative work group. Here, agents have a set of tasks to complete as part of the group activity, and a set of external tasks as part of their individual activities, which conflict with group activities. The proposed socially aware decision-making mechanism is applied to determining whether to defect from group activities and complete personal tasks. It uses a *brownie point model* to support the consideration of social factors in decision-making, calculated based not only on the utility of the task it is considering defecting from, but also on the utility of the outside offer it is considering accepting, and on the agent's history of collaboration (the more the agent has collaborated in the past, the less it will punish itself for defecting).

Norms and organisational structures can be used to enforce reciprocation. Dignum (1999), for example, proposes the use of contracts and norms to determine the obligations and punishments for agents regarding a cooperation. In this sense, norms such as "cooperate with others, if possible" or "reciprocate to others" can result in cooperative behaviour. Of course, to adopt such norms, autonomous agents must be able to reason about the norms and decide whether to adopt or violate those norms (Conte *et al*, 1998). Similarly, Dignum and Dignum (2003) present a multi-agent organisational model in which reciprocity is enforced by concrete and explicit commitments, which establish what each agent is supposed to contribute to, and expects from, a cooperation.

The analysis of reciprocal interactions as an incentive to cooperate with other agents is proposed by Sen *et al* (2003) and Banerjee *et al* (2005) in the form of expected utility-based decision-making. According to this approach, agents agree to cooperate if the cost of helping the requester agent is smaller than the expected benefit of receiving help from the requester and other agents in the future. By considering expected future help in the providers' utility function, agents are motivated to cooperate with each other since the probability of receiving resources increases with the number of times they help others.

Once cooperative behaviour is motivated, agents can form cooperations to achieve individual or common goals. The issues related to the formation of cooperations and existing approaches to form successful cooperations are described next.

2.5 Forming Cooperations

Agents with a range of abilities and resources usually co-exist in open distributed systems. Therefore, agents have the option to join, or to form, alternative cooperations or virtual organisations (VOs) (Norman *et al*, 2004). Some cooperations will be more successful than others, in the sense that they will result in higher or more stable benefits for their members. In multi-agent systems, partnership formation mechanisms and models are concerned with optimising the formation of cooperations.

We are concerned here with cooperations among two agents, in which one performs some service on the other's behalf. In this space, different approaches towards selecting a cooperation partner have been proposed, both in multi-agent systems and in service-oriented literature. While the former refers to partner selection, the latter refers to service selection approaches. From now on we will use the terms, partners and services, interchangeably, meaning that a service can also be viewed as an agent or can be provided by an agent.

Some approaches to partner selection use static selection (like (Stevens *et al*, 2003) and (Gao *et al*, 2005)), in which partners are pre-selected for each needed task, while others use dynamic selection, in which partners are selected at execution time according to information about the state and properties of both partners involved in the cooperation (that is, the service receiver and the service provider).

In dynamic and open systems, the selection of partners must be dynamic in order to cope with changes in the environment and the services that they provide, and the incorporation of new services. A straightforward approach to dynamic selection is to define a set of quality attributes desired for the service provider (e.g., cost and execution time for the service), and to compare it with the set of attributes provided (estimated) by the service providers to find which of them has the best desired attributes (Goble *et al*, 2003). Although this approach is fairly simple, it has at least two limitations: first, one cannot guarantee that the information given by service providers is correct, and second, even if the information is correct in one specific situation, it can be erroneous in other situations (e.g., when a provider has less computer power available due to other processes being executed at the same time, or when messages take longer to reach target machines due to intensive network traffic). Instead, personal experiences with service providers, such as prior service performance and quality, can help to get more accurate information over service attributes when selecting in dynamic and open systems.

In addition to information on service properties, partner selection mechanisms for cooperative systems in which services are free of charge, also consider the cooperative relationships between requester and providers. This is because providers are not obliged to accept requests, and thus requesters rely on their cooperative relationships with providers to find those that are more likely to accept their requests. Since cooperation is not inherent to an agent or any other computational entity, but is commonly found in human social theory, partner selection mechanisms for cooperative systems often use social-related concepts to underpin their models.

In this context, we review in the next sections examples of partner selection mechanisms designed for dynamic systems, which use prior information about service properties (evaluation-based and similarity based selection) and service reliability (trust-based selection), and selection mechanisms for cooperative systems, which use social-related concepts to identify cooperative relationships between providers and requesters (dependence-based selection).

2.5.1 Evaluation-based Selection

If we assume that services (or providers) present a certain regularity of behaviour, such that a service that performed well in the past tends to perform well in the future, then service selection can be based on the evaluation of services received in previous interactions.

In (Casati *et al*, 2004) a service selection platform is proposed which is based on probabilistic and context-sensitive information over services. It uses a service evaluation mechanism together with context information to generate a classification ranking for service providers. Service evaluation is based on user-defined quality goals and on the results of monitoring information: after the user identifies quality goals and metrics, the system deploys a monitoring tool to log conversations between users and providers. This conversation data is then labelled with quality measures as a function of the metrics defined by the user (either by defining a function to generate the measures, or by getting explicit quality measures as feedback from users). Once quality measures are available, service execution data (conversations) can be mined to build a set of models that identify (rank) those service providers that historically, in analogous situations, have provided high quality measures. Selection models are decision trees that, for each context, classify service providers according to the quality measure, and within this, providers are ranked according to the probability of achieving that measure in the specific context. The service provider that fits the quality measure and has the greatest probability of achieving that measure is selected. Although this approach uses more reliable information about services, since service results are monitored and evaluated in comparison to user goals, it assumes a fixed set of service providers, but it is not clear how this set is updated.

A similar approach, which considers prior information about services but also information provided by third-party clients, is presented by Day and Deters (2004). Their approach is to augment the client side, so that it can reason about different service providers and choose the best for its needs. The client selects the best service based on its prior experience with services available in the system, as well as the experiences reported by other clients. The system works as follows: the results of interactions between clients and web services (i.e., values for selected properties of the services) are represented in what are called *semantic models*, which are reported by the clients to a quality of service (QoS) forum; these semantic models are then retrieved and analysed by a reasoning mechanism on the augmented client. However, the system is not scalable due to the size of the client's semantic models archive, which is costly for reporting to the QoS forum.

2.5.2 Similarity-based Selection

A different problem in service selection is addressed by Caverlee *et al* (2004), who describe a group of techniques used to discover, evaluate and rank web services (the targets) according to their similarity with respect to a known service (the source). Their system, BASIL, and its components, comprise a technique to generate service summaries (which indicate the frequency of specific terms in each document returned by the web service), and metrics for measuring the relevance of a web service (target) in comparison to a known service (source).

To evaluate a target service, the system compares the presence or absence of a set of terms in the source service documents with the target service documents. The idea is that the closer the frequency of terms in a target service is to the frequency of terms in the source service, the more related these two services are. Using this similarity measure, target services can then be ranked to find those more relevant to the source service. This method can be used to complement more common evaluation schemes, such as those evaluating an individual service by comparing its expected and delivered quality of service, for example.

2.5.3 Trust-based Selection

In open and dynamic systems with self-interested agents, even if a client and a provider establish an agreement for the service being provided, it is not guaranteed that the provider will actually perform the service, or that the service will be delivered with the agreed quality. To avoid such unsuccessful service provision, agents can select between service providers by using *trust-based models*, which define trust metrics indicating the reliability of providers. Trust-based selection is based on the principle that, if an agent trusts another agent to perform a service, it believes that the latter has the ability and willingness to do it, and thus a trusted provider is preferred over a distrusted one when selecting providers. Related to the concept of trust is that of *reputation*, which is the opinion of others about an individual (so that an agent trusts another agent if the latter has a good reputation).

Sabater and Sierra (2001) and Birk (2001) propose trust models that use prior experience with interaction partners to derive trust and reputation metrics. Birk describes an evolutionary-based approach for learning to trust agents based on their trustworthiness as interaction partners, such that after repeated interactions, agents learn whether to trust other agents (by generating a trust metric), and use this information to decide in which social interactions to engage. In the REGRET system proposed by Sabater and Sierra (2001), trust and reputation metrics are derived not only from the past experiences of the requesting agent with interaction partners, but also from those of others. Although the REGRET system provides a sophisticated model of trust (in that it considers an agent's own prior experiences and the prior experiences of others), it assumes that the information that an agent receives from another which belongs to the same group or VO is always true.

To cope with the case of agents giving false information about their experiences with others, Teacy *et al* (2005) propose a trust model for partner selection based on probability theory in which agents can cope with inaccurate reputation sources by filtering out opinions provided by such sources. In their approach, the final trust metric avoids bias by agents that provide false reputation information.

2.5.4 Dependence-based Selection

In a cooperative system in which agents need to request services from others, one way to achieve cooperation is to observe the dependencies between requesters and providers. The concept of *social dependence* in multi-agent systems was first proposed by Castelfranchi (Castelfranchi, 1990; Castelfranchi *et al*, 1992), for whom dependence represents the situation in which an individual needs a resource (such as an object, a task to be performed, or a piece of information) that it does not possess or have access to, but which can be provided by another, giving the latter influence over the former. The basic definition of social dependence is that an agent x *depends* on an agent y regarding an action a needed for achieving a goal g, if x is not capable of performing a but y is. In this case, agent y's action a is viewed by agent x as a resource for achieving g. When agents x and y depend on each other, there is a *bilateral dependence* between them, and when agent x depends on agent y but y does not depend on x, there is a *unilateral dependence* between them.

Associated with this is the notion of *power of influence* of one agent over another (Castelfranchi, 1990). Dependence relations can be associated with the power of influence in the sense that when agent x depends on agent y for achieving a goal, it becomes susceptible to the influence of the latter agent. In this view, agents structure their interactions to take advantage of situations in which they can exert influence to ensure the compliance of agents with their requests. Consequently, reasoning about these relationships allows agents to achieve better interactions, since an agent is able to control its interactions according to its own interests, by obtaining power through dependence relations.

Partner selection and coalition formation mechanisms that take into account the dependence of services between agents are proposed by Sichman *et al* (1994) and David *et al* (2001). The principle here is that cooperation takes place between two or more agents if they can provide services to each other (they depend on each other for one or more services that they need). Thus, agents with a bilateral dependence are more likely to interact with each other.

When the actual formation of a cooperation requires that agents reach agreements over conflicting interests or over their responsibilities in the cooperative relationship, the selection of partners also involves a negotiation process. We discuss in the next section the selection of partners through negotiation.

2.6 Negotiation

As agents have individual, sometimes conflicting goals and vary in their abilities to perform tasks, the efficiency of a cooperation requires that agents *negotiate* with each other in order to reach agreements and achieve their goals in the best possible way. A similar view is presented by Jennings *et al* (1998), in which negotiation is defined as a method for coordination and conflict resolution: coordination in the sense that agents frequently need to resolve goal disparities in planning (such as simultaneous access to the same resource), and conflict resolution in the sense that agents need to solve problems that arise from inter-dependencies.

Beer *et al* (1999) identify three broad topics for research on negotiation: negotiation protocols, negotiation objects, and reasoning models for negotiation. Negotiation protocols state a set of rules to be applied to the negotiation process between agents, while negotiation objects are the issues related to the agreement to be made, which can concern price, timing, availability, or even access permission. Finally, reasoning models are concerned with the decision-making necessary for agents to reach agreements, and their complexity depends on the protocol used and the range of aspects related to the negotiation object. In the next sections, we describe common negotiation protocols for multi-agent systems and the decisions required to negotiate in each protocol.

2.6.1 Contract Net Protocol

A classic example of a negotiation protocol is the Contract Net Protocol (Smith, 1981), in which an agent can dynamically find a task provider to which it awards a contract. Agents can assume two roles in the protocol: *manager* or *contractor*, with no restrictions on the number or type of role. A manager agent looks for a task provider to which to award a contract, and a contractor agent offers a task it is capable of performing.

The protocol is initiated by the manager agent, which announces a needed task and its desired specifications by broadcasting a message to other agents in the system. The contractor agents that are capable of performing the task send bids to the manager agent with their specifications for the task. After receiving a number of bids, or after a timeout, the manager analyses the received bids, and awards a task contract to the best bidder.

Because of its simplicity, the Contract Net Protocol is widely used for negotiation in multi-agent systems. However, it has some performance limitations, as follows:

- because it assumes the manager does not inform bidders when it rejects a bid, contractors lose time waiting for a reply;
- because managers do not evaluate the performed task, they may continue awarding contracts to low quality contractors; and
- since the manager must send messages to all agents in the system and all of them must reply, a multi-agent system with hundreds of agents generates heavy traffic.

The protocol also has limitations for self-interested agents, since it assumes that potential contractors always send bids unless they are not eligible for a task, but agents may prefer not to perform tasks for others.

2.6.2 Auctions

An auction is a negotiation process between an *auctioneer*, which wants to sell or allocate an item, and a group of agents called *bidders*, which want to acquire the item. An example of an auction system is the internet auction site *eBay*.

Both auctioneers and bidders define strategies to optimise their goals, which aim, respectively, to get the highest possible price for an item, and to pay the lowest possible price for an item. Auction protocols have a number of settings, which specify how the price to be paid by the winner bidder is determined, what kind of knowledge the bidders have about the bid, and how the highest bid is determined (Wooldridge, 2002). For example, depending on how the price is determined, the agent placing the highest bid pays the amount of its bid (first-price auction), or it pays the amount of the second highest bid (second-price auction). Also, regarding the knowledge of the bid, agents might be able to see each others' bids (open cry auction), or not (sealed bid auction). Finally, depending on the auction protocol, the highest bid is determined after one round of bidding (one shot auction), or the highest bid is determined after successive bidding (ascending or descending) until bids stop.

The different combinations of the settings above form different types of auctions, of which the most common are the following:

- English auctions, which are first-price, open cry, ascending auctions;
- Dutch auctions, which are first-price, open cry, descending auctions; and
- Vickrey auctions, which are second-price, sealed bid, one-shot auctions.

To negotiate using an auction protocol, agents use strategies that depend on the protocol settings and type of auction used.

2.6.3 Agreements and Negotiation for Service-oriented Systems

Negotiation is currently being applied to the service-oriented domain to provide a more formal relationship between service providers and consumers, so that consumers have guarantees related to the quality of the services they use. In this context, a theoretical view of negotiation and its issues is presented by Elfatatry and Layzell (2004), who describe negotiation in service-oriented systems as a type of interaction aimed at establishing the users' needs dynamically. They argue that negotiation should be designed as a three-phase process: pre-negotiation, negotiation, and service delivery. The pre-negotiation phase involves the tasks of: service selection, which is based on the analysis of different services and their attributes; provider selection; and prediction of service usage, which is the anticipation of the use of further services to avoid loss in performance due to negotiation procedures. The negotiation phase involves the exchange of messages and contract templates between the parties with the aim of reaching an agreement, with the interaction being governed by the rules of a negotiation protocol. If the parties agree upon a contract, the service delivery phase starts, and involves the implementation of what the parties agreed in the contract.

In the particular case of web services, the web services community has proposed a specification, *WS-Agreement* (Andrieux *et al*, 2004), to provide a unified language and protocol for advertising the capabilities of providers, creating agreements, and monitoring agreement compliance at runtime. An agreement can provide, for example, guarantees on the bounds of service execution time and availability, or on the availability of minimum resources like memory and CPU.

According to the WS-Agreement specification, an agreement includes information on agreement parties, reference to previous agreements, reference to the agreement context, service definition terms, and guarantee terms. It also includes definition of agreement templates and protocols for creating agreements. Protocols for negotiating agreements are not part of the WS-Agreement specification, but the specification must be used as a basis for designing negotiation protocols.

A concrete effort to implement a negotiation model for operating with web services is presented in (Hung *et al*, 2004), which discusses a negotiation model for web service providers and clients, and proposes a declarative XML language, WS-Negotiation, for automating the negotiation process between service providers and clients. The proposed negotiation model has three components:

- a negotiation message, describing the format of the messages exchanged between participants;
- a negotiation protocol, describing the steps and rules the participants should follow; and
- a negotiation decision-making process, which is implemented as an internal decision process of each negotiation party and uses a cost-benefit model.

According to the model, the two parties first negotiate over service requirements (like price, response time and availability), and when the negotiation is finished, a service-level agreement (SLA) document is created, containing the guarantees and obligations of the negotiation parties.

2.7 Limitations of Current Approaches

While this review makes clear that there are readily available techniques for achieving cooperative behaviour among agents in a multi-agent system and for selecting interaction partners, in this thesis we focus on a different problem that has not generally been addressed. In particular, we are concerned with agents operating in open cooperative systems. We assume that such agents are self-interested, in the sense that they might not always be willing to cooperate with others due to individual interests or resource limitations and, additionally, that services are not always priced and may be provided free of charge.

Effective cooperation in this context requires cooperative behaviour to be motivated, so that agents have the incentive to cooperate even in the absence of any economic gain, and agents requesting services do not have to rely on altruism to guarantee service provision. Moreover, since agents in open systems are highly heterogeneous, with different skills and preferences, it is not desirable for all possible cooperations to take place. Therefore, agents need some means for choosing between alternative cooperations.

Although there are various approaches to achieving cooperative behaviour among self-interested agents without economic compensation, they have some limitations. Evolutionary approaches do not consider the real motivations of agents to cooperate with others, nor the properties of alternative partners (since in most approaches partners are selected at random (Feldman *et al*, 2004) or are selected among agents from the same group (Riolo *et al*, 2001)). Therefore, they do not provide any grounds for an agent to make autonomous decisions over interactions.

Most existing approaches using social incentives and reciprocity provide explicit incentives for agents to cooperative with others, but they do not account for the differences in the properties of alternative partners. In particular, in the brownie point approach (Glass and Grosz, 2000), although there is an incentive for group cooperation, there is no concrete benefit for the agent in gaining a brownie point when it depends on others to execute its *own* tasks. This is because brownie points represent an agent's self-valorisation (the agent rewards itself for cooperating in a team) and not the valorisation an agent receives from others in response to the provided service, which could then be used to receive a service in reciprocation in the future.

In utility-based decision-making, which considers expectations of future interactions (Sen *et al*, 2003), agents that depend on each other have the incentive to cooperate to improve expectations of future interactions. However, this approach does not consider the success of the cooperative relations in terms of the counterbalance between provided and received services, which is important if the environment has agents with different preferences and perspectives, or even agents that reciprocate but by providing low quality services. Finally, regarding normative and organisation-based approaches (Dignum, 1999; Dignum and Dignum, 2003), we argue that instead of being used to motivate cooperations, they are more appropriate to *enforce* reciprocation by ensuring that agents do reciprocate through norms, contracts, and so on.

Regarding the problem of forming cooperations, current mechanisms consider either information on the properties of provided services (such as service evaluation and similarity) (Casati *et al*, 2004; Caverlee *et al*, 2004) or reciprocal relationships between providers and requesters (Sichman *et al*, 1994; David *et al*, 2001). However, in an open cooperative system, both types of information are relevant when choosing a cooperation partner, and there is no attempt to balance this information in a single partner selection mechanism.

A key challenge in open systems with free services, therefore, is to enable cooperative behaviour of self-interested providers and requesters to result from their autonomous decision-making. This requires a means not only to provide *non-monetary incentives* for service providers, but also to account for the level of *quality of cooperations*, in terms of the quality of provided and received services. There is also a need for selection mechanisms that *balance* aspects of reciprocation, service quality, and resource limitations, each of which is relevant to cooperation in the kinds of systems we consider. In the rest of this thesis, we develop mechanisms and models to meet this challenge.

Chapter 3

Problem Scenario

3.1 Introduction

Computational systems in which participants share personal tools and data, are becoming very popular, both in commercial and academic communities. Computer programs with different purposes and functionalities are constantly being produced, as well as data of different kinds (including scientific publications, newspapers, experimental results in physics, biology, computer science, and so on). Instead of being just for personal use, these tools and data can be made accessible to others as *services* in a distributed system, so that a participant in such a system can make requests from a remote service and receive results after the service has completed. With such systems, participants can gain access to services they would not otherwise be able to access if they were in an isolated and closed system, and can request services at the time they are needed, without being connected to just one service all the time. Such characteristics are specially desirable for application domains that are constantly changing, since newly discovered data and newly developed tools can be made available to others.

One such domain is bioinformatics, which has seen an explosion in the number of developed services since the start of genome sequencing projects all over the world. Indeed, it continues to see an increase in the number of services being developed for more specific areas like proteomics and drug discovery. In addition, in many bioinformatics laboratories, unique data sets are being created that are not published in public databases, but could usefully be shared with the global community. Conversely, the number of services available in closed bioinformatics systems is limited when compared with the variety of services that are available in the global open community. In particular, access to a wider range of services, including *private* tools and databases, can facilitate the search for, and use of, more suitable services, of better quality, in order to improve the results of bioinformatics experiments more generally.

Although there are benefits for participants to have the opportunity to interact with a large number of providers and requesters, it is not desirable for all possible interactions to take place. First, because services with different characteristics are available, not all of them will have the same quality or will take the same time to return results. With some services being better than others, it is important that participants are able to choose those with better properties, like higher quality or smaller response time. Thus, when a participant requesting a service can find several alternatives with similar functionalities, some mechanism is needed to select the service most suitable for its needs, for example in terms of performance, quality, and speed. In addition, when participants in a distributed application are self-interested, service providers need some compensation to be given in return for their effort and investment in performing a service, and requesters need to be sensitive to this need in order to be able to find a service provider willing to accept its request in a timely fashion.

Since participants providing services may receive many requests from others, and performing such requests may be computationally costly (as in the case of bioinformatics services which usually involve processing large amounts of data), from a provider's perspective it is necessary to limit service provision to avoid being overwhelmed with services to provide for others. In addition, providers must have the autonomy to decide whether to accept requests at all. Thus, when a participant providing a service receives more requests than its available computational power or wants to select which interactions to engage in, some mechanism is necessary to choose between incoming requests.

In this chapter we investigate open cooperative systems in the context of bioinformatics, which we use as a problem scenario, since it offers characteristics of dynamism, service variety, and resource constraints, giving rise to the problems in requesting and providing services described above.

The chapter is organised as follows. We first introduce the bioinformatics domain and discuss the particular area of proteomics in Section 3.2. Next, we present an overview of existing computational tools used in bioinformatics research in Section 3.3, and the key distributed applications that have been proposed to integrate bioinformatics tools in Section 3.4. Our bioinformatics application scenario is presented in Section 3.5. We then identify the requirements for the effective operation of the application scenario in Section 3.6, and discuss the key problems to be addressed in the target domain. Finally, we conclude in Section 3.7.

3.2 Bioinformatics and Proteomics

Bioinformatics is a new field of research characterised by the application of computer technology to the management and analysis of biological data (i.e., to gather, store, analyse and merge genome related information).

The development of this new research area was motivated by the beginning of several sequencing projects for genomes of various organisms such as small bacteria, viruses, insects, plans, mammals and, most importantly, the *Human Genome Project* (The Wellcome Trust, 2001). The amount of data generated by genome sequencing projects grew so fast that manual analysis became almost impossible, demanding the assistance of computational tools.

The fundamental goal of bioinformatics is to uncover the wealth of biological information hidden in the mass of sequence data produced by genome sequencing projects. With this knowledge, scientists can obtain a clearer insight into the fundamental biology of organisms to elucidate complex biological processes that are not yet completely understood, such as the transcription and translation processes, gene expression, protein folding, and the way genes, proteins, and the cell interact with each other. Providing such knowledge of gene and protein-related processes, bioinformatics ultimately aims to produce better and more customised medicines to prevent or cure diseases. Other areas that can also benefit from genomic information and, consequently, from bioinformatics research, include research on environmental issues (like the identification of bacteria for naturally clean waste) and on agriculture (such as the production of high yield and low maintenance crops).

Although the sequencing of entire genomes from various organisms has been a great step forward towards achieving the goal of discovering the richness of biological information hidden in genome sequences, the raw data that has been generated cannot be used without attaching to it relevant biological meaning (i.e., whether a stretch of DNA contains an amino acid coding sequence, or a regulatory sequence and, if an amino acid is coded, its biological function, and so on). As Kim (2002) states, the next step in bioinformatics research is to "*synthesize information into knowledge*".

The challenge facing genome scientists now is to make sense of the wealth of data that has been produced by genome sequencing projects. The huge amount of raw biological data has increased the complexity of annotation processes (in attaching relevant biological meaning to new data) and, as a consequence, has led to a change in the way scientists annotate genome sequences: from manual experimentation and human-based analysis to computer-based analysis.

Although there are still many challenges related to genomics, several of the technological barriers to obtaining genomic information seem to be solved, since entire genomes can be sequenced in a very short time with high-throughput sequencing techniques. Because of this, many believe we are now in the *post genomic era* (Tyers and Mann, 2003), in which the next challenge is to understand the *proteome*: the collection of all proteins of a given organism (Campbell and Heyer, 2002). Different from DNA, proteins have a dynamic nature: while each cell has the same genome, each cell type in an organism has a different proteome, and every given cell changes its proteome over time (for example, due to ageing, infections, and even diet and medication). The study of the proteome on a high-throughput scale is called *proteomics*, which also includes the study of protein interactions, protein modifications, and protein structure. Since proteins are gene products, the study of the proteome is only possible due to the achievements of genomics.

In what follows, we first describe software tools used in bioinformatics, and then applications for integrating and sharing those tools.

3.3 Bioinformatics Tools

To clarify the function of software tools used in bioinformatics and the original need that motivated their development, we can separate them into two categories: *prediction tools* and *analysis tools*.

Prediction tools are used to find new biological information with the application of particular prediction models, based on specific biological knowledge. This is the case, for example, with *gene prediction tools*, which provide information on gene location based on genetic patterns for coding regions in DNA sequences.

On the other hand, analysis tools use existing information stored in biological databases to infer knowledge about novel molecular sequences. This is the case with *search engine tools*, which search sequences of *unknown* structure and function against biological databases to find similarities with sequences whose structure and function are already known.

Prediction and analysis tools are largely used in the process of genome annotation, which consists of attaching relevant biological knowledge to unknown DNA sequences, by combining genomic information from different databases. In proteomics, analysis tools are used as part of the process of *protein identification*. In the next sections we describe prediction and analysis tools in more detail, and discuss the characteristics of the biological databases used by these tools.

3.3.1 Prediction Tools

Once scientists have determined the genome sequence of an organism, they need to find where genes[1] are located, as well as the sequences within genes that code for proteins (the *open reading frames*, ORF). In most organisms this process is not trivial, because the DNA sequences are composed of both coding sequences (*exons*) and noncoding sequences (*introns*). The introns are spliced out before the sequence is mapped into amino acids,

[1]A single gene can range in length from as few as 100 DNA bases to as many as several million.

so the exons are the segments of DNA that actually end up coding for a protein[2]. To find the regions of the gene that code for proteins, scientists look for a variety of *signals* in the genetic code that indicate where the coding regions begin (indicated by a *start codon*) and end (indicated by a *stop codon*), and where splices should occur. Many software tools have been developed to help in gene prediction including, for example, *GeneMark*[3] (Borodovsky and McIninch, 1993), *Glimmer*[4] (Salzberg *et al*, 1998), and *GenScan*[5] (Burge and Karlin, 1997).

3.3.2 Analysis Tools

When scientists isolate a new molecular sequence through laboratory experiments, they want to know all relevant biological information about that sequence. The first thing to do is to determine if a similar sequence has been already discovered and annotated (Phizicky *et al*, 2003). This is achieved by analysis tools.

Analysis tools operate as search engines by comparing some target sequence against one or more biological databases to find similarities in homology, structure or function. They operate under the premise that if two DNA sequences have a similar combination of nucleotides, they probably have similar function and structure, even if they come from different organisms or different cells (and the same premise is applied to protein sequences but here for similar combinations of peptide sequences). Probably the most used computational tool to perform this task is BLAST (Basic Local Alignment Search Tool)[6] (Altschul *et al*, 1990), which searches databases like GenBank and Swiss-Prot for all sequences similar to the target sequence. In the case of search engines for protein identification, known as MS/MS *search engines*, the input data is a set of peptide sequences (the MS/MS spectrum)[7], and the result is a list of candidate proteins matching the peptides, each associated with a score and the number of peptides that matched the protein (the higher the number of matching peptides, the higher the confidence that the protein is present in the unknown protein mixture).

Different search engine services have different properties relating to, for example, the databases that are being searched, the search algorithm used, the scoring system that calculates a measure of how well the sequence matches the database, and the returned information, which can be in many different formats. These different properties can lead to differences in the quality of results since, for example, a specific scoring system can

[2]According to The Wellcome Trust (2001), less than 2% of the human genome contains actually protein coding regions.

[3]Available at the EBI website http://www.ebi.ac.uk/genemark/

[4]Available at the NCBI website http://www.ncbi.nlm.nih.gov/genomes/MICROBES/glimmer_3.cgi

[5]Available at http://genes.mit.edu/GENSCAN.html

[6]Available at the NCBI website http://130.14.29.110/BLAST/

[7]MS/MS spectrum is a specific type of input received by search engines, from mass spectrometry machines.

return a more precise evaluation of the candidate proteins, and one search algorithm can be faster than another.

Search engines also have several configuration parameters, which are fairly similar for those engines with similar functionality and type of input data, and define the search space for the query sequence. In particular, for MS/MS search engines, these search spaces include: *taxonomy* (organism classification), *peptide tolerance* (the error window for experimental peptide mass values), and number of *missed cleavages* (peptides are fragmented with an enzyme which breaks peptide bonds in specific sites, and this measure indicates the number of allowed missed breaks during digestion). The significance of configurations is that search results can be influenced by different configuration parameters. For example, if the peptide tolerance is set to a high value, this can result in a higher number of false matches, since the comparison window for the peptide mass is bigger. Conversely, if the peptide tolerance is set to a low value, it can result in the loss of true matches. In practice, bioinformatics experts adjust these parameters according to their individual preferences, or to the quality of the data produced by the mass spectrometer, so that if peptide masses are accurate and not approximations, parameters are set to narrow the search space, while if they are approximations, parameters are set to increase the search space.

Alternative MS/MS search engines are publicly available, and differ from each other generally in the implementation of the matching algorithm. Examples of such search engines are Mascot (Perkins *et al*, 1999), Tandem (Craig and Beavis, 2003), and OMSSA (Geer *et al*, 2004), some of which run on remote servers, while others run as local services. Although these are alternative MS/MS search engines with the same functionality, they can yield heterogeneous results for the same input data. As a consequence, some services may be more suitable for data with a certain quality or for a particular configuration setting than others. This means that, even if one search engine performs better when using a particular configuration setting, it may vary its performance when used with a different configuration setting.

3.3.3 Biological Databases

Bioinformatics databases store information related to the genes and proteins of organisms, and are tailored to particular types of information or organisms. For example, they can store DNA sequences, protein sequences or entire genomes, or they can store biological information that is related to particular organisms such as humans, mice, fruit flies, viruses and so on. Most data that is stored in bioinformatics databases is annotated, with relevant biological information attached to it, for example indicating to which organism the sequence belongs, or which gene or protein that sequence is related to, and so on.

When existing bioinformatics databases do not contain any matching proteins to an unknown protein sequence, a new database (known as a *six-frame database*) can be created by translating DNA sequences from the organism associated with the protein mixture directly into protein sequences. Although the protein sequences in a six-frame database are not known, a match indicates that the protein exists, but has not yet been annotated, or at least not in publicly available databases.

3.4 Distributed Applications in Bioinformatics

In this section, we describe the key approaches to integrating bioinformatics tools from distributed locations. Multi-agent systems have been used as a technology to integrate heterogeneous data and tools, while grid systems have been used to integrate tools and data for execution as distributed workflows. In addition, cooperative systems have been proposed to allow bioinformatics researchers to share their tools and data. We consider each in turn below.

3.4.1 Multi-agent Systems Applications

Agent-based systems are one of the technologies that can be used to help in solving problems related to biological data generated by genome projects. Distributed, heterogeneous, and dynamic environments, as with the biological domain, are commonly the target domains of agent-based applications. Thus, some key problems of bioinformatics research, like integrating information that is distributed in remote, heterogeneous biological databases over the Internet, and keeping track of existing and updated bioinformatics software and data, make the agent approach very suitable if we view each distributed bioinformatics site, tool or data provider and user as agents. However, the idea of applying agents to tackling key issues of bioinformatics research is still very new and, as a consequence, there are many problems to be investigated.

Nevertheless, some work has already been done in the development of multi-agent system tools for use in prediction of secondary structure proteins (Armano *et al*, 2005), disease gene discovery (Williams *et al*, 2001), and automatic data integration (Karasavvas *et al*, 2002). In particular, the pioneering applications of agents in bioinformatics are described by Bryson *et al* (2000) and Decker *et al* (2002), with a focus on data integration and genome annotation. These are described in the next sections.

3.4.1.1 GeneWeaver

GeneWeaver (Bryson *et al*, 2000) is a multi-agent system designed to tackle problems relating to the integration of genome analysis and structure prediction tools. It is argued that the distributed, heterogeneous, dynamic character of biological information, together with the existence of several types of analysis and prediction programs to be applied to this information, points to the suitability of an agent approach. Here, the multi-agent system comprises a community of agents with distinct functionalities that work together to automate the annotation of genomic data. Agent functionalities are determined according to the tasks that need to be accomplished during the annotation process.

There are five types of agents in the GeneWeaver community: *broker agents*, *primary database agents*, *non-redundant database agents*, *calculation agents*, and *genome agents*. The *broker agent* is responsible for storing information (such as their location, supported communication methods, and abilities) about all the agents in the community. *Primary database agents* are in charge of managing primary sequence databases like Swiss-Prot, PDB, and PIR. Similar to primary database agents, *genome agents* and *non-redundant database agents* are also responsible for managing genome information, the main difference being that genome agents are responsible for controlling information about the genome for a particular organism. Finally, *calculation agents* encapsulate existing software applications used to analyse biological data, so that each program becomes an independent agent in the GeneWeaver community.

Agents communicate with each other within the GeneWeaver community using a specific language based on KQML, the *BioAgent Language* (BAL). BAL messages contain language and ontology fields to help agents understand the content of the message. The meta-data, data, and query expressions in the content field are represented by the *BioAgent Content Language* (BACL). Also, two ontology sets are defined: the *BioAgent Meta Ontology* (BAMO), which defines different types of meta-data and their meanings, and the *BioAgent Data Ontology* (BADO), which defines the data types employed.

GeneWeaver does not introduce new methods or techniques for performing any task related to genomic data annotation, but organises and manages existing ones so that they can operate in a more flexible, and more effective way.

3.4.1.2 BioMAS

Decker *et al* (2002) present a multi-agent system for automated genomic annotation. Their BioMAS system is an extension of previous work (Decker *et al*, 2001) on automated annotation and database storage of sequencing data for the *herpesvirues*, which was expanded to a more generic system that can be used for studying more organisms. The new system also includes extensions for functional annotation, Expressed Sequence Tags (EST)[8] processing and metabolic pathway reasoning.

[8]Expressed Sequence Tag is a small sequence from an expressed gene, and acts as a physical marker for cloning and full length sequencing of the DNA of expressed genes (Lopez, 2003).

The system is composed of four overlapping multi-agent subsystems: *basic sequence annotation*, *query processing*, *functional annotation*, and *EST processing*. The function of the basic sequence annotation and query processing subsystems are, respectively, to integrate remote gene sequence annotations from various sources, and to allow complex queries on local databases via a web interface. The functional annotation subsystem is in charge of assisting the user to make functional annotations of each gene in a sequenced genome, by using Gene Ontology (GO)[9] (The Gene Ontology Consortium, 2000) for annotating gene function. The EST processing subsystem was designed to support the use of expressed sequence tags as input data in the annotation process, in addition to complete sequences of nucleotides or proteins.

There are three types of agents in the system: *information extraction agents*, *task agents*, and *interface agents*. The first group of agents is responsible for *wrapping* public databases like Genbank, Swiss-Prot, PSort and ProDomain. Agents in the second group are divided into: domain-specific agents, which include annotation agents, responsible for guiding the annotation process, and sequence source processing agents, responsible for checking the consistency of sequence format; and domain-independent task agents, which include proxy and matchmaker agents, responsible for facilitating the communication within the system. Interface agents are responsible for helping the user to add new sequences to the local knowledge base, and to query complete annotated knowledge bases.

3.4.2 Grid Applications: myGrid

Moreau *et al* (2002) describe some possible uses of agent technologies in an e-Science Grid project with a focus on bioinformatics, myGrid (Goble *et al*, 2003). This project aims to provide a distributed environment that supports the construction of *in silico* experiments, which are represented by workflows, and can be stored, shared and managed according to user preferences. Other complementary features include the notification to the user of relevant information related to their experiments, and the provision of assistance for less skilled users to manage their experiments.

myGrid has a service-oriented architecture, and provides support for users to create, discover and execute workflows. Services and workflows have semantic descriptions, indicating their functionality, the types of input they require, and the types of output they produce (Lord *et al*, 2003). User discovery of workflows and services is achieved via semantic services (McIlraith *et al*, 2001), which use matching algorithms to search through semantic descriptions for services or workflows compatible with the user query (i.e., preferences, goals, etc). As a result of the discovery process, the user is presented with a list of available services from which they can choose.

[9]http://www.geneontology.org/

The use of agents in this bioinformatics grid aims at addressing a common problem in bioinformatics research, the constant change in resources available to the bioscientist (i.e., their continuous appearance, disappearance, or change without prior notification). Agents are seen as an appropriate technology to tackle this problem since they provide an abstraction for the design of scalable systems, as well as the means to implement aspects like personalisation, communication, and negotiation within the grid environment.

Two types of agents have been defined to act in the grid: a *user agent* and a *broker agent*. The *user agent* is responsible for *representing the user* within the myGrid system, which includes providing the user's personal preferences for other parts of the system, and mediating the communication between grid services and the user. Negotiation within myGrid takes place on the basis of preferred *quality of service* for service providers and service users, in the context of notification support. The agent responsible for managing these negotiations is the quality of service *broker*, which negotiates on behalf of each service user that wants to receive notifications of a specified quality, and then returns a final proposal.

According to Foster *et al* (2004) in relation to the mutual benefits of combining grid and agent-based systems, distributed bioinformatics applications may also be improved by joining grid and agent-based technologies. This would enable *in silico* experiments to be conducted and controlled in a more flexible way in both individual and collaborative work.

3.4.3 Cooperative Applications

Bioinformatics researchers are discovering the advantages of cooperative research, in which different types of information and tools are exchanged in order to improve individual or global results. Here, unique data sets are created in individual laboratories, and not published on public database sites. However, they could be shared with a world-wide community if provided with the right tools to support cooperation. The systems described in the previous sections are mostly concerned with integrating heterogeneous data and tools, or combining remote data and tools for execution in distributed work-flows, but they do not address cooperation explicitly (since the tools and data that are integrated or combined typically belong to the same individual or group).

In an effort to provide such a cooperation support tool, Overbeek *et al* (2004) present a peer-to-peer environment for genome annotation, SEED, which allows researchers to combine publicly available genomic data with individual, non-public data exchanged with other researchers to form an integrated and distributed curated database of genomic data. Each SEED instance has a copy of this integrated database and is a self-contained genome annotation system that allows multiple users to access, update, and extend the annotation database. To support cooperative work, the SEED system uses a peer-to-peer synchronisation facility that permits information sharing between SEED instances. Cooperation members are known (i.e., access is not anonymous), and have the option of choosing whether to participate in an annotation team. Although the system provides support for data exchange, it does not address the problem of selecting between different SEED users and instead it assumes that the user must select candidate SEEDs from a registry to send data requests. Also, it is not clear how annotation groups are formed, whether by finding users with related interests, or by other criteria.

A different approach to supporting cooperative research that uses a web services solution is presented by Gao *et al* (2005), who develop a microarray[10] data-mining system that uses web services in drug discovery. The system is implemented by wrapping data processing modules and databases into web services, integrating them, and providing a portal through which the user can select and aggregate services. A limitation of this approach is the lack of support for the automatic use of services, which is assumed to be carried out by the user, and for the analysis of the quality of the provided services.

Given this overview of bioinformatics tools and key approaches to integrating and sharing bioinformatics tools and data, in the next sections we describe the application scenario, which we take as a case study through the thesis.

3.5 Protein Identification through Cooperative Bioinformatics Applications

Most experiments in biological sciences are *in vivo*, which are performed on living organisms, like tissues and cells. Nowadays, however, parts of these experiments are also performed on computers, as *in silico* experiments. Such experiments may be composed of a single service or of a group of related services that follow a computation sequence to reach a final result. In the latter case, the computation sequence is usually modelled as a workflow, which determines the inputs and outputs that are passed from one service to another. *In silico* experiments are often static and closed, in the sense that the services that compose them are fixed and maintained only in a local system. Such computer-based experiments are used in many sub-areas of bioinformatics including proteomics, metabolomics and genetics (Campbell and Heyer, 2002). Probably the best known computer-based bioinformatics experiments are those for genome sequencing and decodification, like the Human Genome Project initiative, which started in 1990.

In what follows, we focus on proteomics applications, in which the primary goal is to identify proteins from unknown mixtures. Once this is achieved, bioinformaticians can compare proteins from different mixtures and discover, for example, if there is a difference in the set of proteins of a normal person from those of a diseased person (in

[10]DNA microarray is a technique used to simultaneously measure the level of transcription of many genes (Campbell and Heyer, 2002).

order to understand the disease), or if a specific medication changes the set of proteins in a person under treatment (to understand how the treatment evolves)(Hanash, 2003). In the next sections we describe a typical protein identification experiment, discuss its limitations, and then present a distributed scenario to address those limitations and improve experimental results.

3.5.1 Protein Identification Experiments

A primary goal of proteomics is to identify and quantify every protein in a cell at a given time. The most common approach to protein identification is via *mass spectrometry* (MS) followed by sequence database searches (Campbell and Heyer, 2002). Here, the mass spectrometer quantifies protein sequences according to a *mass to charge ratio*, and breaks them into smaller fragments called *peptides*, which are also quantified with a mass to charge ratio. Peptide fragments from each protein sequence are compared against a database of known protein sequences, with the best matches used to identify the original protein.

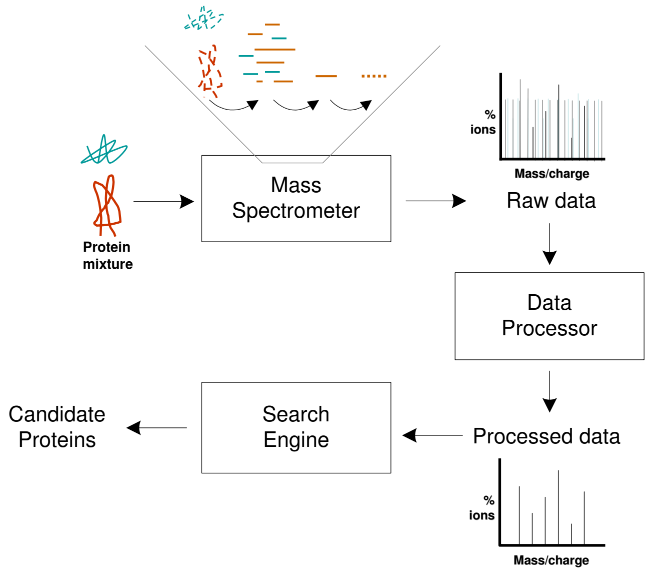


FIGURE 3.1: Basic protein identification experiment workflow.

In a typical *protein identification* experiment, an unknown protein mixture is received as input, and is processed and analysed to generate possible identifications for that protein. For example, as illustrated in Figure 3.1, a protein mixture is passed to a mass spectrometer, in which the peptides that form the protein in the mixture are subjected to *fragmentation* (in order to reduce the complexity of the samples), so that the mass of the fragments, given as a *mass per charge ratio*, can be used as a *peptide fingerprint* (since each of the amino-acids that forms a peptide has a specific mass) through which the peptides and the original protein they compose can be identified (Chamrad *et al*, 2003; Edwards and Lippert, 2002). The mass spectrometer analyses the fragments of one peptide at a time to produce a measure for the peptide mass-charge ratio (or *mass spectrum*). The end result of the mass spectrometry analysis is a set of hundreds to thousands of mass spectra, each representing the fingerprint of a peptide from the original mixture.

After mass identification, the mass spectra of each peptide is interpreted to identify the corresponding amino-acid sequence, a process known as *data processing*. Amino-acid sequences returned from data processing of each peptide are compared against known amino-acid sequences through a database search engine (such as those discussed in Section 3.3.2) to find the peptide candidates and their associated proteins, together with an evaluation of how well the peptides match the database. Reliable protein identification requires the amino-acid sequence of many peptides from the original mixture, so the higher the number of identified peptides that are associated with the same protein, the higher the chance that this protein was present in the original mixture. If only few peptides are identified for a protein, further data analysis may be needed.

Although the process of protein sequencing and quantification via mass spectrometry is automatic, interpretation of MS data results from protein database search engines is still undertaken manually by experts in many bioinformatics laboratories. Since there is much data to be interpreted, bioinformaticians usually simplify the search process and do not use new tools, or experiment with alternative tools that could give more confidence to the data with better quality of results. Even though the interpretation of MS data is a bottleneck in the protein identification process, little work has been done to address this problem (Chamrad *et al*, 2003).

Based on the limitations of current protein identification experiments, we describe a distributed bioinformatics scenario and discuss the advantages of having a cooperative bioinformatics system in the next section.

3.5.2 Distributed Services

In the basic protein identification experiment workflow (depicted in Figure 3.1), services are pre-defined for each task, and there is no support for service variety. Although there are several alternative services (as described in Section 3.3.2) that might perform the task of database search and could increase the confidence and quality of the resulting data, these services are not used in the basic experiment configuration.

Traditional systems to manage computer experiments use static workflows, in which services are pre-selected for each needed task (as in (Stevens *et al*, 2003; Gao *et al*, 2005)), and are suitable for domains with a limited number of services. However, bioinformatics applications can be expanded such that alternative services can be requested from remote

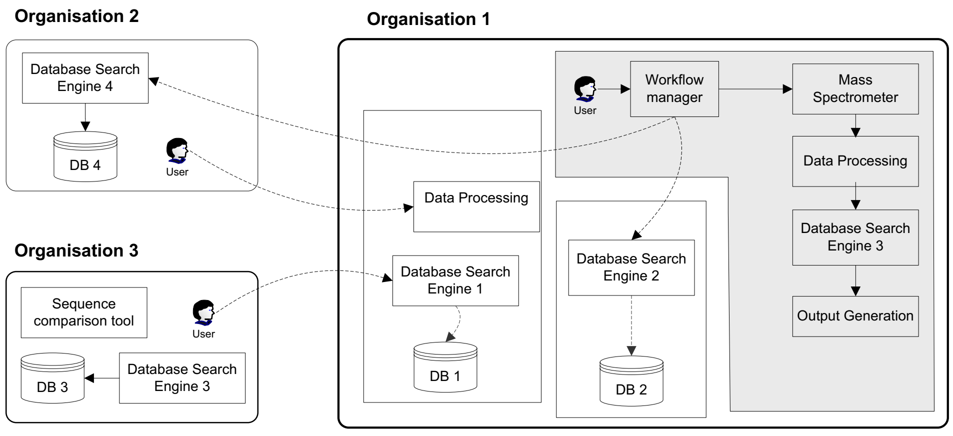


FIGURE 3.2: Closed (highlighted) and distributed scenario for execution of computer experiments.

sources in a distributed system. In particular, having access to a wider range of services, including such private tools and databases, can facilitate the search for, and use of, more suitable services, or services of better quality, in order to improve the results of bioinformatics experiments more generally.

Indeed, aware of the great variety of tools and databases being developed by private individuals and organisations, both commercial and academic bioinformatics communities are focusing their interest on open distributed systems, like that illustrated in Figure 3.2, in which participants *request services from, and provide services to, each other*, so that they can access services such as different search engines and annotated databases resulting from private experiments, that they would not be able to access if they were in an isolated and closed system. The figure shows the situation of a distributed system in which users belonging to a specific organisation can access services such as databases, data processing and sequence comparison tools that are available in a different organisation. Thus, to make use of the global availability and variety of services, we must allow bioinformaticians to interact with other entities (including individual researchers, laboratories, and companies) in a distributed fashion, to provide access to many different types of information and tool, and consequently to improve individual and global results. To do this, however, we must first analyse the way services are shared in the bioinformatics domain. That is, we must determine if there is an economic motivation for service provision or if services are available free of charge, since this has implications for the way in which interactions among participants may take place.

Currently, there are many bioinformatics services that are available free of charge, such as the main public databases for searching genome and proteome related data, like

GeneBank[11], EMBL[12], and NCBI[13]. There are also paid services developed by private companies (such as commercial search engine tools ProteinLynx and Phenyx). However, current proposals that envision the creation of a global bioinformatics community generally adopt cooperative approaches (Stein *et al*, 2001; Stein, 2002; Overbeek *et al*, 2004; Ellisman *et al*, 2004; Gao *et al*, 2005), in which the participants of the cooperative global community share data and tools without economic motivation (because services have no cost), but with the aim of improving individual and global results and discoveries. This suggests that research in this domain has a cooperative character.

3.6 Requirements for Cooperative Bioinformatics Applications

As described in the protein identification scenario, one of the key characteristics of bioinformatics is service variety, not only because there are different services with similar functionality, but also because services may give different performance as a consequence of their properties. For example, since each *configuration* of a search engine can lead to results of different quality, each possible configuration of a search engine can be viewed as a separate service in the selection process. This significantly increases the alternatives for instantiating a service, as in the case of search engines.

Service variety is even wider if we consider services in an open environment (as shown in Figure 3.2). In this case, the number of possible interactions between participants increases not only because (in the role of requesters) they find more service providers to send requests, but also because (in the role of providers) they can receive requests from other participants. In an ideal situation in which computational resources are plenty, individuals could interact with each other without any restriction on the number of services they could provide, the number of authorised accesses to individual databases, or the number of messages sent to service providers. However, as in the majority of real systems, not only are there limited computational resources, and participants must restrict the number of requests they accept, but participants also have their own interests (they are autonomous) and, in the absence of any economic return from service provision, they may choose not to provide services to others.

In addition, from a requester's perspective, service variety means that service quality may also vary from one provider to another (due to different characteristics or skills, for example), which suggests that the selection of a service provider should not be a random choice if the requester needs good quality services.

In this context, to allow participants of computer-based open applications to interact in

[11]http://www.ncbi.nlm.nih.gov/Genbank/
[12]http://www.ebi.ac.uk/embl/
[13]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Protein

a cooperative manner with the aim of sharing their data and tools, and to choose the interactions they want to participate in, the following are required.

1. A mechanism for automatically selecting between alternative services when several with similar functionality but different quality of results are available.
2. A means to incentivise service provision to support cooperation when services are free of charge.
3. A mechanism for analysing incoming requests so that providers can choose for whom to provide services, and also limit service provision when resources are scarce.

These requirements are related to three general issues: first, there is a need for non-monetary incentives to motivate cooperation; second, when participants in open cooperative systems request a service and must select alternative interaction partners, they need a way to find the providers they consider to be more desirable (in terms of quality, performance, etc); and third, when participants provide a service and must decide whether to accept an incoming request, they need to determine the incentive for providing a service and consider their resource availability. These issues are discussed in the next sections.

3.6.1 The Cooperation Problem

Current applications to support distributed bioinformatics, as described in Section 3.4 generally address the problems of integrating heterogeneous data and tools (Bryson *et al*, 2000; Decker *et al*, 2002) or combining different data and tools in a single distributed bioinformatics experiment (Goble *et al*, 2003; Lord *et al*, 2003), and do not focus on how cooperation is achieved between participants. Moreover, systems that do support cooperation between participants so that they can exchange data and tools (such as (Overbeek *et al*, 2004; Gao *et al*, 2005)), assume that providers always cooperate, and when there is any restriction on service provision it is limited to requesters from a particular group or team (which may not apply to an open system in which participants are not always connected to such groups or teams).

In contrast, we consider open bioinformatics applications in which participants are autonomous and thus must choose whether to cooperate, based on, for example, their available resources and the incentives for service provision. The problem in this context is that, since participants are not considered benevolent and there is no monetary compensation for providing services, motivating service providers to cooperate with others is difficult. Since providing a service always incurs some cost, it is likely that a system will have more agents requesting free services than providing them. This disparity may compromise the purpose of open cooperative bioinformatics systems, to allow participants to have access to a greater variety of services.

Therefore, there is a need for *non-monetary incentives* for service providers, so that they are motivated to cooperate and service requesters do not have to rely on altruism to guarantee service provision. Moreover, with increased service variety, it is likely that some cooperations will be more beneficial for an agent than others. For example, an agent may receive a service of better quality when cooperating with agent A than with agent B, or an agent may provide a service of better quality than the service it receives from its interaction partner. The cooperation problem thus requires agents not only having non-monetary incentives to cooperate, but also *a means to analyse cooperations.*

3.6.2 The Provider Selection Problem

When participants in a distributed system need to request a service and find many alternative services with similar functionalities, they need a way to determine which provider is the best interaction partner according to some preference, depending on their experience or current goals.[14] For example, one participant might prefer to use a service from a specific provider or with specific properties depending on the goal (like choosing a service with a lower execution time for experiment testing, or a more accurate service for an experiment with high quality data input, etc). This suggests the need for a *provider selection mechanism.* In addition, the task of selecting providers becomes more relevant when the services that are available in the distributed application are heterogeneous in terms of quality of results, as in the case of bioinformatics services.

Selection is also influenced by the cooperative character of global bioinformatics communities, since participants requesting services need to consider the provider's motivation for accepting a request. Otherwise, requests may take too long to be accepted, which can delay the performance of experiments.

Thus, a selection mechanism for bioinformatics services must consider the following:

- service quality and confidence of results, since service requesters need to be aware of interaction outcomes in order to decide whether interactions should be started or continued;
- the use of subjective criteria in decision-making, since preferences and goals influence the relevance of different service attributes; and

[14]We distinguish provider selection from the task of discovery of the available providers for a particular service. We assume that selection takes place after discovery, over the set of possible candidate services returned. In this way, the discovery mechanism filters the available services according to their functionalities to generate a set of candidates, and the selection mechanism compares candidates according to preferences.

- cooperative relationships with providers and how these influence their decision to accept a request to perform a service.

If these requirements are met, the provider selection mechanism can be used in real cooperative applications, in which participants have their own interests, care about service quality, and deal with limitations in service provision when it is not paid for.

3.6.3 The Requester Selection Problem

Biological data used in bioinformatics research is very interrelated, in the sense that one type of data, say nucleotide sequences (genes), may provide hints about another type of data, like amino acid sequences (proteins). For distributed bioinformatics applications, this means that many participants may be interested in similar services and, as a consequence, that providers of those services may receive several requests from others.

In addition, we assume that participants of such distributed systems are autonomous and are not obliged always to cooperate with others, so they must decide *whether* to accept requests. Since we focus on cooperative applications with free services, this decision is not influenced by monetary compensation, but may involve other elements found in cooperative relationships, like reciprocity and informal commitments, and the quality of interactions like, for example, whether an agent requesting a service is also a good service provider. This suggests the need for a *requester selection mechanism.*

In summary, a mechanism to select requesters must consider the following:

- the future benefits brought by non-monetary incentives in terms of finding available cooperation partners in the future;
- the strategic restriction of service provision to minimise the impact of request denial on the chances of future interactions; and
- the quality of interactions, so that the quality of services to be received in the future is not lower than the quality of services provided.

Meeting these requirements results in an effective requester selection mechanism for cooperative applications with free services and resource limitations, with which self-interested providers can be modelled and cooperation can be considered as part of the provider's decision making.

3.7 Conclusion

In this chapter we have identified the requirements for designing open cooperative applications, in particular in the bioinformatics domain, especially the requirements related to the tasks of requesting and providing free services when providers are self-interested and resource-bounded.

In particular, current applications to support distributed bioinformatics do not address the problem of cooperation among participants of the distributed system, and generally assume that providers always cooperate. However, the purpose of open cooperative bioinformatics systems, to allow participants to have access to a greater variety of services, requires *non-monetary incentives* for service providers, so that they are motivated to cooperate and service requesters do not have to rely on altruism to guarantee service provision. Moreover, with increasing service variety, there is a need to *evaluate the properties* of different services to guide future choice of alternative interaction partners.

In addition to service variety, interactions among participants are influenced by the cooperative character of global bioinformatics communities. Since we assume that providers are self-interested and resource-bounded, and thus are not obliged to accept all requests, participants of open cooperative systems need to *select* among alternative interaction partners, both when they are providing and receiving services. In the light of the requirements mentioned above, in the next chapters we propose a set of solutions for designing agents operating in open cooperative applications.

Chapter 4

Agent Architecture

4.1 Introduction

The description of the problem scenario and the identified issues to deal with cooperation and service request and provision in open cooperative systems, presented in the previous chapter, suggest that entities providing and requesting services need to have at least two internal mechanisms to support interactions: a *requester selection mechanism*, which is responsible for decision-making over service provision; and a *provider selection mechanism*, which is responsible for decision-making over service requests. In addition, both decision-making processes must take into account *cooperative interactions* and *service quality*. To provide a unified solution for effective partner selection and cooperation, we propose in this chapter an architecture for entities participating as providers and requesters in cooperative distributed applications.

The chapter starts with an overview of the proposed architecture in Section 4.2. The individual components of the architecture are described in following sections, with the framework for non-monetary cooperations presented in Section 4.3, the provider selection mechanism presented in Section 4.4, the requester selection mechanism presented in Section 4.5, and the evaluation method presented in Section 4.6. The interaction process among entities providing and requesting services is described in Section 4.7, and we conclude in Section 4.8.

4.2 Architecture Overview

For effective partner selection and cooperation in open cooperative applications, we propose an architecture, shown in Figure 4.1, composed of the following modules and process.

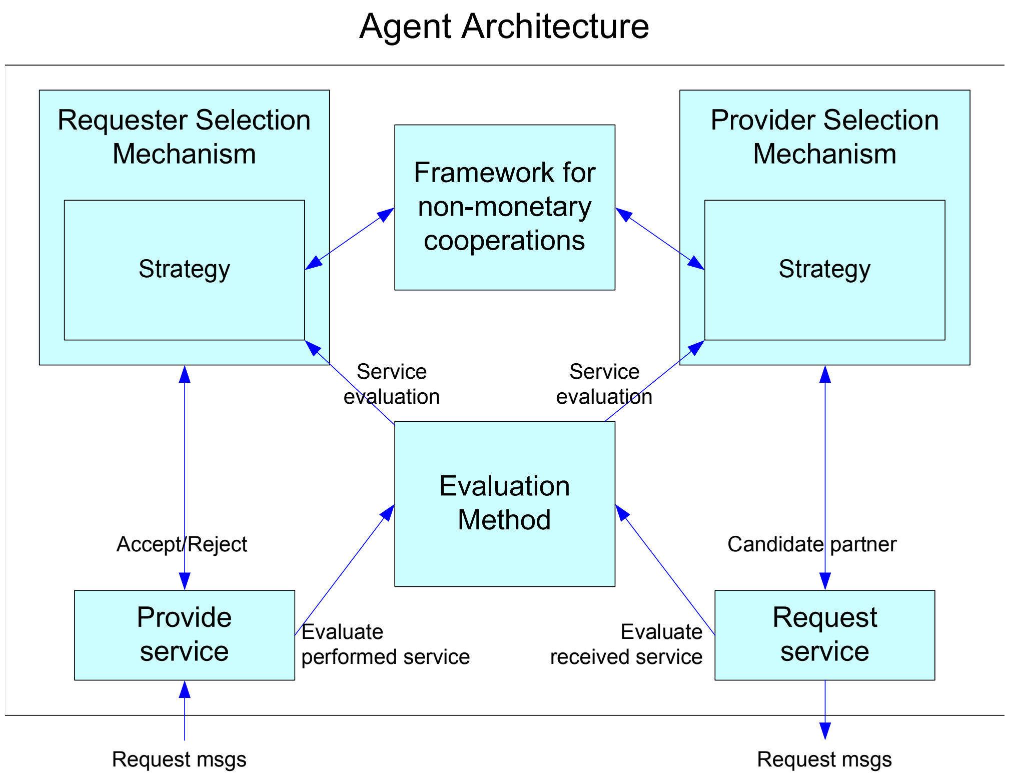


FIGURE 4.1: Agent architecture for service provision and request.

- A *framework for non-monetary cooperative interactions*, which incentivises self-interested agents to provide services, and provides a means for agents to analyse the cooperative interactions in which they engage.
- A *provider selection mechanism*, which is responsible for selecting between candidate providers when a service is needed.
- A *requester selection mechanism*, which is responsible for restricting service provision when resources are scarce, and for analysing incoming requests to decide whether to accept a request.
- An *evaluation method*, which is responsible for generating evaluations for services, so that evaluations of previous results can be used for future selection. Both provider and requester selection mechanisms can use previous service evaluations during the selection process to ensure interactions with good results are continued, and those with poor results are avoided.
- An *interaction process*, which determines how provider and requester interact by using the modules outlined above.

The modules composing the architecture and the interaction process are described in more detail in the next sections.

4.3 Framework for Non-monetary Cooperations

We propose a framework for non-monetary interactions in which the incentive to cooperate comes from *reciprocal relationships.* Since reciprocity and cooperation are not inherent to an agent or any other computational entity, we need computational techniques to achieve such relationships between participants in the distributed community. Given that reciprocity and cooperation are commonly found in human social theory, we aim to use *social techniques* to incentivise service provision and to allow computational entities to reason about cooperation when choosing interaction partners.

Thus, social techniques are used here to encourage the establishment and maintenance of reciprocal interactions between self-interested agents in the open community as part of their behaviour. Agents need a means to choose to maintain or terminate a cooperation; and to make such decisions, agents need to be aware of their reciprocal relationships with others.

Once information on reciprocal relationships with other agents in the system is available, its influences on both the provider and requester selection mechanisms are as follows:

- it influences service provision, since providers are motivated to perform services for others by means of their expectations of getting services in return in the future;
- it influences the choice of requests to be accepted, since providers must constrain service provision due to resource limitations, but must also consider the fact that denying a request can negatively influence the possibilities of future interactions with the requester in a cooperative environment; and
- it influences the choice of alternative providers, since requesters must consider their reciprocal relationships or informal commitments with candidate providers when selecting among them so that they can find those more likely to cooperate; otherwise, requests may take too long to be accepted.

4.4 Provider Selection Mechanism

For effective provider selection in real cooperative applications, like that described in Section 3.5, we propose a provider selection mechanism that supports the task of finding suitable providers, with the characteristics below.

- The selection process is repeated every time a service is needed and considers the performance of services received in previous interactions, which we term *dynamic selection with analysis of service evaluations*, to cope with providers joining or leaving the system over time, and with changes in service performance that may occur from one execution to another.

- Existing or potential reciprocal relationships with candidate providers are taken into account to find those more likely to accept requests, which we call *analysis of cooperative relationships*, so as to avoid requests that may take a long time to be accepted.

Cooperative relationships are analysed with the help of social techniques, and the analysis of service performance comes from previous evaluations of the service, which are generated by an agent's evaluation method.

4.5 Requester Selection Mechanism

For the task of choosing which incoming request should be attended to, we propose a requester selection mechanism with the characteristics below.

- Analysis of *cooperative relationships*, which considers possible informal commitments with reciprocation when deciding whether to accept a request.
- Analysis of *service performance*, which considers the evaluation of previous service results, aiming at avoiding interactions that are not beneficial for the provider in terms of the effort and investment made.

As for the provider selection mechanism, cooperative relationships are analysed with the help of social techniques, and information on service performance is generated through an agent's evaluation method.

4.6 Evaluation Method

The aim of the evaluation method is to generate an evaluation for the service result that can be used as a quality parameter for the service. This quality parameter can be defined in terms of both the satisfaction of the requester and the effort of the provider. Once such an evaluation is available, participants can use it during selection, through their provider selection and requester selection mechanisms, to avoid continuing interactions with poor outcomes (for example, if a participant is receiving from another participant a service of inferior quality than the one it is providing to that same participant). Agents may use different evaluation methods, as long as they follow the characteristics and functionalities required to achieve the purpose of the evaluation method mentioned above, and to cope with the properties of dynamic services and of open systems, which will be discussed later in the thesis.

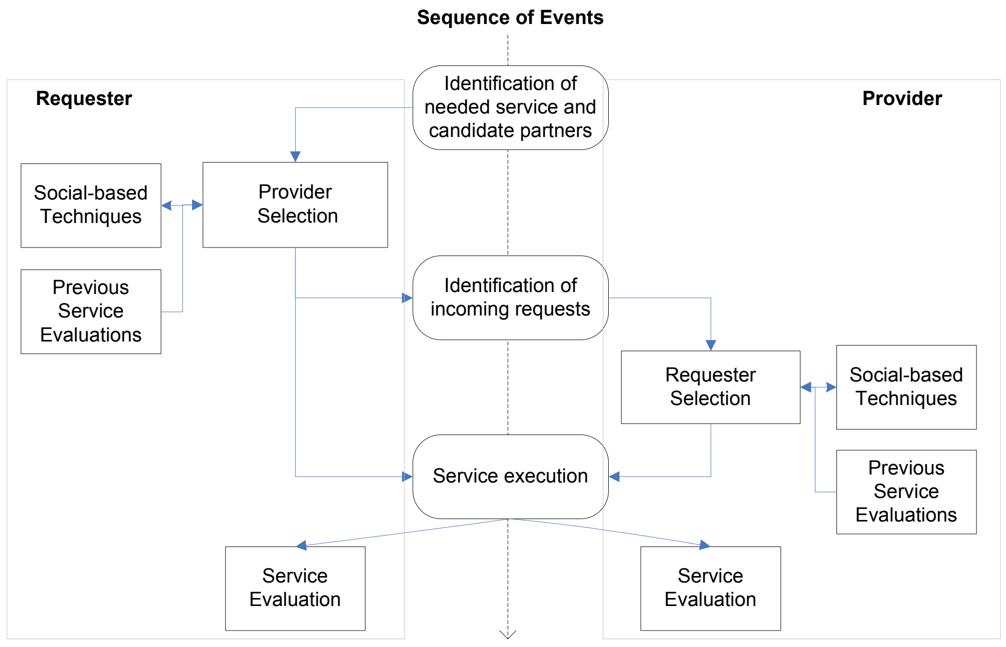


FIGURE 4.2: The interaction process sequence of events. Events flow from top to bottom.

4.7 The Interaction Process

Providers and requesters interact with each other through an *interaction protocol*, which specifies the steps to follow during the selection process and the types of messages they exchange. The interaction process starts when a service request is sent from a requester to a provider, as described in the top of Figure 4.2. When an agent identifies a needed service and has a set of candidate providers that can perform that service, it uses the provider selection mechanism to find the one to which it will send the request. Similarly, when an agent receives a number of requests from other agents to perform some service on their behalf, it uses the requester selection mechanism to choose which requests to accept, taking into account available resources for service provision.

If a request is accepted, the provider performs the service, and the requester waits for the service result from the provider. After service execution is finished, both provider and requester evaluate the service. The interaction process is complete when both evaluations are computed and stored.

4.8 Conclusion

In this chapter we have presented an overview of the proposed architecture for supporting cooperative interactions and the decision-making process of self-interested entities regarding service provision and request in cooperative environments with free services. This architecture is composed of four modules: a framework for non-monetary cooperations, a provider selection mechanism, a requester selection mechanism, and an evaluation method.

The cooperation framework consists of using reciprocity as the key incentive for cooperation in open systems with unpaid services. We propose the use of social techniques to represent reciprocal relationships as part of agent behaviour (instead of being externally imposed by the system). Norms and commitments are not considered here, since they are higher level structures, defining mechanisms to ensure that agents collaborate and reciprocate, but not addressing the basic motivations that direct agents to interact.

The evaluation method aims at providing evaluations for services, so that those with good evaluations can be chosen in future interactions and those with poor evaluations can be avoided. Finally, the provider and requester selection mechanisms support decision-making over interaction partners. To achieve this, these mechanisms use information on service evaluation and reciprocal relationships provided by other components in the agent architecture. The proposed architecture thus combines all the required functionalities for effective cooperation in open cooperative systems.

The next chapters describe in full our proposed solutions, which are represented by each individual component of the architecture.

Chapter 5

Evaluation Method

5.1 Introduction

As described in the previous chapter, in open cooperative systems in which there is a large number and variety of available services, and where service providers can join and leave, it is likely that a participant requesting services will find many alternative providers for similar services. Similarly, there is a good chance that a service provider in such systems will receive many requests from different participants. This requires a means for participants in open systems to choose among interactions partners. Moreover, when available services are not only diverse in terms of functionalities and results, but are also continuously updated, improved or modified, as described in Chapter 3, the problem of choosing alternatives is made even more difficult.

Clearly, the most important thing here is for service users to be aware of the quality of the results they receive, so that they can make better choices of service in future interactions. Then, based on the principle that a provider that has performed well in the past is likely to perform well again in a similar situation, if services manifest a certain regularity of behaviour, an efficient way to select interaction partners is by identifying partners with good outcomes in previous interactions. In order to determine the best outcome in this way, however, service *users* must perform an *evaluation* of services after they are executed and the results are received. This evaluation should reflect the *satisfaction* of the user with the service outcome, in relation to any number of criteria, such as the quality of the interface, the provider's availability to perform the service when requested to do so, the time taken to execute, the quality of the content returned, and so on. Evaluation thus allows a requester to identify, from the set of possible providers, that particular provider with the *best* characteristics, like highest quality of results and lowest time to complete the request.

Similarly, service *providers* can use service evaluation to select which requests to accept. However, unlike the requester's perspective, this evaluation must reflect the effort of the provider with service *execution*, in relation to such things as processing time, memory usage, etc. In this case, for example, the evaluation allows the provider to identify, among all candidate requests, those which, if fulfilled, require less time to perform.

In this way, evaluation provides a *criterion for future decision-making* over alternative interaction partners. Therefore, in this chapter we propose an evaluation method to analyse service outcomes. Such an evaluation is needed to determine how *satisfied* a service user is with a service it has requested and received, and the *effort* invested by a service provider, both of which can be used for future decision-making over alternative interaction partners.

Although the general *functionality* being offered (or requested) in a distributed system can be viewed either as an agent or a service, we distinguish both concepts here such that a *service* is a functionality itself and an *agent* is a service user or provider (or both), so that one agent may provide or request several different services.

The chapter starts in Section 5.2 by analysing the key evaluation properties for dynamic services and discussing alternative approaches to evaluation. We then propose, in Section 5.3, a general evaluation method for dynamic services, and the evaluation process to be followed by service providers and requesters is summarised in Section 5.4. In Section 5.5, we use the proposed evaluation method to evaluate bioinformatics services used for protein identification and discuss the evaluation results. A comparison with similar evaluation methods is presented in Section 5.6, and we conclude in Section 5.7.

5.2 Service Evaluation

5.2.1 Key Evaluation Properties

When evaluating a service, independent of the context or domain in which the evaluation takes place, the first thing to consider is that more than one characteristic may be important to analyse during evaluation, since evaluators may be interested in several different aspects of the service. For example, when evaluating food in a restaurant, customers may take into account, among other things, the quality of the ingredients, the way the food was presented, and the price. In the same way, when evaluating computer services like search engines, users might consider, for example, the time taken to complete the query, the relevance of the content returned to the user in relation to the query, and the way the results were presented. The number of characteristics to be evaluated in a service varies according to the evaluator and the type of the service, in that the more complex the service, the more aspects that might be relevant to observe.

Additional requirements for the evaluation method depend on the characteristics of the services, and on the purpose of the evaluation. In open systems, we can find static services but also many dynamic services, in the sense that they are constantly being updated or modified. In addition, despite manifesting similar behaviour when operating under similar conditions, many services can vary their performance under different conditions. Examples of such services include those in the bioinformatics domain, which are constantly being updated in response to the accumulation of information resulting from genomic and proteomics research. Here, not only is new information continually being uncovered, but service performance varies depending on the *configurations* used or on the quality of the input data (the amount of noise in the spectra). In summary, the fundamental requirement arising from the characteristics of services in open systems is that the evaluation method must use a dynamic evaluation process.

The aim of the evaluation method, to provide a criterion for comparison between alternative interaction partners either when providing or requesting a service, suggests a concern with providing both a general method that can be used by agents acting as providers or requesters to select between potential interaction partners (or services), and a comparable evaluation for partners (or services) over time. Given these observations, we summarise below the key properties of an evaluation method.

1. *Continuity*: the evaluation process must occur each time agents (or services) interact rather than only once, since the performance of services may change from one interaction to another if different input configurations are used, if services have been updated, or if new data has been published.

2. *Generality*: the evaluation method must be general enough to be used by agents in different roles (provider or requester). To cope with an heterogeneous system, having one general method that can be applied in all situations is more advantageous than having several specific methods, since with a general method it is not necessary to define one evaluation method for each agent in a different role and for different types of service. A consequence of this generality is the need to support evaluation according to multiple attributes, since agents in different roles will be interested in evaluating different attributes of the service. For example, when evaluating the effort invested in performing a service, an agent may evaluate memory usage and time taken, while when evaluating a received service the same agent may instead consider the precision and reliability of the service result.

3. *Consistency*: the evaluation method must deliver evaluation measures that are coherent when analysed at different points in time, to allow the correct comparison between evaluations. This is important when interactions between agents are repeated over time under different conditions, and when new services appear.

In the next sections, we describe alternative approaches to service evaluation in relation to these key properties. We then present our general evaluation method for dynamic services.

5.2.2 Alternative Approaches

Traditional evaluation approaches determine the evaluation of a service using scoring or utility functions, which return a quantitative evaluation for the service (Edwards and Newman, 1982; Russell and Norvig, 1995; Yoon and Hwang, 1995). Such utility functions can be calculated based just on observed values, or on the comparison of observed and expected values. In the first case, which we refer to as the *absolute evaluation* approach, utility is derived from values that are observed directly from service outcomes, and the evaluation of a service depends only on its actual performance and is not influenced by *expected* performance (Edwards and Newman, 1982; Yoon and Hwang, 1995). In the second case, which we refer to as the *relative evaluation* approach, the utility of a service or attribute is derived from the comparison of values from the outcome of the service at hand with those of a similar service, or with expected values (Caverlee *et al*, 2004). The difference between these two approaches is that absolute evaluation yields *independent* measures, while relative evaluation renders *comparative* measures which require either information about a similar service, or the identification of ideal or expected performance.

An evaluation method based on either approach can meet the generality and continuity issues described earlier. However, their implications for the consistency requirement need further analysis. In relative evaluation methods, measures are dependent on another service's performance or on an expected value, so if the service used as a comparative basis, or the expected values, change from one evaluation to the other, a comparison of evaluations can lose consistency. In absolute evaluation methods, however, the evaluation process is independent of other services or expected performance, so evaluation measures are not biased. Therefore, to guarantee consistent comparisons between evaluations generated at different points in time, without having to calibrate for changes in expectations, we must use absolute measures rather than relative ones.

In addition, it may not always be possible to identify ideal or expected service results. For example, bioinformatics services are usually applied to discovering new data, which suggests that users often do not know what to expect from the service result. In this way, the need to identify expected measures of performance for services in order to determine relative evaluation measures makes the relative evaluation approach more problematic.

5.3 General Evaluation Method

Given the properties (or non-functional requirements) for service evaluation identified previously, and the discussion of alternative evaluation approaches, we identify the following specific requirements for a general evaluation method for dynamic services, as presented below.

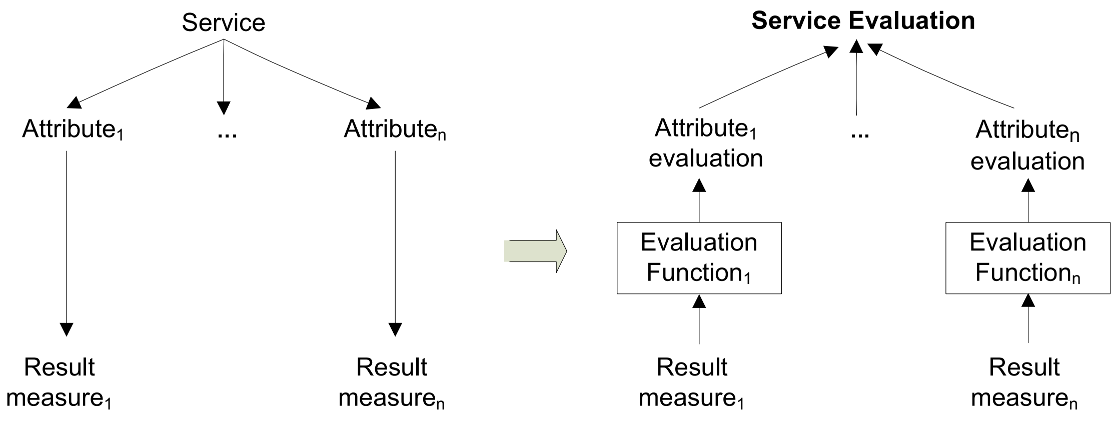


FIGURE 5.1: Service Evaluation Scheme.

- Services must be evaluated every time agents provide or receive a service instead of just once, so that the resulting evaluation can capture any changes that may occur in service execution and results.
- Evaluation must be performed over multiple *attributes* of a service so that agents in different roles, or providing and requesting services with different functionalities, can specify their own set of evaluation attributes, but follow the same evaluation scheme. The only constraint here is on the choice of evaluation attributes, in the sense that services with similar functionality must be evaluated by the same agent using the same set of evaluation attributes, to guarantee a consistent comparison between them.
- All attributes must be evaluated according to an independent evaluation function, which receives as input values that are directly observed from the service outcome, instead of expected or ideal values, following the absolute evaluation approach.

To address these requirements and cope with continuity, generality, and consistency when evaluating dynamic services, we propose a general evaluation method that follows the evaluation scheme described in Figure 5.1. We view services in terms of attributes, which are associated with absolute values observed from service results, which we call *result measures* (as described on the left side of Figure 5.1). From this, as described on the right side of Figure 5.1, the principle is that service evaluation is determined in terms of attribute evaluations, and attributes are evaluated in terms of result measures through the application of evaluation functions. The components and processes that are part of the general evaluation method are presented next.

5.3.1 Evaluation Attributes

The choice of evaluation attributes for a service depends on the service's functionality and on the aspects of the service or service results that the user (or provider) considers important to evaluate. For example, suppose an agent in a cooperative system requests a search engine service and wants to evaluate its results. Traditional web search engine evaluation usually takes into account attributes related to the quality of the search result, such as *precision* and *recall* (Dhyani *et al*, 2002), which measure the ability of the search engine in terms of the relevance of documents retrieved, and the performance of information retrieval (which are important from a user's perspective).

Services can also be evaluated according to their usage, as is the case when applied to the evaluation of web services (Lee *et al*, 2003), which includes evaluation in terms of the reliability of the service (indicating if results are delivered as expected), and its availability (indicating if the service is ready whenever it needs to be used). Although we have mentioned only these attributes for illustrative purposes, an extensive list of possible attributes for information retrieval engines and web services in general can be found in (Dhyani *et al*, 2002) and (Lee *et al*, 2003).

All of these attributes relate to evaluation from a user perspective, but different evaluation attributes must be identified for a provider to evaluate its own performance or execution of a service. Indeed, from a provider's perspective, an important characteristic to be evaluated, for example, is the service processing cost, since this is related to the *effort* of the provider in performing the service. In our proposed evaluation method, attributes are evaluated in terms of result measures, which are described below.

5.3.2 Result measures

Evaluation attributes are measured in terms of computable elements to which they are associated, and which can be directly observed from the service results. For example, the accuracy of a search engine may be measured in terms of the number of matched hits that are related to the input query, while its performance may be measured in terms of the time taken to complete the search. We call these computable elements *result measures*.

Using a more specific view, we define result measures as pieces of information, derived from service results, and which can be used to determine the *service utility*. These result measures are service-dependent, since they relate to the function and purpose of a service. We distinguish between two types of measure, *static measures* and *dynamic measures*, where static measures are those whose values do not change, or rarely change, from one execution to the other, while dynamic measures are those whose values tend to change when the inputs or external conditions vary from one execution to the other. The distinction between these two types of measure is important when considering the frequency of the evaluation process and the characteristics of the services being evaluated.

For example, external conditions such as the size of the input the service received, the number of concurrent jobs the provider is processing simultaneously, the network traffic, and so on, may influence the evaluation of service attributes like *performance*. Thus, service performance must typically be defined in terms of a dynamic result measure, like the time taken to complete a request, or the service processing time, instead of a static measure like processing power, so as to capture result variations.

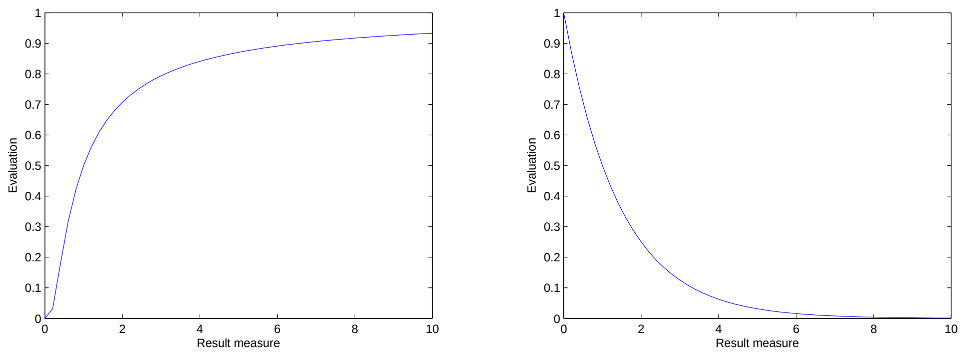
Some attributes, like *accuracy*, may be defined in terms of both static and dynamic measures. When evaluating the accuracy of a database search engine, a static measure might be, for example, whether new entries submitted to the database are verified by human experts through a process known as curation, so that a curated database is considered more accurate than a non-curated one. This measure does not change over time because it is part of the database policy and, thus, can just be evaluated once instead of every time the service is used. A dynamic measure for the search engine's accuracy would be the number of matched items returned by the search algorithm that are not relevant to the input query (the fewer the number of irrelevant matches, the more accurate the search engine). Different from the static measure, the dynamic measure can vary from one execution to another when different input data or parameter configurations are used (and differences can be captured by repeated evaluation).

5.3.3 Evaluation Functions

The result measures of a particular attribute, as described above, are used by evaluation functions to generate an evaluation for that attribute. More specifically, each evaluation attribute of a generic service *srv* has its own evaluation function, which is expressed in the form of a utility function, taking into account the result measures associated with that attribute (as shown on the right side of Figure 5.1).

To represent the set of evaluation attributes that an agent involved in an interaction as a provider (*prv*) or a requester (*req*) wants to evaluate in service *srv*, we use the set $A_{srv}^{prv} = \{a_1, .., a_n\}$ (or A_{srv}^{req} if the agent is a requester), where n is the number of attributes related the *srv*. The choice of evaluation attributes for an agent in different roles is related to the objective of the evaluation. For an agent receiving a service, the evaluation is intended to measure its *satisfaction* with the service results, while for an agent providing a service, the evaluation is intended to measure its *effort* in performing the service. Taking as an example the list of evaluation attributes for service *srv* including *accuracy*, *performance*, *reliability*, and *cost*, a requesting agent would evaluate *srv* in terms of the attributes related to its satisfaction, so that $A_{srv}^{req} = \{accuracy, performance, reliability\}$. On the other hand, a provider would evaluate *srv* in terms of the attributes related to its effort in carrying out execution, so that $A_{srv}^{prv} = \{cost\}$.

Thus, all $a_i \in (A_{srv}^{req} \cup A_{srv}^{prv})$ must have an evaluation function $Aeval_{srv,a_i}$. When more than one attribute is evaluated (as in the example above when *srv* is evaluated from the



(a) Increasing Evaluation Function (b) Decreasing Evaluation Function

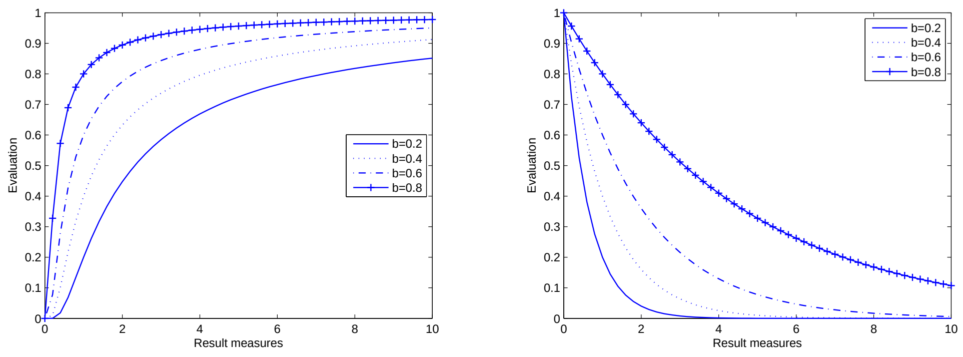
FIGURE 5.2: Increasing and decreasing evaluation functions with $b = 0.5$.

requester's perspective), all evaluations must be on the same scale to allow their combination in a single, consistent evaluation for the service, since all attribute evaluations must have the same impact on service evaluation.

To guarantee that evaluations for attributes a_i, with $i = \{1, .., n\}$, are on the same scale, we define a basic, normalized evaluation function for service attributes. Moreover, we consider service evaluation in a context in which not only are service providers diverse in terms of the quality of the service they provide, but service evaluators are also different in terms of their evaluation *standards* (so that distinct requesters might view the same service result as having different quality). Therefore, we need to find a means to allow the evaluation function to be *tuned* according to the evaluators' standards. To do so, we use a *strictness* variable (b), which allows evaluations to be more or less strict for the same service result. Thus, the basic evaluation function for an attribute is defined as follows:

- $Aeval_{srv,a_i}(c_{a_i}) = b^{c_{a_i}}$, for decreasing utility attributes, or
- $Aeval_{srv,a_i}(c_{a_i}) = b^{\frac{1}{c_{a_i}}}$, for increasing utility attributes

where $b \in (0, 1)$ defines how strict the range of acceptable values is (and alters the shape of the curves in Figure 5.2) and c_{a_i} represents the result measure associated with attribute a_i. For decreasing utility attributes, the evaluation will be higher for smaller values of their results measures (c_{a_i}), as shown in Figure 5.2(b). For example, if the performance attribute of an information retrieval service is measured in terms of the service response time, so that a service with response time of 0.5 minutes has a higher evaluation than another service with response time of 2 minutes, this attribute must have an evaluation function with decreasing utility. For increasing utility attributes, the evaluation is higher for higher values of their result measures, as shown in Figure 5.2(a). For example, considering the same information retrieval service, if its accuracy attribute is measured in terms of the number of relevant pieces of information returned by the



(a) Increasing evaluation (b) Decreasing evaluation

Figure 5.3: Impact of constant b on increasing and decreasing evaluation functions, with values $b = 0.2$, $b = 0.4$, $b = 0.6$, and $b = 0.8$.

service, so that a service that returned 10 relevant pieces of information has a higher evaluation than another service that returned 5 relevant pieces of information, this attribute must have an evaluation function with increasing utility.

To better understand the evaluation functions and how to define evaluation functions with different strictness, we analyse below the impact of the constant b on the evaluation result. We also discuss the difference on service evaluation from provider and requester perspectives, and show how to combine the evaluation of service attributes into an overall evaluation for the service, which can then be used for selection of alternative services.

5.3.3.1 Defining Evaluations with Different Strictness

The evaluation function has two parameters: b, the base of the exponential function, which defines the slope of the evaluation curve; and c, the exponent, which is the result measure of the attribute being evaluated. Constant b can be viewed as the strictness of the evaluation, since it defines the range of acceptable values for result measures of the attribute. The impact of b on service evaluations is therefore as follows.

- For attributes with increasing evaluation, higher values of b represent a strict evaluation while low values of b represent a less strict evaluation (as illustrated in Figure 5.3(a)). For example, assume that the *cost* attribute of a service is evaluated in terms of the processing time consumed by the provider, so that service cost increases with processing time. In a strict evaluation, the cost of service is high for a provider even for relatively small values of processing time, while if the evaluation is less strict, the cost of the service is low for relatively small values of processing time.

- For attributes with decreasing evaluation, higher values of b represent a less strict evaluation, while low values of b represent a more strict evaluation (as illustrated in Figure 5.3(b)). For example, assume that the *accuracy* attribute of a search engine is evaluated in terms of the number of matches returned to the user that are unrelated to the input query, so that service accuracy decreases with the number of unrelated matches. In a strict evaluation, the accuracy of the search engine is low even for relatively small numbers of unrelated matches, while for less strict evaluations service accuracy is high for small numbers of unrelated matches.

To illustrate the impact of different strictness on evaluation, Figure 5.3(b) represents four evaluation functions for the performance attribute of a search engine service. All functions use service response time (in minutes) as the result measure to evaluate performance, so that service evaluations decrease as the service response time increases. The only difference between these evaluation functions is that they have different values for constant b (which are $b = 0.2$, $b = 0.4$, $b = 0.6$, and $b = 0.8$). Now, for comparison purposes, assume that high service performance has an evaluation greater than 0.5, and a low service performance has an evaluation smaller than 0.5.

If the evaluation is strict (such as when $b = 0.2$ in Figure 5.3(b)), services are viewed as having high performance only when their response times are very small (around 0.5 minutes). If the evaluation is not strict (such as when $b = 0.8$ in Figure 5.3(b)), services are considered to have high performance even when their response times is almost six times longer (that is, around 3 minutes).

In summary, the impact of constant b on service evaluation in relation to a particular attribute depends on the type of evaluation of that attribute, so that higher and lower values of b have opposite effects for attributes with increasing or decreasing evaluation. As we will see in Chapter 6, when agents cooperate by providing services to and receiving services from each other, the equilibrium of such cooperations, in terms of the quality of the services provided and received, can be affected by differences in the strictness of the evaluation of interacting agents.

5.3.3.2 Evaluation from Provider and Requester Perspectives

Service evaluation has a different perspective for providers and requesters, since the goal of evaluation is distinct: providers use evaluation to measure their effort in performing the service, while requesters use evaluation to measure their satisfaction with service results.

Taking an example in the bioinformatics domain, consider an agent α that provides a search service on a human genome database (srv_1), and needs to request a search service on a mouse genome database (srv_2) from others, since it is not capable of performing it. As a provider, α evaluates srv_1 in terms of its cost, which is associated with effort, such that $A^{prv}_{srv_1} = \{cost\}$. As a requester, α evaluates srv_2 in terms of srv_2's accuracy, which is associated with satisfaction with service results, such that $A^{req}_{srv_2} = \{accuracy\}$. Now, assume that the result measure for srv_2's accuracy attribute is the number of unrelated matches (um) returned by the service, and for srv_1's cost attribute the result measure is srv_1's response time (rt).

In terms of effort, the evaluation of srv_1's cost is higher for higher values of response time (more response time, more effort), which is represented by the following evaluation function:

$$Aeval_{srv_1,cost}(rt) = b^{\frac{1}{rt}}$$

In contrast, in terms of satisfaction, the evaluation of srv_2's accuracy is higher for smaller numbers of unrelated matches (fewer unrelated matches, higher service accuracy), which is represented by the following evaluation function:

$$Aeval_{srv_2,accuracy}(um) = b^{um}$$

Therefore, even though we use the same evaluation scheme to define evaluation functions for both provider and requester, their evaluation functions represent an evaluation from different perspectives.

5.3.3.3 Using Evaluation Results for Future Selection

When agents need to select alternative services (to request or provide), they can use evaluations of services received or provided in previous interactions to find those with better characteristics (such as those with best performance and accuracy, or requiring smaller cost).

Although all attributes are evaluated over the same scale, so that they have the same impact on service evaluation, prior attribute evaluations can be combined during selection to form the overall service evaluation. For such combination, attributes may be seen as having more or less importance for the service overall evaluation, according to the evaluator's preference. This is achieved by assigning weights to each attribute evaluation in the overall service evaluation. Thus, given the evaluations for service attributes in all previous interactions, the overall evaluation for a service srv ($Seval(srv)$) can be calculated as:

$$Seval(srv) = \sum_{i=1}^{m}(w_{srv,a_i} \times \frac{1}{l}\sum_{j=1}^{l} Aeval_{srv,a_i}(c_{a_ij})) \qquad (5.1)$$

where m is the total number of evaluation attributes of service srv (for example, performance, accuracy, etc), l is the total number of prior interactions in which the service

Algorithm 1 Evaluation process for a received service *srv*.

```
1: input: $srv$
2: for all $a_i \in A^{req}_{srv}$ do
3:     $c_{a_i} = f_{a_i}(result_{srv})$
4:     $Aeval_{srv,a_i}(c_{a_i}) = choice(b^{c_{a_i}}, b^{\frac{1}{c_{a_i}}})$
5: end for
6: output: $\{Aeval_{srv,a_1}(c_{a_1}), ..., Aeval_{srv,a_n}(c_{a_n})\}$
```

was evaluated, $Aeval_{srv,a_i}(c_{a_i j})$ is the evaluation for attribute a_i of service srv in a previous interaction j, and w_{srv,a_i} is the weight given to a_i representing its relevance to the overall evaluation of service srv. Each agent may have its own set of weights for service attributes, and the only restriction is that $\sum_{i=1..n} w_{srv,a_i} = 1$.

Note that overall evaluations are determined during this selection from among alternative services, and not at each point in the evaluation of previous services. This is because the overall evaluation uses the weights that represent each agent's preferences over service attributes and, if preferences change over time, the comparison of two overall evaluations calculated over different sets of weights will be biased. Instead, results from the individual prior evaluations are concerned only with attributes, which are independent of agent preferences, and which can be combined subsequently in a single overall evaluation when an agent needs to select from among different services.

5.4 Evaluation process

To allow agents to evaluate services, it is necessary to identify, for each needed or provided service, a set of attributes that the evaluator considers relevant to evaluate. Then, for each identified evaluation attribute, the result measures that best define that attribute need to be determined and applied to the evaluation function for that attribute. For dynamic result measures, repeated evaluations must be performed instead of single evaluations (generated when the service is first used), since repeated evaluations can capture service behaviour under different conditions, such as different input configurations.

Once attributes, result measures and functions are defined, every time an agent receives or provides a service, it initiates an evaluation process for that service, which is carried out by the evaluation method. The input for the evaluation method is the service result itself, or information about the service execution process, and the output is a set of evaluations for service attributes.

The evaluation process for an agent α that *receives* a service srv with attributes in the set A^{req}_{srv} follows the steps in Algorithm 1. It identifies, for each attribute a_i of the service, the result measure c_{a_i}, which is determined from the service result (we use a generic function $f_{a_i}(x)$ to represent the process of determining c_{a_i}), and the evaluation of a_i through the function $Aeval_{srv,a_i}(c_{a_i})$ (which can have increasing or decreasing utility, depending on the attribute, as represented by the function $choice(b^{c_{a_i}}, b^{\frac{1}{c_{a_i}}})$). The evaluation process when α *provides* a service srv_1 follows the same steps as above, but using the set $A^{prv}_{srv_1}$ instead of set A^{req}_{srv}.

5.5 Evaluating Bioinformatics Services

As described earlier in Chapter 3, the bioinformatics domain has a variety of dynamic services with similar functionalities but that can yield distinct results for the same input data. Given the dynamic and heterogenous character of bioinformatics services, they are used as a case study for the evaluation method described in this chapter. Thus, in this section we apply the proposed evaluation method to the bioinformatics domain. In particular, we evaluate services, MS/MS search engines, that are used in *proteomics* research (Tyers and Mann, 2003) to identify unknown proteins. These services manifest characteristics of dynamism and heterogeneity, and are thus generally representative of services in the bioinformatics domain. We also show empirical results that support the need for repeated evaluation in the case of dynamic services.

5.5.1 Identifying Evaluation Attributes and Result Measures

To evaluate bioinformatics services, we must identify the evaluation attributes to be considered and develop evaluation measures for each of these attributes. To identify the evaluation attributes for MS/MS search services, we need to consider the purpose of these services as requesters and the execution cost of these services as providers.

From a requester's perspective, biological search engine services must be capable of identifying all information that is related to the biological data being analysed, so that they can provide information about its function and structure. It is important that matches returned by the services are associated with the correct degree of confidence, so that those matches with a high degree of confidence can be as trusted as the input data. Such data-related aspects are also found in traditional web search engine evaluation metrics (Dhyani *et al*, 2002), with the difference that in traditional web search the meaning of the query data is usually known by the individual submitting the query (so that identifying relevant matches is straightforward), which does not always happen for MS/MS search users. In addition, services in bioinformatics usually handle great amounts of data, and thus service results can take hours or even days to complete. In this case, performance-related aspects must also be evaluated. Although evaluation criteria regarding performance are generally applied in the evaluation of web services (Lee *et al*, 2003), they are not considered in many benchmarks for bioinformatics services (Chamrad *et al*, 2004; Kapp *et al*, 2005), which are essentially concerned with data-driven aspects.

From a provider's perspective, protein identification services need a large amount of computation resources since they usually deal with large amounts of data, so it is important to evaluate the computational cost involved in performing those services.

Based on this analysis, we have identified relevant evaluation attributes for protein identification (MS/MS search) services that consider both data-related and performance-related aspects, as described below.

- *Sensitivity* refers to the ability to identify all significant information that is related to the input data, independent of its quality.
- *Accuracy* indicates whether result errors were generated from service execution. An algorithm for comparing the similarity of a protein sequences with a sequence database, for example, must be sufficiently accurate to return only those matches that are related to the input and to avoid random matches.
- *Performance* indicates the time needed to complete a task.
- *Cost* indicates the effort needed to complete a task.

Each evaluation attribute must be measured according to concrete values that can be observed from parsed service results or from the execution process. In addition, since search engines match each individual entry in an input file (spectra) with the database, the size of the input file (number of entries) may have an influence over some evaluation attributes such as performance and sensitivity (that is, the larger the input file, the larger the expected response time for the service, and the higher the expected number of matching proteins and peptide ratio). Therefore, the evaluation of some attributes of the MS/MS search engine may consider the size of the input data in addition to the respective result measures. Note that although the attributes above could be used to evaluate services outside the scope of bioinformatics, identifying the result measures for these attributes depends on the service functionality and purpose, thus varying across services and domains. The result measures that we have identified to evaluate each attribute and the evaluation functions that we developed for each of them are described next, in relation to the bioinformatics domain.

5.5.2 Sensitivity

The sensitivity of a MS/MS search engine is related to its ability to identify all matches related to the input spectra. All MS/MS search engines have a significance value, representing the chances of that match being random, associated to each protein match returned with the search result. If the significance value is above a certain threshold, the protein match is considered a *true positive.* Therefore, we developed two result measures for the sensitivity of a MS/MS search engine: the *number of proteins* identified above the significance threshold, and the number of peptides matching those proteins (the *peptide ratio*). The higher the number of significant protein matches and the number of peptides per identified protein, the greater the sensitivity of the search engine.

Both result measures can be directly identified by parsing the search result. We determine the number of proteins by counting all protein matches returned in the result that are sufficiently significant, and the number of peptides per protein by the simple average of peptide concentrations per matching protein, as indicated below:

$$peptide\_ratio = \frac{1}{n} \times \sum_{i=1}^{n} pp(i)$$

where n is the total number of proteins, and $pp(i)$ is the number of peptides matching protein i. We then combine the number of proteins and peptide ratio into a single result measure for sensitivity (sm), which also includes the size of the input file used for search (in Kbytes). We determine the sensitivity result measure as follows:

$$sm = \frac{protein\_number \times peptide\_ratio}{input\_size}$$

Given the result measure for the attribute, we can instantiate its evaluation function. To evaluate sensitivity, it must be noted that a desirable service is one with a result with a higher number of significant protein matches and a higher concentration of peptides per protein, indicating that there was a large coverage of the protein sequence. Therefore, to reflect this result, the evaluation function for sensitivity must have a utility that increases with the ratio (sm) between returned matches and input size. We determine the evaluation function for the sensitivity attribute in the following form:

$$Aeval_{srv,sensitivity}(sm) = b^{\frac{1}{sm}}$$

where b can be any value in the range of $(0, 1)$. Here, the sensitivity evaluation increases as the number of identified proteins and associated peptides increase.

5.5.3 Accuracy

For general Internet search engines like Google or Altavista, one of the approaches to measuring accuracy is to observe the number of occurrences of the query in the matching web site (Dhyani *et al*, 2002). However, the fundamental difference between MS/MS search engines and general web search engines is that users of the latter submit keywords or expressions with the goal of finding related information, while MS/MS search engine users usually submit data with an *unknown* identity with the goal of finding similar data that could provide information about the identity or origin of the submitted data.

The submission of unidentified data to the search engines makes it difficult for the evaluator to determine the accuracy of the matched results. Unless users submit data that is already known, they cannot determine whether the match is a *false positive* (which is a match with a high degree of confidence, but not actually related to the input data) without conducting a further investigation or relying on a detailed human expert analysis. However, some approaches have been proposed by Chamrad *et al* (2004) and Kapp *et al* (2005) to provide some kind of result validation, as described below.

- *Search with multiple input files*: the technique is to send multiple input files from the same protein and analyse each result individually to identify those matches that are common to all input files and those that are not. The idea is that searching with multiple input files from the same original protein sample reduces the chance of false positives being repeated in all individual results, and thus these false positives may be spotted in the results.
- *Corroborate results of different services*: the technique is to repeat the request sent to one service with a different service and compare the matches given by both. The premise is that services give top hits similar enough to validate each others' results and, thus, the comparison of different results can identify both true positives and false-positives that are among the top hits.
- *Compare matched protein masses with expected mass*: before starting a MS/MS search, requesters may have an expected value for the mass of the protein that is present in the input file. Thus, the result can be validated by comparing the mass of top match proteins with the expected mass. True matches must have similar mass to the expected protein.

Apart from the fact that the corroboration approach depends on another service, any of these approaches could be adopted to determine service accuracy, since all provide concrete, observable result measures to be used by the proposed evaluation method. For example, if we take the *number of false positives* as a result measure for service accuracy, all three validation approaches provide different ways of getting the information, and the choice of which to use is left to specialist users.

Thus, we take the evaluation of the accuracy attribute to be a function defined in terms of number of false positives identified in the results. A desirable service will return a result with as small a number of false positives as possible. The evaluation function that reflects this has a decreasing utility in relation to the number of false positives, and is given as:

$$Aeval_{srv,accuracy}(fp) = b^{fp}$$

where fp is the number of false positives, and b can be any value in the range of $(0, 1)$. Here, service accuracy has a higher evaluation for smaller numbers of false positives.

5.5.4 Performance

For the performance of a MS/MS search engine we identify the result measure *response time*, which represents the amount of time a requester has to wait for the service result. Although for local services, response time will be similar to processing time, it may significantly differ from processing time when evaluating remote services, because response time considers the influence of network traffic. From the point of view of a requester, it is more relevant to consider the performance of the MS/MS service in terms of response time.

As with the other result measures, the response time can be determined during the execution process. Like the sensitivity result measure, the size of the input file has an influence on the service's response time, so it is again considered when determining the performance result measure (pm), as follows:

$$pm = \frac{response\_time}{input\_size}$$

where the response time is given in seconds, and the input size is given in Kbytes.

When evaluating performance, a desirable result has small values for response time, and thus the evaluation function for performance must have a decreasing utility in relation to pm. The evaluation of service performance is given by the function:

$$Aeval_{srv,performance}(pm) = b^{pm}$$

where b can be any value in the range of $(0, 1)$. According to the function, service performance is higher for smaller values of pm and, consequently, for shorter response times.

5.5.5 Cost

For the cost of providing a MS/MS search service, we identify the result measure *processing time*. Just like the sensitivity and performance attributes, the size of the input file has an influence on the required amount of processing time, so it is also considered in the cost result measure (cm) as follows:

$$cm = \frac{processing\_time}{input\_size}$$

where the processing time is given in seconds, and the input size is given in Kbytes.

Evaluating the service cost aims at determining the effort of the provider in executing the service. Therefore, the cost of the service must be higher for higher processing times, which is represented by the following evaluation function:

$$Aeval_{srv,cost}(cm) = b^{\frac{1}{cm}}$$

where b can be any value in the range of $(0, 1)$. According to the function, service cost increases with the value of cm and, consequently, the value of $processing\_time$. Note that although the service performance and cost are measured with similar result measures, response time and processing time, the evaluation functions for both attributes have the opposite behaviour, since evaluation is seen from different perspectives by the requester and the provider. This means that small processing and response times represent, respectively, high performance for the service requester but small effort for the service provider.

After having determined the evaluation functions for all evaluation attributes of MS/MS search engine services, we can apply these functions to evaluate real bioinformatics services. The results are presented below.

5.5.6 Evaluation Results

In the previous sections we have developed evaluation attributes, measures and functions for evaluating bioinformatics MS/MS search services, on the basis of our proposed evaluation method. Here, we apply these evaluation functions to evaluate real MS/MS search services.

For this empirical study (also presented in (Rodrigues and Luck, 2006c)), we use real protein search engines that are publicly available, two of which have a local version and two of which are remotely accessed. These search engines are Mascot Remote (Perkins *et al*, 1999), Tandem Local and Remote (Craig and Beavis, 2003), and OMSSA Local (Geer *et al*, 2004).

To investigate if the evaluation method can be used to compare different services under similar conditions, the requests were submitted to all services using the same input spectra (580.8Kb) and the same input configuration. Also, to compare evaluations of the same service under different conditions, we used two different input configurations,

Parameter	C_1	C_2
Database	NCBInr	NCBInr
Enzyme	Trypsin	Trypsin
Taxonomy	Mammals	All entries
Fixed Modifications	Carbamidomethyl (C)	None
Potential Modifications	None	None
Peptide Tolerance	2.0Da	2.0Da
Fragment Tolerance	0.8Da	0.8Da
Missed Cleavages	1	1

TABLE 5.1: Initial configurations for MS/MS search services.

Service	protein_number		peptide_ratio		Sensitivity	
	C_1	C_2	C_1	C_2	C_1	C_2
Mascot Remote	**13**	40	**10**	9	**0.045**	0.327
Tandem Local	8	**36**	11	**51**	0.010	**0.803**
Tandem Remote	2	3	9	6	1.9E-10	1.9E-10
OMSSA Local	31	31	4	4	0.039	0.039

TABLE 5.2: Evaluating MS/MS search services according to the sensitivity attribute.

as shown in Table 5.1. Configurations C_1 and C_2 have different settings for parameters *Taxonomy* and *Fixed Modifications*, while the other parameters are kept the same. We repeated the evaluation process for each input configuration using the same spectra to observe the changes in evaluation results. The evaluation functions for all attributes used a strictness of $b = 0.5$.

Results for the evaluation of the four different MS/MS search services according to the *sensitivity* attribute are shown in Table 5.2, and the services with best evaluation for this attribute using each configuration are highlighted in bold. Here we observe that, for configuration C_1, *Mascot Remote* has a better sensitivity, but for configuration C_2, *Tandem Local* is better. Also, we observe that all services had sensitivity for configuration $C2$ higher than or equal to the sensitivity for $C1$ (as expected since configuration C_2 is more general, allowing more proteins to match the input, while configuration C_1 is more specific, restricting the search space of the MS/MS search engine).

Evaluation results for the *performance* attribute are shown in Table 5.3. Here, we observe that *Tandem Local* has better performance when using configuration C_1, but *Tandem Remote* is better when configuration C_2 is used. All services except for OMSSA Local had better performance when using configuration C_1 than when using configuration C_2 (as expected since the space to search is larger for configuration C_2, which is more general, while the search space is narrowed by configuration C_1, which is more specific).

From a requester's perspective, we observe from the above results that, if the evaluation process had not been repeated after the first interaction with *Tandem Local*, a requester interested in finding the best service in terms of sensitivity would have the wrong information that *Mascot Remote* would perform better when using configuration C_2 as

Service	response_time (sec)		Performance	
	C_1	C_2	C_1	C_2
Mascot Remote	172	200	0.814	0.787
Tandem Local	**11**	41	**0.986**	0.952
Tandem Remote	26	**37**	0.969	**0.956**
OMSSA Local	2581	2489	0.045	0.051

TABLE 5.3: Evaluating MS/MS search services according to the performance attribute.

Service	processing_time (sec)		Cost	
	C_1	C_2	C_1	C_2
Mascot Remote	-	-	-	-
Tandem Local	**11**	41	**1.4E-16**	**5.4E-5**
Tandem Remote	-	-	-	-
OMSSA Local	2581	2489	0.855	0.8507

TABLE 5.4: Evaluating MS/MS search services according to the cost attribute.

well. Similarly, if the evaluation process had not been repeated after an interaction with Tandem Remote, a requester interested in finding the best service in terms of performance would have the incorrect information that Tandem Local would present better performance also for configuration C_2.

From a provider's perspective, service evaluation in terms of cost shows that, from the two local services, Tandem has the smaller cost, since its processing time is much less than OMSSA's processing time, as shown in Table 5.4. Also, for service Tandem Local, the search process is less costly for configuration C_1 than for configuration C_2, and the opposite happens for service OMSSA Local.

In the future, to select from these alternative services, the requester can also determine the overall evaluation for each service by applying Equation 5.1. Evaluations for MS/MS search services using a particular configuration are shown in Table 5.5, in which evaluations in the first column use the same weight for the sensitivity (w_{sens}) and performance (w_{perf}) attributes, and evaluations in the second column use different weights for these attributes (with $w_{sens} = 0.7$ and $w_{perf} = 0.3$). In this case, the service Tandem Local has the highest service evaluation for both configurations using both sets of weights for service attributes.

The results show that dynamic services, like those presented in this empirical study, despite having similar functionally, can yield heterogeneous results under different configurations. Thus, there is a need to dynamically evaluate services to improve the efficiency of future selection of alternative services.

Service	Seval ($w_{sens} = w_{perf}$)		Seval ($w_{sens} \neq w_{perf}$)	
	C_1	C_2	C_1	C_2
Mascot Remote	0.429	0.557	0.275	0.465
Tandem Local	**0.498**	**0.879**	**0.302**	**0.847**
Tandem Remote	0.484	0.478	0.290	0.286
OMSSA Local	0.042	0.045	0.040	0.042

Table 5.5: Evaluation of MS/MS search services.

5.6 Comparison with Similar Methods

Similar methods to evaluate dynamic services include (Casati *et al*, 2004), (Day and Deters, 2004), and (Sun *et al*, 2006). In (Casati *et al*, 2004) and (Day and Deters, 2004), service results are stored after services are used and service evaluation is performed dynamically when the user wants to select a provider. This dynamic evaluation is generated by classifiers that take the information on prior service results and classify the service in one of a pre-defined set of quality categories (such as *poor*, *acceptable*, *good*, or *excellent*), which represent the (qualitative) service evaluation. The difference between both methods is in the implementation of the classifier (with Casati *et al* (2004) using decision trees and Day and Deters (2004) using a Naive Bayes reasoner).

Although both approaches are suitable for dynamic domains, since evaluations are computed dynamically to capture variations in service results, the use of pre-defined quality categories assumes that service users know what quality levels to expect from service results, so that a qualitative evaluation can be generated for each service based on past results. However, this is not possible for all services, as for the protein identification services presented in this chapter, since users of such services often use input data with unknown identity, and service results are influenced by different input configurations and the quality of the input data.

In (Sun *et al*, 2006), service evaluation is given by an aggregation of performance-related measures. Although this method generates absolute evaluations for services that can be consistently compared in future service selection, it defines evaluation measures that are restricted for evaluating service performance, so that its application to evaluate different service attributes is not straightforward.

5.7 Conclusion

This chapter has presented a general evaluation method for dynamic services to be used by agents in open systems. We discussed the issues for efficient evaluation of dynamic services, which include the adoption of a repeated evaluation process, the use of absolute evaluations, and the generation of comparable evaluations, and described the components of the evaluation method.

The evaluation method described here identifies dynamic properties of diverse services without requiring any information on expected service results, and generates consistent evaluations which can be compared at different points in time. In particular, our method provides a means for agents to dynamically evaluate services and to use these dynamic evaluations of service properties to select among alternative services in the future. Moreover, it offers the flexibility required for the diverse evaluators that can be found in open systems, in the sense that it allows for personalised definition of evaluation functions (so that agents can be designed to be more or less strict in their evaluations), and it allows the evaluation of services from the perspective of both service providers and service requesters.

We applied the evaluation method to evaluate services for protein identification, and showed the importance of a dynamic evaluation process for such services through empirical results. Results showed that there is a need to dynamically evaluate services to provide more accurate information about their results, so that agents requesting services in dynamic environments can improve the selection of alternative services in the future.

Regarding the domain of proteomics, although benchmarks for MS/MS services have been presented in the literature (Chamrad *et al*, 2004; Kapp *et al*, 2005), there are many limitations in automating real protein identification experiments, including the lack of consideration for performance-related metrics of these services, the evaluation process based on analysis of annotated biological data, and a static evaluation process in which services are evaluated only once. We have addressed these limitations by defining novel evaluation functions and metrics, for evaluating MS/MS services dynamically, and which are directed at unknown biological data and consider data and performance-driven aspects of services. Therefore, we argue that our evaluation method applied to MS/MS services contributes towards the efficient automation of *in-silico* protein identification experiments, so that such experiments can improve in quality once the properties of alternative services are known (and the best ones can be identified and selected).

Although the case study for our evaluation method targets services in the proteomics domain, it can also be applied to services in other domains that share the same characteristics of dynamism (where different evaluation criteria may be required, but using the same method).

Chapter 6

A Computational Framework for Non-Monetary Cooperation

6.1 Introduction

Agents operating in open cooperative systems with free services must be able to make decisions over the interactions in which they engage. More specifically, agents must decide among alternative interaction partners both when requesting and providing services from and to others. We have already argued in our problem scenario in Chapter 3 that the foundations of such decisions are, first, the evaluations that agents give to services, and second, the non-monetary incentives to start or maintain a cooperation. Since we have addressed service evaluation in the previous chapter, we can now focus on the incentives for cooperation in such systems by developing a framework for non-monetary cooperative interactions.

According to Burt (2000), it can be useful in certain contexts to view a society as a market in which individuals exchange goods, services and ideas to achieve their goals. While markets typically involve monetary exchanges, many do not necessarily involve economic capital. For example, in computer-supported scientific communities like bioinformatics, different types of information and tools can be exchanged in a non-monetary and cooperative way in order to improve individual or global results (Stein, 2002). In particular, in the case when service provision is free of charge, it becomes important to enforce the link between providers and requesters, so that agents can count on *stable* interactions if they need the help of others to achieve their goals.

To strengthen the relation between requesters and providers in non-monetary exchanges it is therefore important to explicitly motivate cooperation, otherwise there is no incentive for autonomous agents to provide services to each other. In a system with self-interested entities, a service provider has an incentive to cooperate if it receives some benefit in return from the requester, either immediately or in the near future. In the case of immediate reciprocation, it is easier to model and to check whether the cooperative interaction is genuine (mutual), since it involves concrete actions that can be clearly observed by both entities. However, immediate reciprocation is not always possible since the provider may not need any service at the time the interaction takes place, or the requester may only be available to provide a service in return in the near future.

Cooperative situations in which reciprocation is not immediate thus raise interesting issues for modeling cooperative agents. First, there is no guarantee that the requesting agent will reciprocate in the future. Second, the provider must receive some *benefit* in return from the requester, so that the provider is motivated to cooperate even in the lack of a concrete, immediate return. Third, if we consider that in a multi-agent system agents might have different perspectives of the same service due, for example, to individual preferences and the relevance of each service to their individual goals, it may be that an agent receives less than expected from a cooperation; that is, a provider can evaluate a service it receives in return in the future as being of lower quality than the service it provided in the past.

Models for non-monetary exchanges in human societies have been proposed in social theories like those of Homans (1962, 1974) and Piaget (1973). Both theories consider interactions between people as exchanges of goods, either material (like objects or services) or nonmaterial (like ideas, advice, or gestures). As such, interactions involve an exchange of *values* between participants, like informal credits and debts, gratitude, and satisfaction, among others. However, while Homans presents a more explanatory and investigative theory of social exchange (with practical experiments and observations), Piaget considers concrete definitions and algebras to support his theory of exchange values. The existence of a richer logical background in Piaget's theory thus makes it less susceptible to subjective interpretations, since the specific values present in every exchange, their nature, and their inter-relation are explicitly identified, and are thus more suitable for developing a computational model of non-monetary exchanges.

Piaget's view of social exchanges is that individuals build up informal credits and debts as a result of their interactions, and these values are seen as *informal commitments* between individuals in the sense that they are not supported by any legal norm or institution. Such informal commitments motivate both interactions between individuals and the maintenance of these interactions over time. Importantly, the credits and debts in Piaget's model are not concerned with economic values, but with what individuals owe to each other as a result of their interactions.

In this context, we propose a model for non-monetary interactions between autonomous agents in which the explicit motivation for cooperation follows Piaget's *exchange values* approach. The aim is that agents structure their interactions based on the informal commitments that exchange values imply: they use reciprocity to ensure compliance with requests, and use the credits and debts that are gained after each interaction to analyse incoming requests and evaluate possibilities of future interactions. Here, commitments are not formal structures of mental representations of an obligation (as discussed in (Jennings, 1996)), but are used to represent an informal commitment of one individual to return a favour to another, to reciprocate in response to a service received in the past.

The key contribution here is a computational model based on *exchange values* (Piaget, 1973) to deal with the issues of modeling cooperative agents that act in (open) non-monetary applications. Exchange values represent the agents' individual subjective evaluation of provided and received services, and are associated with their interactions, indicating the effort, cost, or satisfaction of each agent.

This chapter describes our proposed framework for non-monetary cooperative interactions, and is structured as follows. An overview of the theory of exchange values and a discussion of its limitations to developing a computational model are presented in Section 6.2. Then, the benefits of using exchange values to model non-monetary cooperative interactions and our computational framework for using exchange values are described in Section 6.3. Within this framework, the computational model for exchange values is presented in Section 6.4, including the core model and the description of how individual exchange values are derived. We then show how agents determine the balance of their exchange values in Section 6.5, and summarise the steps of an interaction between two agents in which exchange values are used in Section 6.6. A practical example of how to apply the proposed computational model of exchange values to interactions between agents in a cooperative bioinformatics application is presented in Section 6.7. The chapter finishes with conclusions in Section 6.8.

6.2 Piaget's Theory of Exchange Values

Social exchange is the particular interaction in which an individual performs an action on behalf of another and receives an action in reciprocation either immediately or in the near future. The *theory of exchange values* was proposed by Piaget (1973) as an analysis of human social exchanges and the reasons for their persistence or discontinuity. More specifically, Piaget argues that in all social interactions in which one individual acts on behalf of another there is an *exchange of values* between them. These values result from each individual's assessment of the provided or received action over a common scale of values.

According to Piaget, every action and reaction of two interacting individuals towards each other has an influence on their values: it can increase or decrease their values. In this way, when an individual (the provider) acts for the benefit of another (the receiver), causing the latter to increase its values, the receiver can react in one of the following ways.

1. The receiver can pay back the provider by giving an object or providing another service in return. This is the case, for example, if a researcher who intends to submit a paper to a conference receives comments on his paper from a colleague who is also working on a conference paper, and returns comments on his colleague's paper to reciprocate the favour.

2. The receiver just *valorises* the provider by expressing gratitude or approval, instead of giving something immediately in return. This is the case, for example, when a researcher, who receives comments from a colleague on a paper that he intends to submit to a conference, expresses gratitude and *valorises* his colleague's action to show intentions of future reciprocation.

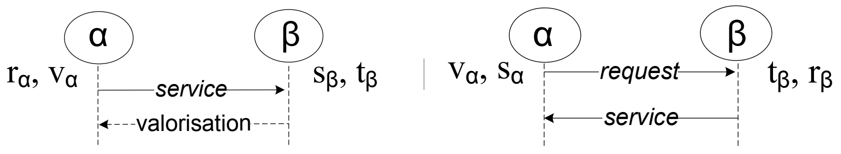
3. The receiver neither returns a service to, nor valorises, the provider.

All the above reactions of the receiver also have an effect on the provider's values. In the first situation, the receiver performs a material action[1] for the provider (comments on a paper), which constitutes an *actual* value for the latter (the comments are valuable to improve the quality of the paper). In the second situation, the receiver performs an abstract action for the provider (an utterance or gesture of approval, gratitude, etc), which constitutes a *virtual* value for the latter, in the sense that his valorisation gives him reputation, respect and authority, which are values he can use to gain some benefit in future interactions (for example, the next time he is writing a paper, he can ask the receiver for comments). The third reaction, however, is disadvantageous for the provider since the receiver does not reciprocate the action in any way, and the receiver is ultimately devalorised by the provider as ungrateful or unjust.

The values that are exchanged between individuals are clear when concrete objects are involved in the exchange, as in the first situation above. However, when the exchange involves virtual values, as in the second situation, a more detailed analysis is needed. Here, the performed action is referred to as an *actual renouncement* for the provider, since it requires the expenditure of time and effort, and it is an *actual satisfaction* or gain for the receiver. The valorisation of the provider by the receiver, as a reaction to the received action, is for the provider a reward, a *virtual credit* that he can draw upon in the future, and for the receiver the valorisation constitutes a promise, a *virtual debt*, in the sense that the receiver feels obliged to return the favour to the provider in the future.

According to Piaget, exchanges between individuals occur in two stages: in the first stage, which we call the *provision* stage, interacting individuals acquire virtual values

[1]We will use action in a general sense also to represent providing an object or a service.



(a) Provision Stage (b) Reciprocation Stage

FIGURE 6.1: Exchange values in the provision and reciprocation stages of the interactions between individuals α and β.

which represent the valorisation of the provider by the receiver; in the second stage, which we call the *reciprocation* stage, these virtual values are realised as actions, representing a reciprocation. The provision and reciprocation stages are described in the next sections.

6.2.1 First stage: Provision

When two individuals α and β interact, and α performs a service on behalf of β then, according to (Piaget, 1973), exchange values are associated with the interaction as follows.

1. α performs a service and associates with its action a *renouncement* value r_α indicating the time or resources invested in that action.
2. β receives a service from α and associates with it a *satisfaction* value s_β.
3. β valorises α, and this constitutes a *debt* value t_β with α in response to the satisfaction it gained.
4. The valorisation of α by β is taken by α as a *credit* value v_α, which can be drawn upon in the future.

The exchange values in the provision stage are shown in Figure 6.1(a), in which the valorisation is represented with a dashed arrow to indicate a *virtual* action, in contrast to the service, which is a *material* action. By the end of the interaction, both agents *acquire* virtual values — a debt for β and a credit for α.

In the future, α can make use of this credit, v_α, and ask β to perform a service on its behalf. Nothing forces β to accept the request, so if β feels gratitude towards α or wants to keep interacting with it, β accepts its debt and returns the favour to α. Otherwise, if β does not admit its debt, α devalorises β as ungrateful. If β agrees to return the favour to α, they enter the reciprocation stage of the interaction, which is described next.

6.2.2 Second Stage: Reciprocation

When α makes use of its credit and interacts with β, and the latter performs a service on the former's behalf then, according to (Piaget, 1973), exchange values are associated with the interaction as follows.

1. α requests a service from β with the belief that it has a virtual credit v_α from previous interactions.
2. β admits a virtual debt t_β from previous interactions, and performs a service for α accordingly.
3. β associates with its action a *renouncement* r_β indicating the time or resources invested in that action.
4. α receives the service and associates with it a *satisfaction* s_α.

By the end of the interaction, virtual values of credit and debt are realised in values of renouncement and satisfaction. The reciprocation stage is illustrated in Figure 6.1(b). When the two stages of the interaction take place, the interaction is said to be complete, since there was reciprocation between the individuals.

6.2.3 Equilibrium of Exchange Values

In every interaction in which a service is provided or received, something is lost and something is gained: a provider α loses *investments* (r_α) of time and resources for providing a service but gains a credit as a result of its *valorisation* (v_α), and a receiver β gains *satisfaction* (s_β) with the benefits of the received service but loses in acquiring a *debt* (t_β) with the provider in return. The relation between these actual and virtual values represents the counterbalance of gains and losses in each stage of the interaction, so deferred exchanges (or reciprocations) are possible.

Since exchange values result from individual evaluations of services, differences in such individual evaluations determine the balance of gains and losses in an interaction. If the gains and losses of participants according to their individual evaluations are equivalent, the interaction between them is said to be in *equilibrium* (provided that the participants' evaluations are estimated over the same scale of values).

In the provision stage, the equilibrium of exchange values is represented in the form of the equation below[2]:

$$(r_\alpha = s_\beta) \wedge (s_\beta = t_\beta) \wedge (t_\beta = v_\alpha) \Rightarrow (v_\alpha = r_\alpha) \tag{6.1}$$

[2]This is an interpretation of the variation of exchange values described by Piaget (1973)

This means that, if the evaluations that α and β give to the service they provide or receive are equivalent, the debt (t_β) acquired by β is the same as its satisfaction (s_β), and the credit (v_α) gained by α is equal to the effort (r_α) it spent.

Similarly, the equilibrium of exchange values in the reciprocation stage is represented as follows:

$$(v_\alpha = t_\beta) \wedge (t_\beta = r_\beta) \wedge (r_\beta = s_\alpha) \Rightarrow (s_\alpha = v_\alpha) \tag{6.2}$$

Here, if the evaluations that α and β give to the reciprocated service are equivalent, then β pays its debt (t_β) by performing a service with equivalent effort (r_β), and α spends its credit (v_α) by receiving a service with equivalent satisfaction (s_α).

However, individual interests and different perspectives over the levels of quality of the services that are provided and received may cause the disequilibrium of exchange values in both the distinct stages of an interaction and in the complete interaction, so that equilibrium situations may not always be achieved.

6.2.4 Limitations of Piaget's Model

Although Piaget's model provides an algebra of exchanges, which makes it mathematically richer than other sociological theories of social exchange like Homan's theory (Homans, 1962, 1974), it lacks a more precise definition of the exchange process than is necessary for a computational representation of the model. In particular, it does not specify the *origin* of virtual values, how the provider is *aware* of his valorisation, nor how virtual values are *accumulated and spent* in the provision-reciprocation cycle.

More specifically, the model does not provide the precise relation between the four exchange values involved in an exchange between two individuals. For example, it is not clear if a debt acquired by the requester is based on its own evaluation of a received service (its satisfaction value), or on the provider's evaluation (the provider's renouncement value). Similarly, it is not clear if the valorisation, which originates the provider's credit, is based on the requester's objective evaluation (the requester's satisfaction value), or on the requester's subjective assessment of its debt.

In addition, if we assume that the individuals' exchange values are interrelated (that is, if the virtual values of one individual have their origin in the values of another individual), we have to deal with a second limitation of the theory, which is related to *awareness*. More specifically, it is not clear in Piaget's model whether individuals are aware of each other's values or if they consider what they *believe* the other individual's exchange values are. In human social exchanges what happens is probably a mixture of both: an individual who receives a service from another *expresses* its satisfaction or valorisation to the provider (through gestures, words, or actions), and the latter *interprets* this expression on his own terms (that is, he keeps to himself what he believes the other individual's expressed values are). However, such a model of social exchange based on expression and interpretation is very difficult to apply to interactions between computational entities. Therefore, either agents must be aware of each other's exchange values through explicit communication of these values, or they must model what they believe the interaction partner's exchange values are, based on their own information.

Finally, at the reciprocation stage, the way credits are spent and debts are paid, so that they decrease in value, is not clear. For example, the model does not explicitly state whether credits and debts can accumulate nor, in case they do accumulate, how the amount of credit is to be spent or debt to be paid.

In what follows, we discuss how exchange values can be applied to interactions between computational entities, and describe our computational framework for using exchange values in open cooperative systems.

6.3 Computational Framework

6.3.1 Exchange Values for Non-Monetary Interactions

Ignoring the limitations above, in relation to open cooperative systems in which computational services are free of charge, such as those in the bioinformatics domain, exchange values can be used to motivate cooperation among autonomous agents. In this way, instead of relying on benevolence or monetary compensation, cooperation is achieved based on service reciprocation, which is motivated by credits and debts that agents gain and spend over interactions.

Moreover, since exchange values are based on individual evaluations of provided and received services, they enable agents to identify (through service evaluation) differences among interaction partners, such as the level of quality of services provided, and the strictness in evaluating services received. More specifically, analysing cooperations in terms of the equilibrium of exchange values allows requesters and providers to identify whether a cooperation is beneficial (when gains compensate for losses) or harmful (when they do not). Such analyses provide a means for agents to choose interaction partners, and to improve the results of their interactions.

To apply Piaget's exchange values model to interactions between autonomous agents in cooperative applications with free services, we propose a *computational framework for non-monetary interactions based on exchange values.* Such a framework needs a *computational model for exchange values* that not only determines the way these values are represented computationally, but also addresses the limitations of Piaget's model discussed in Section 6.2.4. To develop such a model, we first discuss our perspective on the origin of exchange values, present the modeling requirements, and then discuss the alternative representations for exchange values in a computational system.

6.3.2 The Origin of Exchange Values

As mentioned previously, there are two types of exchange values: actual values of renouncement and satisfaction, and virtual values of credit and debt. These *actual* values result from objective evaluations of the service or object involved in the exchange, and are thus represented as *objective values.* By contrast, *virtual* values result from subjective judgements of the objective evaluation, and are represented as *subjective values.*

Clearly, objective values are present in any kind of exchange (economic, non-economic, immediate or deferred), since an agent providing a service always incurs some cost or effort (which is an actual renouncement), and an agent receiving a service always has a benefit or gain (an actual satisfaction)[3]. Although these values have some degree of subjectivity, since two agents might have different perspectives over which aspects of a service are more relevant (like cost or the nature of a "good" result), both satisfaction and renouncement values can be determined in a straightforward way through *objective* evaluation processes, such as those considered in Chapter 5 (or simply through utility functions). This makes clear the origin of objective values: each agent determines its satisfaction or renouncement value based on its own objective evaluation process or utility function.

In contrast, subjective values are present only in non-monetary interactions, the focus of our work. To determine the origin of these values, we use the premise that subjective values (the debt and credit) result from the *valorisation* the receiver agent gives to the provider. Moreover, we consider this valorisation to be based on both the receiver's satisfaction with the service and its subjective judgement over the service or provider (so that the debt acquired by the receiver may not coincide with its satisfaction). We discuss possible subjective influences on credits and debts next.

6.3.3 Subjective Influences

In open systems in which agents with different characteristics and behaviours interact, an agent's subjective evaluation of its debt or credit is likely to be influenced by its differences with other agents and its individual behaviour. Examples of such influences are:

- *fairness*, since an agent may admit a debt that is different from its satisfaction to take advantage of the interaction;

[3]There is always a benefit or gain, except when the action that was supposed to benefit the receiver actually brings it a loss.

- *performance skills*, since differences in the skills of two interacting agents may influence their debt (as shown in a practical experiment about human social exchanges described in (Homans, 1962));
- *social hierarchy* between agents in the society, since the position of two interacting agents in this hierarchy may influence the valorisation of the provider by the receiver;
- *dependence*, in the sense that agents may depend on each other to perform services, and this dependence between interacting agents may influence their subjective evaluation;
- *service demand*, since knowledge of the availability of a service or the demand on its provider in the environment (or a community) may influence the acknowledged debt of the requesting agent, if it considers a demanded service as more valuable than another that is plentiful in the environment;
- *expected service quality*, since an agent may expect a certain quality from the service it requested, and the comparison of the actual quality with the expected quality may change the subjective value of the service from the requester's perspective; and
- *credit saturation*, which refers to the accumulation of a large amount of credit, since one agent may perceive the continual repeated gain of credits from another agent as less valuable (or useful) as they accumulate and see no means of spending them (as described in (Homans, 1974) in relation to human social interactions).

When subjective influences modify an agent's objective evaluations, they may cause the disequilibrium of exchange values (since the requester's debt may not match with its satisfaction, and the provider's credit may not match with its renouncement). Indeed, agents may use these influences to take advantage of the interaction (for example, by decreasing their debt), or to improve their chances of future interactions (for example, by increasing the valorisation given to a highly skilled provider).

6.3.4 Modeling Requirements

Apart from providing an explicit motivation for cooperation, the benefit of the exchange values model lies in the way it enables agents to reason about gains and losses from service provision and their balance, which is particularly important when agents cooperate with others providing services with different levels of quality. To realise this benefit, a computational model of exchange values needs to address the following.

- *Compensation*: two exchange values may be different in quantity but equivalent from a qualitative point of view. In economic (monetary) exchanges, services are provided and received based on a fixed, exact quantitative value. For example, in an economic exchange, the provider of a service s_1, which costs £50.55, must receive this exact amount from the service requester in return for providing the service. However, we adopt the view that, in non-monetary exchanges, individuals are not concerned with the exact value of provided and received services as long as they *broadly* compensate for each other. This is because it is difficult to achieve precise compensation in such non-monetary exchanges, since each individual may have a different perspective over the value of a particular service. Instead, each individual *approximates* the value of the service they provide and receive in non-monetary exchanges so that services that are reciprocated between individuals are not required to have identical quantitative values.

- *Variation*: exchange values increase and decrease their amount from one interaction to another, since in every interaction these values are acquired and spent by agents (that is, a credit gained in one interaction can be spent in the next interaction, and a debt acquired previously can be paid back in a future interaction).
- *Scale*: exchange values need to be represented along some scale, so that interacting agents can compare their exchange values.
- *Comparability*: exchange values must be comparable, so that an analysis of their relative balance is possible, and agents are able to determine whether their gains are smaller than, greater than, or equal to their losses in each interaction with other agents. Thus, once agents are able to determine their balance of exchange values, they may use this information to identify whether they gain or lose with an interaction.

The need for an approximate equivalence of gains and losses that allows compensation of provided and received services in non-monetary exchanges between agents suggests a qualitative approach to the representation of exchange values. However, the necessity to modify exchange values from one interaction to another, and to compare them, points to a quantitative approach to the representation of these values. We consider these alternative approaches below.

6.3.5 Alternative Representations

Following the view that, in non-monetary interactions, individuals are not concerned with precise compensation of service provision, the notion of equivalence in such interactions is more qualitative than quantitative. Qualitative values that have a certain degree of vagueness can be represented with a fuzzy approach, more precisely with fuzzy sets (Dubois and Prade., 1980). Unlike traditional set theory in which values belong to a set with a true or false relation (for example, value a belongs to set A or does not belong to set A), in fuzzy sets each value belongs to a set with a degree of certainty (for example, value a can *partially* belong to set A).

Fuzzy sets usually represent linguistic concepts such as "very high", "high", "medium", and "low", or "A", "B", "C", and "D" to which quantitative values (members) belong with a $[0, 1]$ degree of certainty (varying from 0 for non-member values to 1 for values with complete membership). For example, with the fuzzy approach, an agent may receive a credit with value "very high" with a certainty of 0.8, and in the future spend this credit by receiving another service as reciprocation which it evaluates as having a "very high" satisfaction but with a 0.6 certainty. In this way, even though the values of credit and satisfaction are not exactly the same, both have the same qualitative value for the agent ("very high"), and thus compensate for each other.

The fuzzy approach copes with the issue of approximate compensation, since different services can have the same fuzzy value, but with different degrees of membership. Regarding the issues of comparison of exchange values, since fuzzy sets are not ordered, it is necessary to define an ordering relation for all possible fuzzy sets that represent exchange values, such as "very high" > "high" > "medium" > "low". However, operations to modify exchange values are expensive, since they require the definition of an algebra relating all fuzzy sets so that exchange values can increase and decrease in value.

An alternative approach to representing exchange values qualitatively is through numerical intervals. Here, each interval comprises a range of integer or real values delimited by a lower and upper bound. To determine an exchange value using the numerical interval approach, the quantitative evaluation of a service must be *mapped* to one of the defined intervals. Since intervals comprise a range of quantitative values, it is possible to have equivalent exchange values represented by intervals for similar, but not equal, evaluations. Thus, as for fuzzy sets, the numerical intervals approach also copes with the issue of approximate compensation of services. Regarding the comparison and variation issues, operations for comparing, increasing and decreasing exchange values are less computationally costly using numerical intervals than using fuzzy sets. Nevertheless, comparing and varying intervals is more expensive computationally than using a quantitative representation.

Indeed, although in many real life situations in which non-monetary exchanges take place it is difficult to quantify the precise value of a service (such as the value or contribution of each comment to a paper review), the evaluation of computational services is generally a quantitative operation. For example, computational services in the bioinformatics domain give quantified results (e.g., number of proteins, number of peptides, proportion of sequence coverage, etc), so determining a quantitative evaluation for those services is straightforward. Therefore, an alternative to qualitative approaches is to represent exchange values quantitatively through integer or real numbers. Quantitative scales are ordered, and thus comparison operations are already defined, in contrast to qualitative approaches. In addition, increasing and decreasing quantitative values is computationally simple and does not require additional algebra definitions. However, the broad compensation of services that is characteristic of non-monetary exchanges is difficult to achieve with a quantitative equivalence of exchange values.

Considering that computational services can generally be evaluated through quantitative operations, and due to the advantages of the quantitative approach mentioned above, in our model we adopt a quantitative representation for exchange values. To address the problem of broad compensation of services, we redefine the quantitative equivalence for exchange values by defining a *tolerance* threshold when comparing two exchange values, so that if the difference between these values is below the threshold, they are considered equal. This satisfies our need, but also enables us to adjust such a threshold to suit the particular domain according to user demands.

More specifically, with this representation of exchange values, we must define a *scale* along with they are compared, so that interacting agents can compare their evaluations of the service they provide or receive on the same basis. Since exchange values are based on service evaluation, we can adopt the same scale that is used in the evaluation method proposed in Chapter 5, which is a $[0, 1]$ scale. Although such a quantitative scale does not have an explicit notion of service quality or provision effort, we assume that evaluations are made sensibly over the 0 to 1 range (that is, good quality results tend to 1 and poor quality results tend to 0, while highly demanding service executions tend to 1 and trivial service executions tend to 0). This also gives a coherent basis for comparison of exchange values from two interacting agents.

6.4 A Computational Exchange Values Model

Having specified the origin of exchange values and the requirements for a computational model of these values, here we describe a core model to represent exchange values computationally. We then show how exchange values are determined by two interacting agents, and accumulated and spent over repeated interactions. We determine exchange values separately according to their origin, so that we have: *objective values*, which are based on objective evaluations; *subjective values*, which are based on subjective evaluations; and *reciprocation values*, which are based on the credits and debts acquired by agents in previous interactions.

6.4.1 The Core Model

In each interaction between two agents in which a service is being provided or received, four exchange values are involved: the *renouncement* (r), the *satisfaction* (s), the *debt*

(t), and the *credit* (v). These exchange values are not necessarily the same in the provision and reciprocation stages of the interaction since each stage is a distinct interaction in which different services may be provided, and thus the satisfaction and renouncement values determined by agents in each stage may be different. Exchange values acquired in the provision and reciprocation stages must therefore be represented as distinct values. To distinguish exchange values in each stage we use the notation r, s, t and v for values acquired in the provision stage, and the notation r', s', t' and v' for values acquired in the reciprocation stage.

In particular, in each interaction in which an agent participates either as a provider or a requester, there is a *distinct set of exchange values*, which is different for each stage of the interaction. Thus, for interactions between two agents α and β in the provision stage, the set of exchange values of α can be represented as a tuple, $EV_{\alpha\beta} = (r_{\alpha\beta}, s_{\alpha\beta}, t_{\alpha\beta}, v_{\alpha\beta})$, and the set of exchange values of β as $EV_{\beta\alpha} = (r_{\beta\alpha}, s_{\beta\alpha}, t_{\beta\alpha}, v_{\beta\alpha})$. For interactions in the reciprocation stage, the set of exchange values of α is represented as $EV'_{\alpha\beta} = (r'_{\alpha\beta}, s'_{\alpha\beta}, t'_{\alpha\beta}, v'_{\alpha\beta})$, and the set of exchange values of β as $EV'_{\beta\alpha} = (r'_{\beta\alpha}, s'_{\beta\alpha}, t'_{\beta\alpha}, v'_{\beta\alpha})$.

Only two values in the agent's set of exchange values are determined in each interaction. For example, in the provision stage of the interaction in which α provides a service to β, α's exchange values are renouncement ($r_{\alpha\beta}$) and credit ($v_{\alpha\beta}$), and β's exchange values are satisfaction ($s_{\beta\alpha}$) and debt ($t_{\beta\alpha}$). In the reciprocation stage of the interaction in which β provides a service to α in return, α's exchange values are satisfaction ($s'_{\alpha\beta}$) and credit ($v'_{\alpha\beta}$), and β's exchange values are renouncement ($r'_{\beta\alpha}$) and debt ($t'_{\beta\alpha}$).

In addition, because agents need to consider reciprocation and the quality of their interactions in terms of gains and losses of values to judge whether to interact with other agents, they must be aware of their credits, debts and the balance of their exchange values in previous interactions. Thus, agents maintain a *history of exchange values*, which contains detailed information about previous interactions with other agents in which exchange values were acquired or spent.

Each interaction in which an agent participates has an entry in its history of exchange values. Each entry contains information about a specific interaction, including the time at which it took place (**time**), the service that was performed or received by the agent (**srv**), the identification of the partner agent (**prt**), the set of exchange values associated with that interaction (EV or EV'), the balance of the exchange values in each stage of the interaction (**stb**), and the overall balance of the exchange values in the complete interaction (**oab**), which considers exchange values acquired in the provision and reciprocation stages.

For example, consider an agent α that is providing a service srv_1 to another agent β. α does not have any existing debt with β from previous interactions, so the interaction is in the provision stage. In this interaction α has a renouncement of $r_{\alpha\beta} = 0.6$ and gains a credit of $v_{\alpha\beta} = 0.8$, which results in the current set of exchange values

TABLE 6.1: Example of exchange values history for agent α.

time	**prt**	**srv**	**EV**				**EV'**				**stb**	**oab**
			r	**s**	**t**	**v**	**r'**	**s'**	**t'**	**v'**		
10	β	srv_1	0.6	–	–	0.8	–	–	–	–	p	-
41	β	srv_2	–	–	–	–	–	0.6	–	0.8	n	e
60	γ	srv_1	0.6	–	–	0.7	–	–	–	–	p	-
...												

$EV_{\alpha\beta} = (0.6, -, -, 0.8)$. The balance of these exchange values is positive (p), since α's credit is greater than its renouncement, and the results of this interaction are stored in α's history of exchange values, as shown in the first line of Table 6.1.

Now, suppose that α interacts with β again but, this time, α receives a service srv_2 from β in return for the credit it previously acquired, so the interaction is in the reciprocation stage. In this interaction α has a satisfaction of $s'_{\alpha\beta} = 0.6$ and spends a credit of $v'_{\alpha\beta} = 0.8$, which results in the set of exchange values $EV'_{\alpha\beta} = (-, 0.6, -, 0.8)$. Unlike the first interaction, the balance for the reciprocation stage is negative (n), since the credit that α spent is larger than its satisfaction with the service. With the reciprocation stage finished, the interaction between α and β is complete, and α can also determine the overall balance of its exchange values, which in this case is in equilibrium (e), since the gains and losses in both stages are equivalent. The results of this interaction are stored in α's history of exchange values, as shown in the second line of Table 6.1.

Now that we have the basic model, we need to consider how these values are *derived* so that they can be used appropriately. This is described below.

6.4.2 Objective Values

In the provision stage of an interaction, an agent (say β) receives a service (say srv_1) from another agent (say α) and exchange values are instantiated, as discussed above. Renouncement and satisfaction values in this stage are objective values and are determined directly from each agent's evaluation of its effort or utility. In the case of bioinformatics services, we already have a means for such evaluation, as described in detail in Chapter 5, and can simply substitute those values here, so nothing new is needed.

Thus, the *renouncement* of α and the *satisfaction* of β are denoted as follows:

$$r_{\alpha\beta} = evalRenouncement_{\alpha}(srv_1)$$

$$s_{\beta\alpha} = evalSatisfaction_{\beta}(srv_1)$$

where $evalRenouncement_{\alpha}(srv_1)$ is a function that calculates α's evaluation of the provided service in terms of effort, and $evalSatisfaction_{\beta}(srv_1)$ calculates β's evaluation of the received service in terms of benefit. We have shown in Section 5.3.3.2 that, even though agents use the same evaluation method, services are evaluated differently from the perspective of providers and requesters. For example, α's renouncement can be calculated in relation to srv_1's cost, and β's satisfaction can be determined in relation to srv_1's performance.

The objective values in the reciprocation stage of the interaction are determined in the same way as the provision stage. Thus, if, in the reciprocation stage of the interaction between α and β, β provides a service srv_2 to α in return, the *renouncement* of β and the *satisfaction* of α are denoted as follows:

$$r'_{\beta\alpha} = evalRenouncement_{\beta}(srv_2)$$

$$s'_{\alpha\beta} = evalSatistaction_{\alpha}(srv_2)$$

where $evalRenouncement_{\beta}(srv_2)$ is β's evaluation of the provided service in terms of effort, and $evalSatisfaction_{\alpha}(srv_2)$ is α's evaluation of the received service in terms of benefit. Note that the objective values that agents must determine in the reciprocation stage of the interaction are not the same as those for the provision stage.

More problematic, however, are the remaining exchange values, which are subjective values, considered in detail next.

6.4.3 Subjective Values

Subjective values of credit and debt result from the valorisation that the agent receiving a service gives to the provider, as defined previously in Section 6.3.2, so that the origin of subjective values is clear. However, this raises one limitation of Piaget's model, which is how the provider is aware of the receiver's valorisation. As discussed in Section 6.2.4, this issue must be addressed either by explicit communication of the valorisation from the receiver to the provider, or by enabling the provider to model what it believes the receiver's valorisation is based on its own information.

Moreover, subjective values are affected by influences that may distort (by increasing or decreasing) the original value on which they are based, as described in Section 6.3.3. To consider such subjective influences on determining debts and credits, we need to represent influences computationally.

In this context, before determining the subjective values of debt and credit, we address the communication of the debt value from the requester to the provider and the computational representation of influences, as described below.

6.4.3.1 Communication

To allow agents to reason about their interactions in terms of the compensation of provided and received services, it is important that they are aware of the valorisation that others give to their services, so that providers can identify possible disequilibrium situations caused by under-valorisation or over-valorisation of their services. Through explicit communication, the provider can gain accurate knowledge of the valorisation it is receiving, and even if the receiver lies about its valorisation, a resulting disequilibrium situation can then be identified by the provider.

To communicate its valorisation, the receiver needs only to reply to the message containing the service result with an acknowledgement message including its debt value in the content. Thus, if this acknowledgement message is already part of the interaction protocol between agents, this communication does not increase the number of messages exchanged between them. If the acknowledgement to the service result message is not part of the protocol, the number of messages in the total interaction increases only by 1, which does not impose a message overload on the system.

Therefore, based on the above analysis, we assume that the service receiver first determines its debt value, and then *communicates* this debt to the provider as a valorisation.

6.4.3.2 Influences

Influences over subjective values may increase or decrease the values on which they are based. We define influences as tuples $\iota = (\mathit{condition}, \delta)$, where *condition* specifies the situation in which the influence is valid (if the influence is always valid, the condition is *true*), and $\delta \in [-1, 1]$ is the influence intensity, with $\delta < 0$ representing a *negative influence* that decreases the subjective value and $\delta > 0$ representing a *positive influence* that increases the subjective value. Note that the intensity of each influence must be defined according to the desired impact of that influence on the subjective evaluation, such that influences with high impact have intensity close to -1 or 1, those with low impact have intensity close to 0, and those with null impact have intensity equal to 0. The set of all possible influences on an agent's subjective evaluation is denoted as $I = \{\iota_1, .., \iota_n\}$.

We have already discussed in Section 6.3.3 examples of subjective influences that may be present in the kind of cooperative applications we are modeling. Here, a number of conditions can be identified for these influences to be valid, such as $\mathit{HighlySkilled}(prt)$, $\mathit{HigherPosition}(prt)$, $\mathit{UnilateralDependence}(prt)$, $\mathit{Abundant}(srv)$, $\mathit{ReachedExpectation}(srv)$, and $\mathit{Saturated}(prt)$, where prt is the interaction partner and srv is the service being provided or received. Although we consider just this group of influences, others may apply to different applications or domains, and thus additional

Algorithm 2 Algorithm for the influence function *influence*(prt, srv).

```
1: input: $prt$,$srv$
2: $total\iota = 0$
3: for all $(\iota_j = (condition_j(prt, srv), \delta_j)) \in I$ do
4:    if $condition_j(prt, srv) = true$ then
5:       $total\iota = total\iota + \delta_j$
6:    end if
7: end for
8: output: $total\iota$
```

conditions can be identified (but for ease of presentation, we discuss only the influences above, instead of presenting an extensive list).

For example, if agent β exhibits unfair behaviour that always has a negative effect on its subjective evaluation (that is, it always decreases its debt), such an influence is represented by $\iota_1 = (true, x)$, where $0 > x > -1$. Similarly, if β's unilateral dependence with the interacting partner (identified as agent α) has a positive influence on its subjective evaluation (that is, it may increase the debt), this influence is represented by $\iota_2 = (UnilateralDependence(\alpha), x)$, where $1 > x > 0$, and the condition $UnilateralDependence(\alpha)$ is true if β depends on α but α does not depend on β. Thus, if there is a mutual dependence between the two agents, influence ι_2 does not affect β's subjective judgement (it is not valid).

The set of influences that an agent considers in its subjective evaluation depends on the agent's design objectives (for example, agents can be designed to have fair or unfair behaviour), and on what is important to capture from the environment (for example, in a system with no social hierarchy, agents do not need to consider the influence of social position).

6.4.3.3 Determining Influenced Debt and Credit

Until now, we have seen that an agent receiving a service first determines its debt value and then communicates this debt to the agent providing the service, which takes it as a credit. The process of determining debt and credit can be affected by subjective influences on those values.

To determine the debt that is acquired when a requester receives a service, we must consider the satisfaction of the requester with this service, and the subjective influences that may apply over this satisfaction. Given the representation and use of influences described previously, β determines its *debt* value as follows:

$$t_{\beta\alpha} = k(s_{\beta\alpha} + \mathit{influence}(\alpha, srv))$$

where the function $\mathit{influence}(\alpha, srv)$ specifies the influence that β applies to its objective evaluation $s_{\beta\alpha}$ when interacting with partner agent α and receiving service srv, and the function $k(x)$ constrains the resulting value to a $[0, 1]$ scale.

The function $\mathit{influence}(\alpha, srv)$ is described in Algorithm 2 and calculates the total influence $total\iota$ over a value (which for the requester is $s_{\beta\alpha}$). To combine influences, we assume that positive and negative influences can cancel each other out, and that two or more positive (or negative) influences have greater impact on the original value than one influence alone. Thus, the total influence over the original value is determined in Algorithm 2 by adding the intensities (δ_j) of all individual influences ι_j in the set I that have $condition_j(prt, srv) = true$.

Suppose that agent β has an influence set $I = \{\iota_1, \iota_2\}$, where $\iota_1 = (Abundant(srv), -0.4)$ and $\iota_2 = (UnilateralDependence(prt), 0.2)$. This means that when β is requesting a service (srv) it undervalues the provider if this service is abundant in the system, and β overvalues the provider if it depends on that provider (prt) but the provider does not depend on β. If both conditions are true, the total influence calculated according to Algorithm 2 is $total\iota = -0.2$ (as a result of $-0.4 + 0.2$). If β's satisfaction value is $s_{\beta\alpha} = 0.6$, this results in a debt of $t_{\beta\alpha} = 0.4$ (from $k(0.6 - 0.2)$). Here, the influence of service demand decreases β's debt in relation to its satisfaction. If only the condition $UnilateralDependence(prt)$ is true, the total influence on $s_{\beta\alpha}$ is $total\iota = 0.2$, which results in a debt of $t_{\beta\alpha} = 0.8$ (from $k(0.6 + 0.2)$). Here, in contrast, the influence on the agent's subjective evaluation increases its debt in relation to its satisfaction. In this case, if β's satisfaction is $s_{\beta\alpha} = 0.9$, the function $k(0.9 + 0.2)$ limits the value to 1, resulting in a debt of $t_{\beta\alpha} = 1$.

After the requester has determined its debt, it communicates this debt to the provider agent to express its valorisation of the provider's service. The provider then accepts this valorisation as a credit for the future. However, as for the debt value, subjective influences may alter the original valorisation, increasing or decreasing the provider's credit. Given the same representation and use of influences described previously, a provider α determines its *credit* value as follows:

$$v_{\alpha\beta} = k(valorisation(\beta) + \mathit{influence}(\beta, srv))$$

where the function $valorisation(\beta)$ just reads the valorisation ($t_{\beta\alpha}$) communicated by the requester (β) through the service completion acknowledgement message, and the function $\mathit{influence}(\alpha, srv)$, described in Algorithm 2, calculates the total influence over the valorisation that α received from β. One possible influence over the provider's credit is *saturation*, as described in Section 6.3.3.

If there is no influence on the accepted credit, there is a direct correspondence between β's debt and α's credit, $v_{\alpha\beta} = t_{\beta\alpha}$. This direct correspondence is based on the assumption that agents do not lie about their debts when communicating them, but if they do lie (such that $v_{\alpha\beta} < t_{\beta\alpha}$), causing a provider's balance of exchange values to be negative (that is, the renouncement is greater than the credit), such behaviour may *reduce* the chances of future interactions with that provider. Moreover, the direct correspondence between the provider's credit and the requester's debt allows the provider to identify any under-valorisation or over-valorisation given by the requester.

6.4.4 Reciprocation Values

In the reciprocation stage of the interaction, the subjective values of credit and debt acquired in the provision stage can be realised as objective values. Here, a credit is spent by requesting a service that has some satisfaction value, and a debt is paid by providing a service that has a renouncement value.

Suppose that two agents α and β have interacted previously and, as a result of these interactions, α acquired a credit with β and β acquired a debt with α. Now, in the reciprocation stage of the interaction, α can spend its credit ($v'_{\alpha\beta}$) by asking β for a service in return. Agent β may or may not admit the corresponding debt ($t'_{\beta\alpha}$) but, if it does, it provides the requested service on α's behalf. The service provided by β has a renouncement for β ($r'_{\beta\alpha}$) and a satisfaction for α ($s'_{\alpha\beta}$).

Unlike the provision stage of the interaction in which subjective values must be determined, when agents are in the reciprocation stage of the interaction, their subjective values already exist (and were acquired in the provision stage). Thus, the determination of credits and debts as reciprocation values involves only retrieving these values from where they are stored. In the case of single interactions, this task is simple but, when subjective values of credit and debt accumulate over time, we must determine how much of an accumulated credit an individual should spend when requesting a service, and how much of an accumulated debt an individual should pay when reciprocating.

The way accumulated credits are spent and accumulated debts are paid is not clear in Piaget's model, and there can be different approaches to determining reciprocation values. However, it is desirable that solutions for determining reciprocation values allow the following.

- Agents should be able to compare the services they provide with the services they receive in reciprocation, so that they can identify interaction partners that provide services with similar quality to their own services.
- Agents should be able to make use of high valorisations received in the provision stage, yielding a high credit.

- Service reciprocation should cope with environments in which agents exchange services with different levels of effort or computational demand (so that a service with higher effort can be reciprocated with more than one service with lower effort).

As mentioned before, reciprocation values correspond to existing values of credit or debt, acquired in previous interactions, in the provision stage. This correspondence between service provision and reciprocation can be direct, such that one service provision corresponds to one reciprocation, or multiple, such that one or more service provisions correspond to one or more reciprocations. Based on this, there are two possible approaches to retrieving credits and debts, as follows.

1. *Direct correspondence* (DC): each credit or debt resulting from an interaction is seen as a unit that can be used for one reciprocation. The problem is then to choose which credit (unit) to spend, or which debt to pay in each reciprocation when they accumulate over time.
2. *Creditor's choice* (CC): each credit resulting from an interaction can be spent in parts by receiving more than one service in return (in different interactions), or accumulated credits resulting from many interactions can be spent by receiving one service in return. The problem is then to choose the amount of credit to spend or the amount of debt to pay.

The advantage of the DC approach is that the comparison between services that agents provide with services that they receive in reciprocation is straightforward (since there is a one to one correspondence between them). However, making use of high valorisations is difficult, since agents would need to receive one service that has the worth of such high credit in reciprocation. In addition, it restricts applications in which services have different computation demands, since all interactions in which a highly demanding service is reciprocated with one trivial service will result in a negative balance of exchange values for the agent providing the highly demanding service.

In the CC approach, one provision may yield one or more reciprocations, and the requester (or creditor) *chooses* the amount of credit to spend with each reciprocation. To do so, the requester might analyse the quality of the reciprocated service by comparing its satisfaction with the reciprocated service against the average satisfaction associated with the same service in previous interactions. The idea is that a requester, say α, can choose how much of the credit received from a partner agent, say β, to spend each time, depending on the service received. Thus, if there is a remaining credit and α is interested in receiving another service from the same agent (that is, if the satisfaction with the reciprocated service is greater than or equal to the average satisfaction), it spends only the credit that is equivalent to the satisfaction value, so that $v'_{\alpha\beta} = s'_{\alpha\beta}$. However, if the provider β reciprocates a low quality service causing α's satisfaction to be less than

Algorithm 3 Algorithm for instantiating $credit(\beta, s'_{\alpha\beta})$ according to the CC approach.

```
1: input: $\beta, s'_{\alpha\beta}$
2: $v^{acc}_{\alpha\beta} = \sum_1^n v'_{\alpha\beta_n}$
3: **if** $|v^{acc}_{\alpha\beta} - s'_{\alpha\beta}| > tolerance$ **then**
4:    $average\_satisfaction = CalculateAverage(srv)$
5:    **if** $s'_{\alpha\beta} \geq average\_satisfaction$ **then**
6:       $choice = s'_{\alpha\beta}$
7:    **else**
8:       $choice = v^{acc}_{\alpha\beta}$
9:    **end if**
10: **else**
11:    $choice = v^{acc}_{\alpha\beta}$
12: **end if**
13: output: $choice$
```

Algorithm 4 Algorithm for instantiating $debt(\alpha, reciprocation\_value)$ according to the CC approach.

```
1: input: $\alpha, reciprocation\_value$
2: $t^{acc}_{\beta\alpha} = \sum_1^n t'_{\beta\alpha_n}$
3: $t'_{\beta\alpha} = minimum(t^{acc}_{\beta\alpha}, reciprocation\_value)$
4: output: $t'_{\beta\alpha}$
```

average, α may not be interested in requesting another service from the same agent, and may spend all accumulated credit, which we now define as $v^{acc}_{\alpha\beta}$, so that $v'_{\alpha\beta} = v^{acc}_{\alpha\beta}$. This solution requires that α communicates $v'_{\alpha\beta}$ to β, in a manner similar to the process of determining subjective values by the requester communicating its valorisation to the provider (and this is possible because the credit to be spent is determined after the service was received and α has calculated its satisfaction value).

Although with the CC approach both the reciprocation credit and the reciprocation debt are determined based on the creditor's choice, the maximum amount of credit that the requester can spend is all the credit that it has received from β in previous interactions. This guarantees that β will not pay more debt than it owed.

The advantages of the CC approach to determining reciprocation values is that it allows an agent to identify if a reciprocated service compensates for any service that was provided in previous interactions, since the agent decides how much of an accumulated credit to spend based on the quality of the reciprocated service. It also allows agents to make use of high valorisations and copes with an environment in which agents provide services with different computational demands, since one provided service can yield more than one reciprocated service.

Since the CC approach addresses all issues in determining reciprocation values, we adopt this approach instead of the DC approach. Therefore, the reciprocation values are determined as follows:

$$v'_{\alpha\beta} = credit(\beta, s'_{\alpha\beta})$$

$$t'_{\beta\alpha} = debt(\alpha, reciprocationValue(\alpha))$$

where the function $credit(\beta, s'_{\alpha\beta})$ is shown in Algorithm 3, the function $reciprocationValue(\alpha)$ just reads the credit to be spent in this reciprocation (communicated by α through the service completion acknowledge message), and the function $debt(\alpha, reciprocationValue(\alpha))$ is instantiated as in Algorithm 4 (ensuring that the reciprocation value communicated by α is not greater than the debt that β has accumulated with α in previous interactions).

In Algorithm 3, $v^{acc}_{\alpha\beta}$ is the credit that α has accumulated with β, and the difference between this accumulated credit and α's current satisfaction being higher than the tolerance value (expressed in line 2) indicates that there is a remaining credit, which can be spent in parts. The function $CalculateAverage(srv)$ uses α's history of exchange values to calculate α's average satisfaction with the reciprocated service in previous interactions. In Algorithm 4, $t^{acc}_{\beta\alpha}$ is the debt that β has accumulated with α.

The disadvantage of this approach is that malicious agents may always spend low amounts of credit, even if the reciprocated service was good, just to exploit the partner agent. However, every time a low credit is spent, it causes a negative balance of exchange values for the agent in debt when its renouncement to provide the service is greater than the debt it is paying ($r'_{\beta\alpha} > t'_{\beta\alpha}$). Therefore, the agent in debt can identify this situation and deny the request, avoiding the action of such malicious agents.

6.4.5 Devaluation

Now, suppose an agent in debt, say β, may not admit its debt when asked for a reciprocation by another agent, say α. If this happens, the interaction is considered to have a negative outcome for α, since a credit could not be realised into any objective value. We call this situation a *devaluation* of β by α.

To represent this devaluation, the following is added to α's history of exchange values: $(time, \beta, srv, (0, 0, 0, 0), (0, 0, 0, 0), \text{“}n\text{”}, \text{“}n\text{”})$, where the sets of exchange values for both stages of the interaction are null, and both the stage and overall balances of exchange values are negative (n) (a description of all elements of the tuple can be found in Section 6.4.1). Although there are no exchange values involved, since the exchange did not take place (and thus, exchange values do not increase or decrease), the balances are set as negative to indicate that there was no reciprocation. Note that since the credit is not spent, the requester agent can still try to request in the future (although it may prefer other providers more likely to reciprocate).

An agent can identify the interaction partners that it devalued by searching its history of exchange values for those whose interactions have no exchange values but a negative stage and overall balances. These agents are not seen as good cooperation partners, so devaluation may be used to restrict the selection of interaction partners both when providing or requesting a service.

6.5 Balance of Exchange Values

Since agents gain and lose values in each interaction, it is reasonable to say that successful interactions are those in which the agent's gain is at least equal to its loss and that unsuccessful interactions are those in which the agent's gain is smaller than its loss (and the same is valid for the complete interaction cycle, when considering overall gains and losses). More specifically, a *successful* interaction for an agent is one in which the balance of its exchange values is positive or in equilibrium, while an *unsuccessful* interaction is one in which the balance of its exchange values is negative.

To compare exchange values so that their relative balance can be determined, the equivalence operation on exchange values must be determined so that small differences in evaluation are ignored. To do so, a *tolerance threshold* ($tol \in [0,1]$) is set, and evaluation differences below this threshold are ignored so that two values are considered equal.

An agent's balance of exchange values is given by the function $balance(g,l)$, where g is any value representing a gain and l is any value representing a loss. This function is defined as follows:

$$balance(g,l) = \begin{cases} equilibrium, & |g-l| \leq tol; \\ positive, & g > l; \\ negative, & g < l. \end{cases}$$

Note that, in this function, the conditions are exclusive and considered in order, so that the tolerance value is not needed in the second and third conditions because if the first condition is false, then the difference between gain and loss ($|g-l|$) is greater than the tolerance.

The balance of exchange values can be determined in relation to three different stages of the interaction: the provision stage, the reciprocation stage, and the complete interaction. A summary of the exchange values that represent gains and losses in each stage of the interaction between two agents, α and β, in which α provides a service in the provision stage, and β returns another service in the reciprocation stage of the

Table 6.2: Gains and Losses of Values in Both Stages of the Interaction.

Interaction	**Agent**	**Gain**	**Loss**
provision stage	β	$s_{\beta\alpha}$	$t_{\beta\alpha}$
	α	$v_{\alpha\beta}$	$r_{\alpha\beta}$
reciprocation stage	β	$t'_{\beta\alpha}$	$r'_{\beta\alpha}$
	α	$s'_{\alpha\beta}$	$v'_{\alpha\beta}$

Table 6.3: Examples of Balance of Exchange Values.

Agent	**EV**				**EV’**				**stb**	**oab**
	r	**s**	**t**	**v**	**r’**	**s’**	**t’**	**v’**		
β	–	0.4	0.4	–	–	–	–	–	e	-
α	0.6	–	–	0.4	–	–	–	–	n	-
β	–	–	–	–	0.5	–	0.4	–	e	e
α	–	–	–	–	–	0.6	–	0.4	p	e

interaction, is shown in Table 6.2. The balance of the complete interaction compares gains and losses in both stages of the interaction.

For example, according to Table 6.2, in the provision stage, β determines its balance of exchange values with $balance(s_{\beta\alpha}, t_{\beta\alpha})$, and α determines its balance of exchange values with $balance(v_{\alpha\beta}, r_{\alpha\beta})$. An example is shown in lines 1 and 2 of Table 6.3, in which the stage balance (**stb**) is in equilibrium (e) for β and negative (n) for α, assuming a tolerance threshold of $tol = 0.1$.

In the reciprocation stage, α, which is now the requester, determines its balance of exchange values with $balance(s'_{\alpha\beta}, v'_{\alpha\beta})$, and β, which is now the provider, determines the balance of its exchange values with $balance(t'_{\beta\alpha}, r'_{\beta\alpha})$. An example is shown in lines 3 and 4 of Table 6.3, in which the stage balance (**stb**) is in equilibrium (e) for β and positive (p) for α, assuming a tolerance threshold of $tol = 0.1$.

When the interaction is complete, agents can determine the overall balance of their exchange values. It is important to calculate the overall balance of exchange values in addition to the balance in each stage because gains in one stage may compensate for losses in another stage, resulting in an equilibrium cooperation between the two agents even though there may have been disequilibrium in each separate stage.

To determine the overall balance of the complete interaction, agents compare the total gain and the total loss in both stages. Thus, α determines its overall balance with $balance((v_{\alpha\beta}+s'_{\alpha\beta}), (r_{\alpha\beta}+v'_{\alpha\beta}))$, and β determines its overall balance with $balance((s_{\beta\alpha}+t'_{\beta\alpha}), (t_{\beta\alpha} + r'_{\beta\alpha}))$. Following the examples in Table 6.3, the overall balance of exchange values (**oab**) is in equilibrium for β (in line 3) and also for α (in line 4). Note that in the provision stage of the interaction, α’s balance of exchange values is negative, and in the reciprocation stage of the interaction α’s balance is positive, which results in an overall balance that is in equilibrium. Thus, from α’s perspective, even though there was disequilibrium in different stages of the interaction, the final balance is in equilibrium,

Algorithm 5 Determining exchange values in the provision stage when the agent is a provider (α).

```
1: input: $\beta, srv_1$
2: $result = Perform(srv_1)$
3: $r_{\alpha\beta} = evalRenouncement_{\alpha}(result)$
4: $Send(\beta, NOTIFY, result)$
5: $Wait(\beta, ACK, valorisation)$
6: $v_{\alpha\beta} = k(valorisation(\beta) + influence(\beta, srv_1))$
7: $EV_{\alpha\beta} = (r_{\alpha\beta}, -, -, v_{\alpha\beta})$
8: $b_1 = balance(v_{\alpha\beta}, r_{\alpha\beta})$
9: $UpdateHistory(1, timestamp, srv_1, \beta, EV_{\alpha\beta}, b_1, -)$
10: output: $EV_{\alpha\beta}$
```

Algorithm 6 Determining exchange values in the provision stage when the agent is a requester (β).

```
1: input: $\alpha, srv_1$
2: $Wait(\alpha, NOTIFY, result)$
3: $s_{\beta\alpha} = evalSatisfaction_{\beta}(result)$
4: $t_{\beta\alpha} = k(s_{\beta\alpha} + influence(\alpha, srv_1))$
5: $Send(\alpha, ACK, t_{\beta\alpha})$
6: $EV_{\beta\alpha} = (-, s_{\beta\alpha}, t_{\beta\alpha}, -)$
7: $b_2 = balance(s_{\beta\alpha}, t_{\beta\alpha})$
8: $UpdateHistory(1, timestamp, srv_1, \alpha, EV_{\beta\alpha}, b_2, -)$
9: output: $EV_{\beta\alpha}$
```

which indicates a successful cooperation with agent β.

The key idea here is that the balance of exchange values indicates the outcome of an interaction in terms of what the agent gained and lost and, therefore, it may be used to distinguish beneficial cooperations and provide a means for agents to select future interaction partners.

6.6 Interaction Steps

According to the exchange values model, interactions between two generic agents α and β occur in two stages: the provision stage, and the reciprocation stage. Although in both stages the protocol for requesting and providing services is the same, the steps followed by the agents through the interaction are different, since the exchange values for each stage are determined in different ways.

When agents α and β are in the provision stage of the interaction, which starts with β requesting a service srv_1 from α, and α accepting the request, α follows the steps in Algorithm 5, and β follows the steps in Algorithm 6.

According to Algorithm 5, α performs srv_1, determines its renouncement value, sends

Algorithm 7 Determining exchange values in the reciprocation stage when the agent is a requester.

```
1: input: $\beta, srv_2$
2: $Wait(\beta, NOTIFY, result)$
3: $s'_{\alpha\beta} = evalSatisfaction_{\alpha}(result)$
4: $v'_{\alpha\beta} = credit(\beta, s'_{\alpha\beta})$
5: $Send(\beta, ACK, v'_{\alpha\beta})$
6: $EV'_{\alpha\beta} = (-, s'_{\alpha\beta}, -, v'_{\alpha\beta})$
7: $b_3 = balance(s'_{\alpha\beta}, v'_{\alpha\beta})$
8: $b_4 = balance((v_{\alpha\beta} + s'_{\alpha\beta}), (r_{\alpha\beta} + v'_{\alpha\beta}))$
9: $UpdateHistory(2, timestamp, srv_2, \beta, EV'_{\alpha\beta}, b_3, b_4)$
10: output: $EV'_{\alpha\beta}$
```

Algorithm 8 Determining exchange values in the reciprocation stage when the agent is a provider.

```
1: input: $\alpha, srv_2$
2: $result = Perform(srv_2)$
3: $Send(\alpha, NOTIFY, result)$
4: $r'_{\beta\alpha} = evalRenouncement_{\beta}(result)$
5: $Wait(\alpha, ACK, reciprocationValue)$
6: $t'_{\beta\alpha} = debt(\alpha, reciprocationValue)$
7: $EV'_{\beta\alpha} = (r'_{\beta\alpha}, -, t'_{\beta\alpha}, -)$
8: $b_5 = balance(t'_{\beta\alpha}, r'_{\beta\alpha})$
9: $b_6 = balance((s_{\beta\alpha} + t'_{\beta\alpha}), (t_{\beta\alpha} + r'_{\beta\alpha}))$
10: $UpdateHistory(2, timestamp, srv_2, \alpha, EV'_{\beta\alpha}, b_5, b_6)$
11: output: $EV'_{\beta\alpha}$
```

a notification message ($NOTIFY$) to β containing the service result, and waits for β's acknowledgement message (ACK), which contains β's valorisation of srv_1. Meanwhile, β waits for the notification containing the service result, as shown in the first line of Algorithm 6. When the notification arrives, β determines its satisfaction value based on the service result, determines its debt value, and sends an acknowledgement message to α containing the debt value. After that, β determines its current set of exchange values, the balance of its exchange values (b_2), and updates its history of exchange values with the interaction results (line 7). The function *UpdateHistory*() takes the stage of the interaction, a time-stamp at which the interaction took place, the service involved in the interaction, the partner agent, the current set of exchange values for the corresponding stage, and the balance of exchange values, and adds a new entry in the agent's history of exchange values. At this point, β finishes the provision stage of the interaction with α. When α receives the acknowledgement message containing β's valorisation, it determines its credit value as shown in line 5 of Algorithm 5. After that, α determines its current set of exchange values, the balance of its exchange values (b_1), and updates its history of exchange values. At this point, α also finishes the provision stage of the interaction with β.

Algorithm 9 Devaluation of β by α.

```
1: $EV'_{\alpha\beta} = (0, 0, 0, 0)$
2: $UpdateHistory(2, timestamp, srv_2, \beta, EV'_{\alpha\beta}, \text{“}n\text{”}, \text{“}n\text{”})$
```

In the future, if α wants to spend its credit by requesting a service srv_2 from β in return, and if β accepts the request, they start the *reciprocation stage* of the interaction. The provider β follows the steps in Algorithm 8, and the requester α follows the steps in Algorithm 7, which are similar to those for the provision stage. When the reciprocation stage of the interaction is finished, there was reciprocity between the two agents.

If in the reciprocation stage, β does not accept α's request, a devaluation occurs from α to β, and α follows the steps in Algorithm 9.

6.7 A Worked Example

We take as a practical example a cooperative bioinformatics system in which agents provide and request protein identification services (with configurations C_1 or C_2), such as those described in Section 5.5. Agents in the system evaluate these bioinformatics services using the evaluation method proposed in Chapter 5 (so that the agents' satisfaction and renouncement values are based on the results for service evaluation presented in Section 5.5). To represent agents having different perspectives over service evaluation, we assume that, when using the evaluation method, agents can either see all service properties as having the same importance (equal evaluation weights), or they can see some properties as having more importance than others (distinct evaluation weights). We assume that the subjective evaluation is influenced by the expected service quality only, which we will refer to as expectation influence, defined as $\iota_1 = (\neg ReachedExpectation(srv), 0.2)$, so that the requester's debt value is increased by 0.2 if the satisfaction with the received service $\boldsymbol{srv}$ is smaller than the expected satisfaction with services with the same configuration from previous interactions. (Note that the condition $ReachedExpectation(srv)$ indicates that the service has reached the expected service quality).

Now, consider an interaction in the provision stage in which agent α provides the service *Tandem Local* with configuration C_2, TL(C_2), to agent λ (the first interaction in Figure 6.2). After the service is provided, α and λ evaluate the service to calculate their objective values. Since the results for this evaluation already exist in Section 5.5, we use the results in Tables 5.5 and 5.4, so that λ's satisfaction value is the same as the final service evaluation, $s_{\lambda\alpha} = 0.879$ (using equal evaluation weights), and α's renouncement value is the same as the service cost evaluation, $r_{\alpha\lambda} = 5.4E\text{-}5$. To determine its debt value, λ takes its satisfaction and checks the expectation influence over this value. Since TL(C_2) reaches expectation, the influence does not apply and λ's debt is equal to its

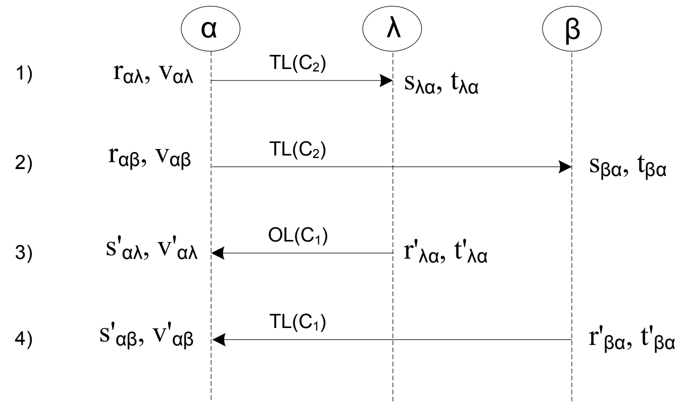

α
λ
β
1)
$r_{\alpha\lambda}, v_{\alpha\lambda}$
TL(C₂)
$s_{\lambda\alpha}, t_{\lambda\alpha}$
2)
$r_{\alpha\beta}, v_{\alpha\beta}$
TL(C₂)
$s_{\beta\alpha}, t_{\beta\alpha}$
3)
$s'_{\alpha\lambda}, v'_{\alpha\lambda}$
OL(C₁)
$r'_{\lambda\alpha}, t'_{\lambda\alpha}$
4)
$s'_{\alpha\beta}, v'_{\alpha\beta}$
TL(C₁)
$r'_{\beta\alpha}, t'_{\beta\alpha}$

FIGURE 6.2: Interactions between α, β, and λ.

TABLE 6.4: History of exchange values for agent α.

time	**prt**	**srv**	**EV**				**EV'**				**stb**	**oab**
			r	**s**	**t**	**v**	**r'**	**s'**	**t'**	**v'**		
29	γ	$M(C_1)$		0.429	0.429	–	–	–	–	–	e	-
51	θ	$TR(C_2)$	–	0.478	–	0.478	–	–	–	–	e	–
55	λ	$TL(C_2)$	5.4E-5	–	–	0.879	–	–	–	–	p	–
60	β	$TL(C_2)$	5.4E-5	–	–	0.847	–	–	–	–	p	–
70	λ	$OL(C_1)$	–	–	–	–	–	0.04	–	0.879	n	–

satisfaction, $t_{\lambda\alpha} = 0.879$. Agent λ then communicates its debt to α, which takes it as its credit for future interactions, $v_{\alpha\lambda} = 0.879$ (note that the credit value is not altered by α since the expectation influence does not apply here).

At the end of this interaction, the sets of exchange values for the different agents are $EV_{\alpha\lambda} = \{5.4E\text{-}5, -, -, 0.879\}$ for α and $EV_{\lambda\alpha} = \{-, 0.879, 0.879, -\}$ for λ. The balance of exchange values for each interacting agent is determined using a tolerance value of 0.15, so that the balance is in equilibrium for λ (since $s_{\lambda\alpha} = t_{\lambda\alpha}$) and positive for α (since $r_{\alpha\lambda} < v_{\alpha\lambda}$). As the final step, agents update their history of exchange values, and the entry in α's history of exchange values corresponding to this interaction is shown in the third line of Table 6.4.

Consider now a second interaction in which agent α provides the same service TL(C_2) but to agent β (the second interaction in Figure 6.2). The exchange values for both agents are determined as in the previous example and, this time, the set of exchange values for α is $EV_{\alpha\beta} = \{5.4E\text{-}5, -, -, 0.847\}$, and the set of exchange values for β is $EV_{\beta\alpha} = \{-, 0.847, 0.847, -\}$, since, as opposed to λ, β uses distinct evaluation weights (as shown in the last column of Table 5.5). The balance of exchange values is in equilibrium for β and positive for α. The entry for this interaction in α's history of exchange values is shown in the fourth line of Table 6.4.

Later on, α asks λ for the service *Omssa Local* with configuration C_1, OL(C_1), in return for its credit (the third interaction in Figure 6.2). After the service is performed, both agents calculate their objective values, which are $r'_{\lambda\alpha} = 0.855$ and $s'_{\alpha\lambda} = 0.04$, and for this

α uses distinct evaluation weights (according to Tables 5.5 and 5.4). Agent α determines the credit to be spent as in Algorithm 3. Since the service provided by λ has a much lower satisfaction value than the average satisfaction taken from α's history of exchange values (which is 0.429), α spends all its accumulated credit, $v'_{\alpha\lambda} = 0.879$ (showing that it is not interested in asking λ for another service). Agent α then communicates this value to λ, which determines its debt to be paid as in Algorithm 3, according to which λ's debt to be paid with the reciprocation is $t'_{\lambda\alpha} = 0.879$ (since the credit communicated by α is not greater than λ's accumulated debt with this agent). At the end of the reciprocation stage, the sets of exchange values are $EV'_{\alpha\lambda} = \{-, 0.04, -, 0.879\}$ for α and $EV'_{\lambda\alpha} = \{0.855, -, 0.879, -\}$ for λ, and the balance of exchange values is negative for α (since $v'_{\alpha\lambda} > s'_{\alpha\lambda}$) and in equilibrium for λ (since $r'_{\lambda\alpha} = t'_{\lambda\alpha}$, when using a tolerance value of 0.15).

A negative balance of exchange values indicates that the cooperation was not successful for α (due to α receiving a poor quality service in reciprocation), and thus α tries a different provider for the next interaction. It then asks β to provide service TL(C_1) in reciprocation, since it has a credit with β (the fourth interaction in Figure 6.2). After the service is reciprocated, the agents' objective values are $r'_{\beta\alpha} = 1.4E\text{-}16$ and $s'_{\alpha\beta} = 0.302$ (according to Tables 5.5 and 5.4). The credit to be spent by α in this reciprocation is the same as its satisfaction, $v'_{\alpha\beta} = 0.302$, since the service provided by β has a satisfaction that is higher than the average taken from α's history of exchange values (which is 0.234)[4], and thus α is interested in keeping its remaining credit to request another reciprocation from β in a future interaction. At the end of the reciprocation stage, the sets of exchange values for both agents are $EV'_{\alpha\beta} = \{-, 0.302, -, 0.302\}$ for α and $EV'_{\beta\alpha} = \{1.4E\text{-}16, -, 0.302, -\}$ for β, and the balance of exchange values is in equilibrium for α (since $v'_{\alpha\beta} = s'_{\alpha\beta}$) and positive for β (since $r'_{\beta\alpha} < t'_{\beta\alpha}$, when using a tolerance value of 0.15).

In the future, based on the analysis of its exchange values in previous interactions, it is likely that α will maintain its cooperation with β, since its balances of exchange values when interacting with β are in equilibrium or positive, and that α will cease its cooperation with λ, since α identifies a negative balance of exchange values caused by receiving a poor quality service from λ as a reciprocation.

6.8 Conclusion

When analysing non-monetary exchanges in human societies it is easy to observe that, even though interacting individuals do not always correctly interpret each other's valorisations, and misunderstandings often arise, such exchanges do work and individuals

[4]Note that the tolerance value is not used in this comparison since we are not comparing two exchange values.

exchange services without the precision of monetary compensation. Theories of social exchanges are aimed at *modeling* human social interactions so that we can *understand* the process of non-monetary exchanges and why some interactions continue over time while others cease. Using such social theories to model cooperative interactions in agent-based systems aims at providing the basis for motivated and effective cooperations among autonomous agents that operate in open systems with free services.

In this context, the key contribution of this chapter is a novel computational framework for motivating and modelling non-monetary cooperative interactions, based on the theory of exchange values. To develop such a framework, we have addressed the limitations of Piaget's theory for a computational representation of exchange values, proposed a valorisation system to determine subjective credits and debts, and developed a computational representation for subjective influences on those credits and debts.

We argue that the exchange values approach is suitable for addressing the issue of motivating interactions in cooperative, non-monetary applications, especially in cases of deferred reciprocation, in the sense that exchange values provide a system of credits and debts that motivates interactions by giving expectations of future gains (for example, a credit that is gained by performing a service can be charged in the future).

However, for non-monetary interactions between self-interested agents, reciprocation by itself may not be enough to maintain a cooperation, since there must be a certain tradeoff between efforts and benefits. Indeed, our exchange values model allows agents to analyse the outcome of interactions in terms of whether services they receive compensate for services they provide.

Finally, the information related to exchange values, which is provided by the computational framework, can be used by agents to reason about their past interactions with others so that they can better select their future interactions.

Chapter 7

Exchange Values for Provider Selection

7.1 Introduction

The previous chapters have addressed the foundations for decision-making over interactions in open cooperative systems with free services, but we have not yet considered how these elements can be used to support the selection of service providers in such systems from among multiple possible providers. In particular, with dynamic services in open systems, requesters must use updated information about available providers and the quality of their services, since these may change from one selection process to another. In addition, when service provision is free of charge, and interactions between autonomous agents are based on non-monetary cooperations, finding an available provider might take a long time, since agents may not accept all service requests if they are not willing to cooperate, or may need to limit the number of simultaneous requests they can accept due to computational resource constraints. In this context, since services are free, requesters need to use alternatives to monetary compensation, such as the future benefits of cooperative relationships, to influence providers to accept their requests. Thus, in addition to considering the differences in the services being provided, agents requesting services need to take into account existing or potential relationships with candidate providers to find those more likely to cooperate. This gives rise to the following requirements for provider selection.

- *Dynamic selection with analysis of service evaluations*: to cope with providers joining or leaving the system over time, and with changes in service performance that occur from one execution to another, the selection process must be repeated every time a service is needed and must take into account the performance of services received in previous interactions.

- *Use of cooperative relationships*: to avoid requests taking too long to be accepted, existing or potential relationships with candidate providers must be taken into account to find those more likely to accept requests. Otherwise, the efficiency of an agent in achieving its goal may be compromised if there are time constraints, or if the goal involves performing several interdependent tasks where delays in finding available providers for individual tasks may result in a significant overall delay.

To effectively select providers in dynamic applications with non-monetary cooperations, in this chapter we propose a provider selection mechanism to meet the requirements above in support of the task of finding suitable providers. The chapter starts with a description of the selection process that is carried out by service requesters, in Section 7.2. The criteria used for selecting providers, including service evaluation, and exchange values, are presented in Section 7.3, and the selection strategies which use these criteria to analyse and order alternative providers as part of the selection process are introduced in Section 7.4. The chapter finishes with conclusions in Section 7.5.

7.2 The Selection Process

The process of selecting providers is dynamic, taking place every time a service is needed and using updated information about providers; it is achieved by analysing all possible providers for a needed service and ordering them by preference at execution time[1]. When an agent needs to request a service for which there are multiple providers, it should choose those offering the best service (in terms of quality, performance, or other attributes the requester considers relevant), and with which it has existing or potential cooperative relationships (and may thus be considered as more likely to cooperate), so that the request does not take a long time to be accepted.

In this context, the selection process consists of choosing the *best* provider, from a set of possible providers. This is achieved through a general decision-making algorithm to select a provider for any needed service, as described in Algorithm 10. Here, and in the remainder of this chapter, we use the following notation to describe the selection process: we take β to be the requesting agent, srv_i to be the requested service, $P = \{\alpha_1, .., \alpha_n\}$ to be a set of possible providers for srv_i, and P_o to be a sequence of the elements of P ordered by some predetermined criteria. Moreover, we assume that the communication between requester and providers follows a simple request-reply protocol.

The general decision-making algorithm for provider selection, detailed in Algorithm 10, receives as input the set of possible providers P, and applies a selection strategy to determine the sequence P_o from the elements of P, according to some criteria (and

[1] In the case of large systems, the set of possible providers can be reduced by optimisation, for example by filtering it according to some criteria before selection.

Algorithm 10 General decision-making algorithm to select a provider for srv_i.

```
1: input: $P, srv_i$
2: $acceptance = false$
3: $P_o = ProvSelectStrategy(P)$
4: $j = 0$
5: while ($\neg$ $acceptance$) and ($\alpha_{j+1} \in P_o$) do
6:     $j = j + 1$
7:     Send(REQUEST,$\alpha_j$,$srv_i$)
8:     Wait(REPLY,$\alpha_j$)
9:     $acceptance = Accepts(\alpha_j, srv_i)$
10: end while
11: if $acceptance$ then
12:     output: $acceptance, \alpha_j$
13: else
14:     output: $acceptance$
15: end if
```

represented by the function $ProvSelectStrategy(P)$ in Algorithm 10). After determining the preferred providers, a request is sent to the first agent in P_o. If it accepts the request then no further action is necessary (the function $Accepts(\alpha_j, srv_i)$ checks whether the reply was positive); if not, the request is sent to the next provider in the list, and then to the following ones if the previous provider did not accept the request.

While the general decision-making algorithm describes the selection process as a whole, the selection strategy that determines sequence P_o is specifically responsible for finding the best providers according to some specified criteria. Thus, before proposing selection strategies to instantiate the function $ProvSelectStrategy(P)$, we describe below the selection criteria that are used by those strategies in order to meet the selection objective.

7.3 Criteria for Selection

A provider can be selected from multiple possible providers using several different criteria. For applications in which providers have different skill levels, and thus services have different qualities, an effective way to select is based on service evaluation (as described in Chapter 5). However, when interactions between autonomous agents are based on non-monetary cooperations, agents must also find providers that are more likely to accept requests, since providers are not obliged to cooperate with others or may need to limit their provision due to resource restrictions.

In particular, with free services, the motivation for service provision must be achieved through alternatives to monetary approaches, such as reciprocity. Thus, to find those providers more likely to accept requests, agents must consider their reciprocal relationships with others, because an agent is more likely to reciprocate if it receives services from others. This can also determine if the agent is likely to be influenced by others. In this context, reciprocal relationships can be analysed in terms of exchange values, since credits and debts represent the reciprocation cycle between providers and requesters. Thus, if β has provided a service to γ in a previous interaction but not to α, so that β has a credit with γ and not with α, it is more likely that γ will accept β's request, since γ is in debt with β.

Service dependence can also be used to analyse reciprocation. This is based on the premise that the relation between the services an agent needs with those others can provide determines if it is likely to be influenced by others (a similar notion is found in Castelfranchi (1990)). That is, if a requester β finds two alternative providers for its needed service, say α and γ, and β knows that α needs a service that β can provide but γ does not need any service from β, it is more likely that α will accept β's requests since there is a chance that α will need β's services in the future. Therefore, to effectively find providers that are, at the same time, more likely to perform a good quality service and more likely to accept a request, we propose a provider selection mechanism that takes into account the criteria described below.

- *Service evaluation* considers the evaluation of services received in previous interactions with each candidate provider, in order to select those providers that perform better.
- *Individual exchange values* that the requester associates with each candidate provider enable the identification of first, those more likely to accept requests, based on the reciprocation implications of credits and debts, and second, those more likely to provide *good* services, based on the satisfaction value.
- The *balance of exchange values*, of the requester in previous interactions with each candidate provider, enables the identification of those more likely to be *good* cooperation partners, in terms of the compensation of the services that are provided and reciprocated.
- The *dependence* of each candidate provider with the requester allows the identification of those partners more likely to accept requests, since dependent providers might need the requester's services in the future.

To analyse and order alternative providers according to these criteria, in order to determine which provider is more suitable for the interaction, agents use selection strategies. Either individual selection criteria or a combination of different criteria can be used to compose selection strategies for agents requesting a service. The selection criteria listed above are described in more detail in the next sections.

7.3.1 Service Evaluation

Selection based on service evaluation uses a single evaluation per provider, combining all previous evaluations of the service being requested. These previous evaluations are generated by the evaluation method proposed in Chapter 5, and the single evaluation for the provider (or service) is generated during selection, as described in Section 5.3.3.3. This single evaluation of each candidate provider α_i of service srv, represented as $Peval(\alpha_i, srv)$, is then used to rank them in order of best evaluation.

Note that there are many possible ways to determine the single evaluation $Peval(\alpha_i, srv)$ for each provider agent from all service evaluations in previous interactions $Seval(srv)$, such as selecting the *best* overall evaluation for the service, or calculating the *average* evaluation for the service. Each alternative can be viewed as a different evaluation-based selection strategy for the agent.

7.3.2 Individual Exchange Values

Selection based on exchange values takes into account the individual exchange values of a requester agent with each candidate provider as a result of previous interactions. We have proposed in Chapter 6 a computational model for exchange values, according to which an agent maintains four exchange values for each stage of the interaction. From a requester's perspective, it is relevant to consider two of these values when selecting providers: the *satisfaction value*, which may be used to select candidate providers performing a better service; and the *credit value*, since a credit with another agent suggests a disposition of the latter to reciprocate the favour in the future (that is, the latter has a debt)[2]. Although the satisfaction value corresponds to the service evaluation, which is used in the previous strategy, we refer to each of these in different ways to distinguish the evaluation-based and the exchange values-based strategies, since the latter considers subjective evaluations in addition to objective evaluations (to give credit values) while the former considers objective evaluations only (as pure service evaluations).

In short, from the requester's perspective, agents in debt are more likely to cooperate when requested to do so, and agents with which the requester has high satisfaction values in previous interactions are more likely to yield successful interactions. Information on the balance of exchange values can also be used to choose the provider more likely to provide a good service, as described in the next section.

[2]Norms and commitments, which determine obligations and penalties for participants of an interaction, can be used to ensure that a debt will be paid in the future (Conte and Falcone, 1997; Dignum, 1999; Dignum and Dignum, 2003). However, this is not discussed in the present work, since here we assume that agents decide whether to pay their debts according to their interest in maintaining a cooperation (and therefore, the penalty for not reciprocating is a possible reduction in the chances of future interactions).

7.3.3 Balance of Exchange Values

In the previous chapter, in Section 6.5, we described how agents can determine the balance of their exchange values as part of our framework for non-monetary interactions, and use the balance as a means of choosing whether to continue or avoid such interactions in the future. In this section, we describe situations in which exchange values are in equilibrium and disequilibrium for an agent β requesting a service from an agent α, and investigate the causes of such results. We analyse equilibrium and disequilibrium situations in the provision stage, and then in the reciprocation stage of the interaction.

The equilibrium situation is represented by Equations 6.1 and 6.2, for the provision and reciprocation stages respectively, in which the exchange values representing gains and losses for provider and requester are equal; if we modify these equations, the exchange values of the interacting agents are in disequilibrium. When β requests a service in the provision stage of the interaction, its balance of exchange values is determined by its satisfaction and debt values, $s_{\beta\alpha}$ and $t_{\beta\alpha}$, respectively (with satisfaction representing a gain, and debt representing a loss). Given a heterogeneous population of individuals, and possible subjective influences that may affect the requestor's debt, (discussed in Section 6.3.3), the situations in which exchange values are in equilibrium or disequilibrium, and their possible causes are presented below.

1. The balance of exchange values is in *equilibrium* for agent β when its satisfaction is equal to its debt, that is, $s_{\beta\alpha} = t_{\beta\alpha}$ in Equation 6.1. This is because there is no subjective influence on the debt accepted by β due, for example, to β exhibiting fair behaviour towards α, β and α being in the same social position, β and α having similar skills, or β and α depending on each other mutually. The situation is beneficial for β.

2. The balance of exchange values is *negative* for β when its satisfaction is less than its debt, $s_{\beta\alpha} < t_{\beta\alpha}$ in Equation 6.1. Possible causes of this disequilibrium are related to influences on β's subjective evaluation of services, as follows.

   (a) β is in a lower social position than α, so β over-valorises α by accepting a debt that is higher than its satisfaction.

   (b) β is less skilled than α and accepts a higher debt to maintain the cooperation with α, since β is aware of not being able to reciprocate at the same level.

   (c) β is not able to provide any service that α needs (a *unilateral dependence*) so valorises α more highly, resulting in a debt that is greater than its satisfaction, since a higher credit from β may motivate α to keep interacting.

   (d) α is a very high demand provider (receiving many requests), and β valorises α more highly, so it accepts a debt that is greater than its real satisfaction.

Table 7.1: Requester's Balance of Exchange Values in the Reciprocation Stage.

Balance	**β's Values**	**Causes**
equilibrium	$s'_{\beta\alpha} = v'_{\beta\alpha}$	$v'_{\beta\alpha} = t'_{\alpha\beta},\ t'_{\alpha\beta} = r'_{\alpha\beta},\ r'_{\alpha\beta} = s'_{\beta\alpha}$
negative	$s'_{\beta\alpha} < v'_{\beta\alpha}$	(a) $v'_{\beta\alpha} = t'_{\alpha\beta},\ t'_{\alpha\beta} > r'_{\alpha\beta},\ r'_{\alpha\beta} = s'_{\beta\alpha}$
		(b) $v'_{\beta\alpha} = t'_{\alpha\beta},\ t'_{\alpha\beta} = r'_{\alpha\beta},\ r'_{\alpha\beta} > s'_{\beta\alpha}$
		(c) $v'_{\beta\alpha} > t'_{\alpha\beta},\ t'_{\alpha\beta} = r'_{\alpha\beta},\ r'_{\alpha\beta} = s'_{\beta\alpha}$
positive	$s'_{\beta\alpha} > v'_{\beta\alpha}$	(d) $v_{\beta\alpha} = t_{\alpha\beta},\ t'_{\alpha\beta} = r'_{\alpha\beta},\ r'_{\alpha\beta} < s'_{\beta\alpha}$
		(e) $v'_{\beta\alpha} = t'_{\alpha\beta},\ t'_{\alpha\beta} < r'_{\alpha\beta},\ r'_{\alpha\beta} = s'_{\beta\alpha}$

(e) the service provided by α did not reach β's expected quality level (either because α has low skills or β has high quality standards), so β believes it must spend more effort on reciprocating than its satisfaction with the received service.

The impact of this situation on β's choice of alternative interaction partners is that, even though it may be necessary in some cases for β to over-valorise α's service if it is a highly-skilled or demanded provider, it is sensible for β to seek an equilibrium in its social interactions by interacting with agents providing services on the same level or those in lower demand, thus avoiding a negative balance of exchange values. The same consequence may be observed for interacting agents with different social positions or social dependence, so that when β requests a service, it may prefer interacting with providers with which it has a mutual dependence or in the same social position, instead of α with which it has a unilateral dependence or which is in a higher social position.

3. The balance of exchange values is *positive* for β when its satisfaction is greater than its acquired debt, $s_{\beta\alpha} > t_{\beta\alpha}$ in Equation 6.1. Possible causes are related to subjective influences on β's accepted debt, as given below.

(a) β exhibits unfair behaviour towards α, and thus accepts a debt that is less than its satisfaction.

(b) β is in a higher social position than α. For example, in a computer-supported scientific community in which β is an expert and α is a novice, the debt accepted by β is smaller than its satisfaction since it believes that α has done only its obligation in providing the service (which in this case could be a new algorithm for processing data).

(c) α offers a service that is common or easily found in the environment, so β admits a debt that is smaller than its satisfaction.

Although this situation is beneficial for β, since its balance of exchange values is positive due to subjective influences that decrease its debt, it causes a negative balance of exchange values for α, so α may not accept β's request in future.

Similarly, when β requests a service in reciprocation for one provided earlier (the reciprocation stage of the interaction), its balance of exchange values is determined by its satisfaction and credit values, $s'_{\beta\alpha}$ and $v'_{\beta\alpha}$, respectively. Unlike the provision stage, here the balance of the requester's exchange values does not depend on subjective influences on its own evaluation, but on the evaluations and skills of the partner agent and its willingness to reciprocate. In this case, the equilibrium and disequilibrium situations and their possible causes are presented below and summarised in Table 7.1.

1. The balance of exchange values is in *equilibrium* for β when its satisfaction with the reciprocated service is equal to the credit it is spending; that is, $s'_{\beta\alpha} = v'_{\beta\alpha}$ when all other values in Equation 6.2 are the same. Possible causes are that α reciprocates expending similar effort to the worth of its debt ($t'_{\alpha\beta} = r'_{\alpha\beta}$), and α and β have similar skills, so that the services they receive from each other compensate for the services they provide to each other. This equilibrium situation is beneficial for β.

2. The balance of exchange values is *negative* for β when its satisfaction is less than the credit it is spending; $s'_{\beta\alpha} < v'_{\beta\alpha}$ in Equation 6.2. Possible causes are as follows.

   (a) α wants to exploit β by asking more in the provision stage than it is willing to reciprocate, so α reciprocates by expending less effort than its debt (that is, $t'_{\alpha\beta} > r'_{\alpha\beta}$ in Equation 6.2).

   (b) α provides a poor quality service, or β is very demanding in its evaluation of the received service (β's satisfaction is low), so that α's effort is greater than β's satisfaction (that is, $r'_{\alpha\beta} > s'_{\beta\alpha}$ in Equation 6.2).

   (c) α acknowledges a smaller debt in reciprocation than the debt it acquired in provision, so that the credit β spends is greater than the debt α pays back (that is, $v'_{\beta\alpha} > t'_{\alpha\beta}$ in Equation 6.2).

   The impact of negative balances of exchange values in β's future interactions is that β avoids choosing α as its interaction partner, since α does not accept its full debt or reciprocates with a service of lower quality than the one β provided previously, so α is not seen as a good cooperation partner.

3. The balance of exchange values is *positive* for β when β's satisfaction is greater than the credit it is spending; $s'_{\beta\alpha} > v'_{\beta\alpha}$ in Equation 6.2. Causes are given below.

   (d) α provides a better service than β did in the provision stage, so that β's satisfaction is greater than α's effort ($r'_{\alpha\beta} < s'_{\beta\alpha}$ in Equation 6.2).

   (e) α pays its debt by expending more effort than its debt merits (that is, $t'_{\alpha\beta} < r'_{\alpha\beta}$ in Equation 6.2).

   Here, the impact of positive balances of exchange values on β's future interactions is that β tends to choose α as its interaction partner, since α has accepted its debt and reciprocated to at least the same quality as the service β provided previously.

In summary, from the requester's perspective, candidate providers with which previous interactions resulted in *equilibrium or positive balances* of exchange values are expected to yield successful interactions in future. When selecting partners based on the balance of exchange values, the requester considers overall balances of complete interactions with each candidate provider, since gains in one stage may compensate for losses in another stage, resulting in equilibrium (and successful cooperation) even though there may have been a disequilibrium in both stages. Balances in the provision and reciprocation stages (stage balances) may then be considered if the requester has not completed any interaction with a candidate provider by the time of selection.

7.3.4 Dependence

An agent β depends on another agent α when β needs a service that it is not able to perform, but α is. If α also depends on β for any needed service, β may use this *mutual dependence* to influence α when it wants to request a service. Thus, information about dependence is used by requesters with the assumption that a candidate provider is likely to collaborate only if the requester is capable of performing a service that the former needs. Agents may not always have well-defined goals and plans, so here we use the notion of dependence in its basic form, which is dependence on another agent's action (not goal), since it provides a more general approach, while other notions of dependence that require the analysis of the agents's goals and plans, such as dependence *relations* and dependence *situations* (Sichman *et al*, 1994), are not considered.

The dependence between two agents β and α in relation to a service can be of two types: *bilateral or mutual dependence* (BD), which occurs when β needs a service that α can provide and α also needs a service that β can provide; and *unilateral dependence* (UD), which occurs when β needs a service that α can provide but α does not need any service that β can provide. Candidate providers with which the requester has a bilateral dependence are therefore considered as more likely to cooperate than those with which the requester has a unilateral dependence. This is because a unilateral dependence indicates that the requester may not be able to reciprocate in the future (unless the provider's needed services or the requester's offered services change, or the requester did not have complete information about all services the provider needs when the unilateral dependence was identified).

To determine the agents with which the requester has a bilateral dependence, it needs to find out which services other agents need and if it can provide any of them. In (Sichman *et al*, 1994) agents use information about bilateral dependence in a globally accessible description, the *external description*, of all dependence relations between the agents in the society. However, such a global, centralised description is not suitable for open and dynamic systems. The alternative is for the requester agent to maintain local information about other agents that have previously requested services from it, since this suggests that they depend on the requester agent in relation to those services. Maintaining information locally is simpler than maintaining a global source of dependence information, since it requires only that agents maintain a record of other agents that have previously made a service request. Although this local description may need regular updates, this is less costly than a global update. For example, agents can update their dependence information according to the frequency with which requests are received.

7.4 Strategies

The selection criteria described above can now be used in different selection *strategies* to determine which provider is more suitable for an interaction, by analysing and ordering possible providers. Although there are many different ways of using each selection criterion, potentially resulting in a large number of possible strategies, the aim here is to compare the different criteria, instead of analysing an extensive list using the same criterion. Thus, we propose several strategies for selecting providers, based on the criteria described in Section 7.3, as follows:

1. *evaluation-based selection*, which considers the evaluation of services in previous interactions with each possible provider;
2. *exchange values-based selection*, which considers the individual exchange values acquired in previous interactions with each possible provider, and their balances;
3. *dependence-based selection*, which considers the dependence between each possible provider and the requester; and
4. *combined selection*, which considers a combination of service evaluation, exchange values and dependence.

These strategies instantiate the function *ProvSelectStrategy*(P) in the general decision-making algorithm, Algorithm 10, and are described in more detail below.

7.4.1 Evaluation-based Selection

To select providers according to the evaluation of their services ($Peval(\alpha_i, srv)$), the set of possible providers is ordered with higher values of $Peval(\alpha_i, srv)$ first. To calculate $Peval(\alpha_i, srv)$ for each possible provider based on previous evaluations, we consider a simple evaluation approach, although others can be used. We take the best *overall* evaluation for the service being requested. To determine the best overall evaluation for a service srv of each candidate provider α_i in P, the requester agent (β) must find, for each α_i, the maximum evaluation of srv ($Seval_{max}(srv)$) from all previous interactions.

Algorithm 11 Algorithm for the refusal condition $Devalued(\alpha)$.

```
1: input: $\alpha$
2: $condition = false$
3: for all $it_j \in IT$ do
4:     $(r_{\beta\alpha}, s_{\beta\alpha}, t_{\beta\alpha}, v_{\beta\alpha}) = GetValues(it_j, \alpha, \text{“}all\text{”}, 1)$
5:     $(r'_{\beta\alpha}, s'_{\beta\alpha}, t'_{\beta\alpha}, v'_{\beta\alpha}) = GetValues(it_j, \alpha, \text{“}all\text{”}, 2)$
6:     $condition = (r_{\beta\alpha} = 0 \wedge\ s_{\beta\alpha} = 0 \wedge\ t_{\beta\alpha} = 0 \wedge\ v_{\beta\alpha} = 0 \wedge\ r'_{\beta\alpha} = 0 \wedge\ s'_{\beta\alpha} = 0 \wedge\ t'_{\beta\alpha} = 0 \wedge\ v'_{\beta\alpha} = 0)$
7: end for
8: output: $condition$
```

Thus, the single evaluation for a candidate provider α_i regarding a service srv is given by:

$$Peval(\alpha_i, srv) = Seval_{max}(srv) \mid \forall\ Seval_{max}(srv), Seval_n(srv) \in PSE_{srv,\alpha_i} \cdot Seval_n(srv) \leq Seval_{max}(srv)$$

where PSE_{srv,α_i} is the set of all previous evaluations of service srv provided by α_i, determined by β as in Equation 5.1. With this information, the sequence of candidate providers P_o is ordered as follows:

$$P_o = \{\langle \alpha_1, ..., \alpha_n \rangle : P| \ (Peval(\alpha_i, srv)\ >\ Peval(\alpha_{i+1}, srv)) \Rightarrow \alpha_i > \alpha_{i+1}\}$$

Thus, P_o is ordered with providers with higher evaluations first.

7.4.2 Exchange Values-based Selection

We propose two selection strategies based on exchange values. The first strategy, which we call *simple reciprocation*, takes into account the credits of the requester, so as to consider reciprocation and find providers that are more likely to cooperate. The second strategy, which we call *analysing cooperative situations*, in addition to the credits of the requester, takes into account the balance of exchange values in the requester's previous interactions, so as to be more restrictive in relation to the quality of interactions and avoid those that may result in unsuccessful outcomes.

Here, if a requester with credit has its request refused, it devalues the provider since the latter did not fulfill its commitment. Such providers are not considered good cooperation partners so, for future selection, the exchange values-based strategies identify, in set P, those candidate providers that were devalued in previous interactions (represented by the function $Devalued(\alpha_i)$, for $\alpha_i \in P$), and place them last in sequence P_o (represented by function $Last(P_o, \alpha_i)$).

Devaluations are identified as described in Section 6.4.5, and the function $Devalued(\alpha_i)$ is defined as in Algorithm 11. Here, IT is the set of all previous interactions of agent β,

Algorithm 12 Algorithm for provider selection through Simple Reciprocation.

```
1: input: $P$
2: $P_o = \{\langle \alpha_1, ..., \alpha_n \rangle : P | \ (totalv_{\beta\alpha_i} > totalv_{\beta\alpha_{i+1}}) \Rightarrow \ \alpha_i > \alpha_{i+1}\}$
3: for $\alpha_i \in P_o$ do
4:     if $Devalued(\alpha_i)$ then
5:         $P_o = Last(P_o, \alpha_i)$
6:     end if
7: end for
8: output: $P_o$
```

and the function $GetValues(it_j, \alpha_i, \text{"}all\text{"}, \{1, 2\})$[3] retrieves, from β's history of exchange values, *all* exchange values acquired by β in previous interaction it_j with α_i, and which are associated with stages 1 and 2 of the interaction.

7.4.2.1 Simple Reciprocation

For a generic requester β, the provider selection strategy based on *simple reciprocation* aims at finding more likely cooperations by analysing β's credit value. This is specified in Algorithm 12, in which a set of possible providers (P) is provided as input, returning an ordered sequence of the elements of P (P_o) to be used in the selection process specified in Algorithm 10.

The selection starts by ordering P with those providers with which β has higher credits first (since providers with debts are more likely to cooperate), resulting in the ordered sequence P_o. Since agents may accumulate credits with others over time, we consider here the total credit of the requester with each candidate provider, represented by $totalv_{\beta\alpha_i}$. Then, the candidate providers that did not pay debts in previous interactions are placed last in sequence P_o. Finally, P_o is ordered so that earlier candidate providers in the sequence are considered to be more likely to accept β's request based on the analysis of reciprocal relationships.

7.4.2.2 Analysing Cooperative Situations

For a generic requester β, the provider selection strategy based on *analysing cooperative situations* aims at finding more likely cooperations but also interactions with a potentially better outcome in terms of the balance of exchange values. This is done by analysing, in addition to β's credit value, β's balance of exchange values in previous interactions with each candidate provider, as shown in Algorithm 13.

[3]This function retrieves either all exchange values or specific exchange values (e.g., credit or satisfaction) resulting from a particular interaction, with a particular partner, and regarding a particular stage (where 1 indicates the provision stage, and 2 the reciprocation stage).

Algorithm 13 Algorithm for provider selection through *Analysing Cooperative Situations.*

```
1: input: $P$
2: $P_o = \{\langle \alpha_1, ..., \alpha_n \rangle : P | \ (totalv_{\beta\alpha_i} > totalv_{\beta\alpha_{i+1}}) \Rightarrow \alpha_i > \alpha_{i+1}\}$
3: $Sort(P_o, oab^n, stb^n)$
4: for $\alpha_i \in P_o$ do
5:     if $Devalued(\alpha_i)$ then
6:         $P_o = Last(P_o, \alpha_i)$
7:     end if
8: end for
9: output: $P_o$
```

According to this strategy, requesters first order P with providers with which β has higher credits earlier, resulting in sequence P_o. Then, requesters analyse two consecutive providers in P_o, and if requesters have credit with both providers, or if requesters have no credit with both providers (note that here the comparison of credit values is qualitative instead of quantitative), they identify interactions with a potentially better outcome in terms of the balance of exchange values. To achieve this, β compares its balance of exchange values in previous interactions with the providers, first the overall balance of exchange values and, if they are the same, the stage balances. The provider with fewer negative balances of exchange values is preferred. To perform the comparison, β determines the proportion of negative overall balances associated with two consecutive providers in P_o, α_i and α_{i+i}, which we call oab^n (and we call stb^n the proportion of negative stage balances), until P_o is completely ordered. This is represented in Algorithm 13 by the function $Sort(P_o, oab^n, stb^n)$. Finally, the candidate providers that did not pay debts in previous interactions are placed last in sequence P_o, as in the previous strategy.

Ultimately, P_o is ordered such that candidate providers more likely to reciprocate and with fewer unsuccessful interactions are earlier, and candidate providers less likely to reciprocate and with more unsuccessful interactions are later.

7.4.3 Dependence-based Selection

The dependence-based strategy is based on the assumption that a generic provider α is more likely to collaborate with a requester β if β is capable of performing a service that α needs. We use the notation D_{bd} to represent the set containing all agents with which β has a bilateral dependence; that is, those agents that have previously made a service request. Thus, when β needs to request a service from others, it should give preference to those with which it identifies a dependence, represented by the set D_{bd}. Thus, the sequence of candidate providers P_o is ordered as follows:

$$P_o = \{\langle \alpha_1, ..., \alpha_n \rangle : P | \ (\alpha_i \in D_{bd} \wedge \alpha_{i+1} \notin D_{bd}) \Rightarrow \alpha_i > \alpha_{i+1}\}$$

Algorithm 14 Algorithm for provider selection with combined criteria.

```
1: input: $P$
2: $P_{suc} = \{\langle \alpha_1, ..., \alpha_n \rangle : P| \ ((Peval(\alpha_i) > Peval(\alpha_{i+1})) \vee (Peval(\alpha_i) = Peval(\alpha_{i+1}) \wedge oab^n_{\alpha_i} < oab^n_{\alpha_{i+1}})) \Rightarrow \alpha_i > \alpha_{i+1}\}$
3: $P_{cop} = \{\langle \alpha_1, ..., \alpha_n \rangle : P| \ ((totalv_{\beta\alpha_i} > 0 \wedge totalv_{\beta\alpha_{i+1}} = 0) \vee (totalv_{\beta\alpha_i}, totalv_{\beta\alpha_{i+1}} = 0 \wedge \alpha_i \in D_{bd} \wedge \alpha_{i+1} \notin D_{bd})) \Rightarrow \alpha_i > \alpha_{i+1}\}$
4: for $\alpha_i \in P$ do
5:     $S[\alpha_i] = Score(P_{cop}, P_{suc})$
6: end for
7: $P_o = \{\langle \alpha_1, ..., \alpha_n \rangle : P| \ (S[\alpha_i] > S[\alpha_{i+1}]) \Rightarrow \alpha_i > \alpha_{i+1}\}$
8: output: $P_o$
```

where P_o is ordered with dependent providers first.

7.4.4 Combined Strategy

Up to this point, we have presented strategies that use service evaluation, dependence, or exchange values individually to select providers. However, these criteria are complementary, and thus can be combined in a mixed strategy. By complementary we mean that, to find candidate providers more likely to accept requests, information about the requester's credit value and dependence in relation to each provider can be combined, so that when the requester does not have any credit with the candidate providers it can use information about dependence. Similarly, to find interactions with a potentially better outcome in terms of the quality of the received service and balance of gains and losses, information about service evaluation and balance of exchange values can be combined. This information is complementary, since service evaluation is an objective measure for service quality, and the balance of exchange values is relative to both objective and subjective values.

To combine these different criteria in order to determine which providers are at the same time more likely to cooperate and more likely to yield a successful interaction, the combined strategy is defined such that providers are ranked according to each separately, and then both results are aggregated in a single ranking of preferred providers[4]. This process is detailed in Algorithm 14, in which providers more likely to cooperate are ordered in sequence P_{cop}, providers more likely to yield a successful interaction are ordered in sequence P_{suc}, and the final preferred providers are ordered in sequence P_o.

The sequence P_{suc} is ordered with providers with higher service evaluations first (that is, $Peval(\alpha_i) > Peval(\alpha_{i+1})$). If evaluations from two consecutive candidate providers

[4]This technique is similar to the Borda count voting protocol, which is used to choose an agent from a group of candidates for performing some task, based on the preferences of a distinct group of agents (the voters). To determine the winner candidate based on the majority of votes, the protocol consists of assigning points to candidates in each voter's preference list according to their position in the list, and summing these points across voters to find the candidate with the highest points total (Sandholm, 1999).

TABLE 7.2: Scoring Candidate Providers According to Ordering Criteria for P_{suc}.

α_i	$Peval(\alpha_i)$	$oab^n_{\alpha_i}$	**Score**
α_1	0.8		4
α_2	0.6		3
α_3	0	0	2
α_4	0	50	1

TABLE 7.3: Scoring Candidate Providers According to Ordering Criteria for P_{cop}.

α_i	$total_{v_{\beta\alpha_i}}>0$	$\alpha_i \in D_{bd}$	**Score**
α_1	$true$	$true$	4
α_2	$true$	$false$	3
α_4	$false$	$true$	2
α_3	$false$	$false$	1

in P_{suc} are the same, preference is given to the one with fewer negative overall balances of exchange values in previous interactions ($oab^n_{\alpha_i} < oab^n_{\alpha_{i+1}}$). Service evaluation and the proportion of negative overall balances are considered here as *quantitative* criteria, and a service evaluation $Peval(\alpha_i)$ for each candidate provider α_i is calculated as in Section 7.4.1. The sequence P_{cop} is ordered with providers with which the requester has credit first (with $totalv_{\beta\alpha_i} > 0$). If the credits from two consecutive candidate providers in P_{cop} are zero, preference is given to the one with which the requester has a mutual dependence (that is, $\alpha_i \in D_{bd}$). Credits and dependence are considered here as *qualitative* criteria, in the sense that quantities are not taken into account, but only whether a credit is greater than or equal to zero, and a dependence is true or false. This improves the combination of the two sequences, since we assume that having a credit with a provider (regardless of the quantity) is already an indication that this provider is more likely to reciprocate, so when the two rankings (P_{suc} and P_{cop}) are combined and the requester has credit with two different providers, the one which is earlier in P_{suc} (and thus more likely to provide a good service) is preferred.

After sequences P_{cop} and P_{suc} are ordered, a score is assigned to each candidate provider according to each set's ordering criteria. The highest score is the number of candidate providers in P ($|P|$). Candidate providers in P_{suc} receive scores according to their *position* in the sequence (since the ordering criteria for this set are quantitative), with scores decrease by 1 as agents get further from the first position in the ordered sequence, as shown in Table 7.2. Candidate providers in P_{cop} receive scores depending on whether they *satisfy* each ordering criteria, such that the highest score $|P|$ is assigned to all agents that satisfy both $total_{v_{\beta\alpha_i}} > 0$ and $\alpha_i \in D_{bd}$, the score $|P| - 1$ is assigned to all agents that satisfy only $total_{v_{\beta\alpha_i}} > 0$, the score $|P| - 2$ is assigned to all agents that satisfy only $\alpha_i \in D_{bd}$, and the score $|P| - 3$ is assigned to all agents that do not satisfy both criteria. Note that the score for agents that satisfy only $total_{v_{\beta\alpha_i}} > 0$ is higher than the score for agents that satisfy only $\alpha_i \in D_{bd}$, since the former criterion is applied first when ordering P_{cop} (it has more importance in the ordering process). An example of scores for the elements of P_{cop} when $|P| = 4$ is shown in Table 7.3.

The final sequence P_o of preferred providers is ordered according to the total score of each candidate provider α_i in both P_{cop} and P_{suc} (represented in Algorithm 14 as $S[\alpha_i]$). For example, consider four providers $P = \{\alpha_1, .., \alpha_4\}$ that are ordered in P_{cop} and P_{suc} as in Tables 7.3 and 7.2. The score for α_1 is 8 $(4 + 4)$, for α_2 is 6 $(3 + 3)$, for α_3 is 3 $(2 + 1)$, and for α_4 is 3 $(1 + 2)$, which results in the sequence of preferred providers $P_o = \langle \alpha_1, \alpha_2, \{\alpha_3, \alpha_4\} \rangle$. If two providers receive the same score, the requester may choose the one that has higher score either on sequence P_{cop}, if the requester is more concerned with the time taken to find an available provider, or on sequence P_{suc}, if it is more concerned with the success of the interaction. In the end, the ordered sequence P_o represents providers that are, at the same time, more likely to provide good services and to accept requests.

7.5 Conclusion

In this chapter, we have presented a provider selection mechanism for service requesters operating in dynamic cooperative applications. The mechanism addresses two problems: first of selecting a service provider that performs a good quality service from among alternatives providing similar services; and second of finding providers more likely to accept requests so that requests do not take a long time to be accepted. This is achieved by considering, as selection criteria, information on both previous evaluations of services and reciprocal relationships with other agents, through service dependence, service evaluation, and exchange values.

From these selection criteria, exchange values provide the only criterion that allows the analysis of both service quality (through the satisfaction value and the balances of exchange values) and reciprocal relationships (through the credit value).

Five provider selection strategies that combine different criteria have been proposed. Some focus on one aspect of the providers, while others try to balance selection according to both aspects: service quality and reciprocal relationships. Based on the strategies' design properties, we expect that the latter strategies, represented here by the exchange values-based *analysing cooperative situations* and *combined* strategies, can improve the performance of agents in the task of finding providers for the services that they need, when they operate in open cooperative systems in which providers are resource constrained, and services are free and have different levels of quality.

Chapter 8

Requester Selection Mechanism

8.1 Introduction

While service requesters need to select providers that are more likely to reciprocate and provide good quality services, as discussed in the previous chapter, service providers must also select partners, but from a different perspective. In particular, autonomous agents wanting to engage in interactions that bring benefits in terms of expected future interactions must be able to decide whether to accept other agents requests. This is important for the kind of cooperative applications we focus on here, in which interactions are based on reciprocity rather than benevolent behaviour or monetary compensation.

Requester selection is also desirable to allow providers to sensibly manage their resources so as not to be overwhelmed with services to perform for other agents. The need to sensibly managing resources is more evident when providers have a significant number of requests to attend to, and when the services provided consume a large amount of processing power and time, as described by Foster (2005). This is the case of applications in the bioinformatics domain, in which the increasing number of requests that automated experiments can generate, and the large amount of computation that service requests may need (since they usually deal with large amounts of data), place a heavy load on service providers, which can limit the number of requests that can be processed.

Since agents have bounded resources and are not obliged to accept requests, therefore, determining when and why agents should provide services to others is complex. In non-monetary cooperative applications, when agents cannot rely on the benevolence of others to get services they need, they must provide services to others in order to be able to receive services. Thus, agents might be motivated to provide services to others for two reasons: to reciprocate for a service received and to maintain a cooperation with a partner agent on the one hand, or to improve the chances of receiving a service in return in the future on the other. In consequence, providers must consider their cooperative relationships with others, and their available resource capacity, in order to decide whether to accept requests.

In this context, we propose a requester selection mechanism for resource-bounded agents providing services in cooperative non-monetary applications, to enable them to efficiently select requesters, as follows.

- Providers consider the *incentives for service provision* when selecting requesters, so that these incentives can be used as future benefits, such as receiving services in future interactions;
- Providers *analyse incoming requests* so that they can decide whether to cooperate based on existing relationships and the results of previous interactions with other agents, and limit service provision when resources are scarce.

Here, the incentive for service provision comes from the expectations of future interactions, while the decision of whether to accept requests is achieved through *requester selection strategies* and *selection criteria* to analyse existing and potential cooperative relationships, in the context of resource limitations. Establishing and maintaining cooperative relationships provides stable links for an agent to draw upon when it needs a service, except when the cooperation is not beneficial. For example, if an agent provides a good quality service, maintaining a cooperation with a partner providing a poor service is not beneficial since its effort in providing a good service is not compensated by receiving a poor one.

The chapter starts with a description of the selection criteria used by the requester selection mechanism in Section 8.2. A formal model for the provider's decision-making and selection strategies are presented in Section 8.3. The chapter finishes with conclusions in Section 8.4.

8.2 Selecting Among Service Requests

As discussed above, agents operating in open cooperative systems with bounded resources require some means of selecting among received requests in order to satisfy those most likely to contribute to lasting and valuable cooperative relationships.

A simple way to analyse incoming requests is to determine if the requester can provide any service needed by the provider, so that not only does the provider help the requester by performing a service the latter is not able to execute, but the requester can also be a provider in the future. Here, the *incentive* to provide a service is a dependence with the requester, which provides some expectation of future reciprocation. This notion of *dependence of service* is the same as that introduced in Chapter 7 for provider selection, but is now considered from the provider's perspective.

Although selecting requesters based on dependence helps to form cooperations in which agents can provide services to each other, it has the disadvantage that if needed and provided services change over time due to the dynamic characteristics of the environment or the agents themselves, established cooperations may be affected (as shown in (Rodrigues and Luck, 2006a)). However, even if a dependence between provider and requester is identified, the provider also needs to consider whether the cooperation is worthwhile. For example, it must assess whether the effort in reciprocating a service compensates for the satisfaction with the service received in a previous interaction. Service dependence alone does not allow such an analysis.

An alternative way of analysing incoming requests as potential cooperations is based on the provider's *exchange values*, following the model for non-monetary interactions proposed in Chapter 6. Here, the incentive to provide a service is to maintain a cooperation with the requester by paying a debt, or to improve the chances of future interactions by gaining a credit with the requester (which can be spent in the future in return for another service). When exchange values are considered for requester selection, the changes in an agent's needed and provided services do not affect existing cooperations between agents since the links between requesters and providers are their corresponding credits and debts, as shown in (Rodrigues and Luck, 2006b).

Agents can derive information about cooperations and their quality by analysing their exchange values. For example, a provider can consider its *satisfaction value* and *balance of exchange values* from previous interactions with an agent, since these indicate the likelihood of providing a good service. It can also consider its *debt value*, since this indicates a commitment (assuming cooperative behaviour) to reciprocate. Using these values, a provider can filter requests by accepting only those from agents with which it has higher satisfaction or fewer negative balances of exchange values, or those with which it has a debt.

In summary, we consider two criteria for requester selection that allow providers to find reciprocal relationships with requesters: dependence and exchange values. These criteria are used both individually and combined to select requesters and to limit service provision when resources are scarce, by means of *selection strategies*.

8.3 Formal Model

To allow agents to select requesters, we must specify both the decision-making process through which selection occurs, and the means for selection, through selection strategies. Below, we describe a general decision-making process for selecting requesters, and alternative selection strategies, each considering different selection criteria.

8.3.1 General Decision-making

We take α to be an agent that has received a request from another agent β or from a group of agents $\beta_1, .., \beta_i \in Q$. Q_o is a sequence of the elements of Q ordered based on a particular criterion, *current_allocation* is the number of requests being attended to at the same time by α, *maximum_allocation* is the maximum number of requests that α can accommodate at the same time, *allocation_thresold* is the value over which α starts restricting the acceptance of requests, and *available_capacity* is the number of further requests that α can accommodate.

To consider both resource constraints and reciprocity during requester selection, the decision process comprises two parts. First, an agent must decide if a request should be considered for acceptance or discarded. This is achieved by analysing what we call *refusal conditions*, which represent situations that, if true, indicate that a requester is a poor cooperation partner and thus its request should be discarded. Requests that do not satisfy any refusal conditions are those the provider considers for acceptance.

Second, from these requests, the provider must select which to accept based on one or more selection criteria and on its available resources. When there is only an individual request, the selection criteria are used as binary conditions (either true or false) to decide whether to accept the request, while when there is a group of requests, the selection criteria are used to order them so that, in case of scarce resources, those requests that are preferred over others are accepted first.

The general decision-making process for selecting requesters is shown in Algorithm 15. Here, when the provider α reaches its maximum capacity, it always refuses the request. If there are available resources but the requester satisfies any refusal condition $rc_j \in RC$, where $rc_j \in \{true, false\}$ and RC is the set of all refusal conditions, the request is denied. Now, if there is only one requester to analyse (that is, $|Q| = 1$), selection depends on the provider's current allocation of resources: if it is below a certain threshold (which indicates that resources are scarce), the request is always accepted; if it is above this threshold and the provider needs to restrict the acceptance of requests, further selection criteria, which we refer to as *individual selection criteria*, are applied to consider a requester.

If there is a group of requesters to be analysed, say agents $\beta_1, .., \beta_i \in Q$, they are subjected to an ordering procedure, represented by function *GroupSelection*(Q) in Algorithm 15, which results in the ordered set Q_o, and takes into account *group selection criteria*. Then, requests are accepted in order of preference until the provider uses all available capacity, and if there are any remaining requests, they are refused. Accepted

Algorithm 15 General requester selection.

```
1: input: $Q$
2: if $maximum\_allocation$ then
3:    Refuse request
4: else
5:    for all $\beta_i \in Q$ do
6:       if $\exists\ rc_j \in RC \wedge rc_j = true$ then
7:          Refuse request
8:          $Q = Q \backslash \{\beta_i\}$
9:       end if
10:   end for
11:   if $|Q| = 1$ then
12:      if $current\_allocation < allocation\_threshold$ then
13:         Accept request
14:      else if $\beta$ satisfies $individual\ selection\ criteria$ then
15:         Accept request
16:      else
17:         Refuse request
18:      end if
19:   else if $|Q| > 1$ then
20:      $Q_o = GroupSelection(Q)$
21:      for all $\beta_i \in Q_o$ do
22:         if $available\_capacity > 0$ then
23:            Accept request
24:            $available\_capacity = available\_capacity - 1$
25:            $Q_a = Q_a \cup \{\beta_i\}$
26:         else
27:            Refuse request
28:         end if
29:      end for
30:   end if
31: end if
32: output: $Q_a$
```

requests are stored in the set Q_a, representing the requests that the agent needs to perform.

The choice of refusal conditions, individual selection criteria and group selection criteria (through the instantiation of function *GroupSelection*(Q)) depends on the agent's *selection strategy*. Alternative selection strategies that use the selection criteria described in Section 8.2 are presented next. We first describe selection strategies that use a single selection criterion — dependence or exchange values — and then strategies that use a combination of those criteria.

8.3.2 Dependence-based Strategy

Using the dependence-based strategy, requesters on which the provider depends are preferred. To reason about dependence, the provider needs information about which agents can provide the services it may need in the future. It does this by maintaining a set, D_{on}, which contains the agents that the provider depends on for any needed service. To determine this set, agents must add to the set D_{on} all possible providers for services they have requested (even if they have not received services from those providers), since they depend on those providers. For example, if in a previous interaction in which agent α requested service s_1, the possible providers were β_1, β_2, and β_3, then α depends on these agents to perform service s_1, resulting in $D_{on} = \{\beta_1, \beta_2, \beta_3\}$.

For the dependence-based strategy, Algorithm 15 is instantiated as follows.

- There is no refusal criterion, so $RC = \emptyset$, since no further information (such as the quality of received services or whether there was compensation of provided and received services) can be derived.
- With an *individual request*, the provider always cooperates if it has available resources but, when resources are scarce, it limits service provision to those requesters on which it depends. Thus, the *individual selection criterion* is $\beta \in D_{on}$.
- When analysing a *group of requests*, the ordered sequence Q_o is determined as:

  $$Q_o = \{\langle \beta_1, ..., \beta_n \rangle : Q | \ (\beta_i \in D_{on} \wedge \beta_{i+1} \notin D_{on}) \Rightarrow \beta_i > \beta_{i+1}\}$$

  where sequence Q_o is ordered with agents in D_{on} first.

8.3.3 Exchange Values-based Strategies

We propose two requester selection strategies based on exchange values. The first strategy, *simple reciprocation*, takes into account the debts of the provider with each requester, to find reciprocal relationships. According to this strategy, requesters with which the provider has higher debts are preferred.

In addition to the debts of the provider, the second strategy, *analysing cooperative situations*, takes into account the balance of exchange values in previous interactions with the requester, to find partners that are likely to yield successful interactions. To develop such a strategy we first analyse the situations in which the provider's balance of exchange values is in equilibrium, negative or positive, and possible causes of these situations, in the next section.

TABLE 8.1: Provider's Balance of Exchange Values in the Provision Stage.

Balance	**α's Values**	**Causes**
equilibrium	$r_{\alpha\beta} = v_{\alpha\beta}$	$r_{\alpha\beta} = s_{\beta\alpha}, s_{\beta\alpha} = t_{\beta\alpha}, t_{\beta\alpha} = v_{\alpha\beta}$
negative	$r_{\alpha\beta} > v_{\alpha\beta}$	(a) $r_{\alpha\beta} > s_{\beta\alpha}, s_{\beta\alpha} = t_{\beta\alpha}, t_{\beta\alpha} = v_{\alpha\beta}$
		(b) $r_{\alpha\beta} = s_{\beta\alpha}, s_{\beta\alpha} > t_{\beta\alpha}, t_{\beta\alpha} = v_{\alpha\beta}$
		(c) $r_{\alpha\beta} = s_{\beta\alpha}, s_{\beta\alpha} = t_{\beta\alpha}, t_{\beta\alpha} > v_{\alpha\beta}$
positive	$r_{\alpha\beta} < v_{\alpha\beta}$	(d) $r_{\alpha\beta} < s_{\beta\alpha}, s_{\beta\alpha} = t_{\beta\alpha}, t_{\beta\alpha} = v_{\alpha\beta}$
		(e) $r_{\alpha\beta} = s_{\beta\alpha}, s_{\beta\alpha} < t_{\beta\alpha}, t_{\beta\alpha} = v_{\alpha\beta}$
		(f) $r_{\alpha\beta} = s_{\beta\alpha}, s_{\beta\alpha} = t_{\beta\alpha}, t_{\beta\alpha} < v_{\alpha\beta}$

8.3.3.1 Balance of Exchange values from the Provider's Perspective

In the previous chapter, we analysed the situations in which the balance of exchange values is in equilibrium, positive, or negative for an agent requesting a service. Here, we present a similar analysis, but from the perspective of the agent *providing* the service. The purpose of this analysis is to examine the possible causes of equilibrium and disequilibrium of the provider's exchange values, so that it is clear why the balance of exchange values can be used to find good cooperation partners, and to filter incoming requests.

We start by considering equilibrium and disequilibrium situations in the provision stage, then in the reciprocation stage of the interaction. When an agent α is providing a service to another agent β in the provision stage of the interaction, α's balance of exchange values is determined by its renouncement and credit values, $r_{\alpha\beta}$ and $v_{\alpha\beta}$, respectively. Here, α's balance of exchange values generally depends on its skills in performing the service and on the valorisation received from the partner agent, β. However, α's balance may be affected by subjective influences on both β's valorisation of α's service, and the credit taken by α (as described in Section 6.3.3).

When the exchange values representing gains and losses for provider and requester are equal, the interaction is said to be in equilibrium. This situation is represented by Equations 6.1 and 6.2, for the provision and reciprocation stages respectively. If the relations in those equations are modified, the exchange values of the interacting agents are in disequilibrium. In this context, the situations in which α's exchange values are in equilibrium and disequilibrium and their possible causes are presented below, and summarised in Table 8.1.

- The balance of exchange values is in *equilibrium* for α when its renouncement is equal to the credit it gains, that is, $r_{\alpha\beta} = v_{\alpha\beta}$ when all other values in Equation 6.1 are the same. This is because: α and β have the same perspective over service evaluation (that is, $r_{\alpha\beta} = s_{\beta\alpha}$); there is no subjective influence on the debt accepted by β (that is, $s_{\beta\alpha} = t_{\beta\alpha}$); and there is no subjective influence on the credit taken by α (that is, $t_{\beta\alpha} = v_{\alpha\beta}$).

- The balance of exchange values is *negative* for α when its renouncement is greater than the credit it gains; that is, $r_{\alpha\beta} > v_{\alpha\beta}$. Possible modifications to Equation 6.1 that cause this are as follows.

  (a) α provides a poor quality service or β is very demanding in its evaluation of the received service (so β's satisfaction is low), and α's effort is greater than β's satisfaction, $r_{\alpha\beta} > s_{\beta\alpha}$.

  (b) α is under-valorised by β, which accepts a smaller debt than its satisfaction ($s_{\beta\alpha} > t_{\beta\alpha}$) due, for example, to β exhibiting unfair behaviour towards α, β being in a higher social hierarchy than α, or α offering a service that is common or easily found in the environment.

  (c) α's credit is saturated, so that $v_{\beta\alpha} < t_{\beta\alpha}$, since α has been valorised by β many times, causing large amounts of credit with β to accumulate. α has not had the chance to use these credits, so that every new valorisation that α receives from β in this context is seen as having less value for α.

- The balance of exchange values is *positive* for α when its renouncement is less than the credit it gains; that is, $r_{\alpha\beta} < v_{\alpha\beta}$. Possible reasons for this are as follows.

  (d) α is over-skilled and can perform the service with little effort, or β is not very demanding in its evaluation of the received service, so that α's effort is less than β's satisfaction, $r_{\alpha\beta} < s_{\beta\alpha}$.

  (e) α is over-valorised by β, which accepts a debt greater than its satisfaction ($s_{\beta\alpha} < t_{\beta\alpha}$) to motivate α to keep interacting. This might happen because β is in a lower social position than α, β is less skilled than α, β is not able to provide any service that α needs with its current set of capabilities (a *unilateral dependence*), or α is in very high demand (receiving many requests).

  (f) α takes a credit that is greater than β's debt, $t_{\beta\alpha} < v_{\beta\alpha}$. Although we do not identify any possible cause for this situation, it is mentioned here since it is a possible modification in Equation 6.1.

When α reciprocates, its balance of exchange values is determined by its renouncement and debt values, $r'_{\alpha\beta}$ and $t'_{\alpha\beta}$, respectively, and generally depends on the skills required to reciprocate. In this case, the situations and their possible causes are as described below.

- The balance of exchange values is in *equilibrium* for α when its renouncement is equal to the debt it is paying; that is, $r'_{\alpha\beta} = t'_{\alpha\beta}$ in Equation 6.2. This is because α reciprocates, expending similar effort to the worth of its debt.

- The balance of exchange values is *negative* for α when its renouncement is greater than the debt it is paying; $r'_{\alpha\beta} > t'_{\alpha\beta}$ in Equation 6.2. A possible cause is that α

reciprocates with a service of better quality than the one β provided previously, so α pays its debt by expending more effort than its debt merits.

- The balance of exchange values is *positive* for α when its renouncement is less than the debt it is paying; $r'_{\alpha\beta} < t'_{\alpha\beta}$ in Equation 6.2. A possible cause here is that α wants to exploit β by asking more in the provision stage than it is willing to return in the reciprocation stage, and thus α reciprocates by expending less effort than its debt.

In summary, based on the outcome of interactions in terms of the balance of exchange values, a provider can decide whether it should continue with the interaction (in cases of benefit and equilibrium), drop the interaction (in cases of loss), or review its actions in an attempt to improve its valorisation and adjust its evaluation standards if it is being too strict (or lenient) with the evaluation of services it receives (in cases of loss or under-evaluation). For example, in cases in which α is under-valorised by β, α can either continue cooperating with β and improve the quality of its service to get a better evaluation, or cease its cooperation with β if the latter is not being fair in its valorisation. Also, α tends to maintain the cooperation with β if it does not have to expend more effort providing the service than the debt it acquired previously.

8.3.3.2 Simple Reciprocation

The simple reciprocation strategy instantiates the general decision-making in Algorithm 15 as below.

- The provider α does not cooperate with a requester β_i if the latter did not reciprocate services in previous interactions, and thus it is unlikely to reciprocate in the future. Thus, the *refusal condition* for this strategy is defined as:

$$RC = \{Devalued(\beta_i)\}$$

  The function, $Devalued(\beta_i)$, identifies agents that were devalued for not fulfilling their debts, and is defined as in Section 7.4.2, Algorithm 11.

- When analysing an *individual request*, acceptance is restricted to those requesters with which α has a debt. Since agents may accumulate debts with others over time, we consider here the total debt that α has with each requester β_i, represented by $totalt_{\alpha\beta_i}$. Thus, the *individual selection criterion* is $totalt_{\alpha\beta_i} > 0$.

- For a *group of requests*, Q_o is ordered with those requesters with which α has higher debts first, indicating that the provider is more likely to reciprocate when it has higher debts. Thus, Q_o is determined as:

$$Q_o = \{\langle\beta_1, ..., \beta_n\rangle : Q|\ (totalt_{\alpha\beta_i} > totalt_{\alpha\beta_{i+1}}) \Rightarrow \beta_i > \beta_{i+1}\}$$

Algorithm 16 Algorithm for the refusal condition $Low\_Valorisation(srv, \beta)$.

```
1: input: $srv, \beta$
2: $condition = false$
3: $average\_valorisation = GetAverageValorisation(srv)$
4: for all $it_j \in IT$ do
5:    $v_{\alpha\beta} = GetValues(it_j, \beta, \text{“}credit\text{”}, 1)$
6:    if $v_{\alpha\beta} < average\_valorisation$ then
7:       $condition = true$
8:    end if
9: end for
10: output: $condition$
```

Algorithm 17 Algorithm for the refusal condition $Only\_Provider(\beta)$.

```
1: input: $\beta$
2: $condition = false$
3: for all $srv_j \in Need$ do
4:    $P = FindProviders(srv_j)$
5:    if $\beta \subset P \wedge |P| = 1$ then
6:       $condition = true$
7:    end if
8: end for
9: output: $condition$
```

8.3.3.3 Analysing Cooperative Situations

The *analysing cooperative situations* strategy selects according to the balance of exchange values of the provider, in addition to its debts, in previous interactions with each requester. To avoid interacting with requesters with which the provider had negative balances of exchange values in previous interactions, this strategy uses the causes of unsuccessful interactions described in Section 8.3.3.1 as refusal conditions to discard requests. Thus, there are three cases in which the provider does not cooperate even with available resources: when the requester did not reciprocate in previous interactions (that is, if it was devalued); when α received a low valorisation from the requester in a previous interaction; and when the requester provided in a previous interaction with lower quality than the service α reciprocated (that is, the situation in which $r'_{\alpha\beta_i} > t'_{\alpha\beta_i}$ occurs). Since most unsuccessful interactions from the provider's perspective are filtered by the refusal conditions, apart from the exception cases, the remaining requests are those that α considers for acceptance. Thus, this strategy instantiates Algorithm 15 as follows.

- There are three *refusal conditions*, which represent possible causes of unsuccessful interactions, such that $RC = \{rc_1, rc_2, rc_3\}$. They are described below.

Algorithm 18 Algorithm for the refusal condition *Poor_Compensation*(β).

```
1: input: $\beta$
2: $condition = false$
3: for all $it_j \in IT$ do
4:     $r'_{\alpha\beta} = GetValues(it_j, \beta, \text{“renouncement”}, 2)$
5:     $t'_{\alpha\beta} = GetValues(it_j, \beta, \text{“debt”}, 2)$
6:     if $r'_{\alpha\beta} > t'_{\alpha\beta} \land r'_{\alpha\beta} > 0 \land t'_{\alpha\beta} > 0$ then
7:         $condition = true$
8:     end if
9: end for
10: output: $condition$
```

1. A devaluation (rc_1), represented as:

$$rc_1 = Devalued(\beta_i)$$

where the function *Devalued*(β_i) is defined in Algorithm 11.

2. A low valorisation (rc_2), represented as:

$$rc_2 = Low\_Valorisation(srv, \beta_i) \land \neg\ Only\_Provider(\beta_i)$$

where the function *Low_Valorisation*(srv, β_i), defined in Algorithm 16, identifies an under-valorisation of a service *srv* provided by α to β_i, which occurs when a valorisation $v_{\alpha\beta_i}$ (or credit) received from β_i is less than the valorisation received from previous partners (calculated by the $GetAverageValorisation(srv)$ function in Algorithm 16). The exception for this condition is if β_i is the only provider for a service that α needs (to avoid α losing a cooperation with the only provider), represented by the function *Only_Provider*(β_i), defined in Algorithm 17.

3. A poor compensation (rc_3), represented as:

$$rc_3 = Poor\_Compensation(\beta_i) \land \neg\ Only\_Provider(\beta_i)$$

where the function *Poor_Compensation*(β_i), defined in Algorithm 18, identifies the relation $r'_{\alpha\beta_i} > t'_{\alpha\beta_i}$ in α's history of exchange values, and the exception for this condition is if β_i is the only provider for a needed service.

- When analysing an *individual request*, service provision is restricted to those requesters with which α has a debt, since most unsuccessful interactions from the provider's perspective are filtered by the refusal conditions. Thus, the *individual selection criterion* for this strategy is $totalt_{\alpha\beta} > 0$.

- For a *group of requests*, Q_o is ordered according to two criteria, in sequence: higher *debts*, and fewer *negative balances of exchange values*. That is, Q_o is ordered with requesters with which α has higher debts first and, if two consecutive requesters in

Q_o have the same debt, the one with which α has smaller proportion of negative balances of exchange values in previous interactions ($stb^n_{\alpha\beta_i}$) is preferred over the other. This is represented as:

$$Q_o = \{\langle \beta_1, ..., \beta_n \rangle : Q | \ ((totalt_{\alpha\beta_i} > totalt_{\alpha\beta_{i+1}}) \vee \\ (totalt_{\alpha\beta_i} = totalt_{\alpha\beta_{i+1}} \ \wedge \ stb^n_{\alpha\beta_i} < stb^n_{\alpha\beta_{i+1}})) \Rightarrow \beta_i > \beta_{i+1}\}$$

8.3.4 Combined Strategies

Although the exchange values-based strategy of *analysing cooperative situations* can capture most aspects of the provider's previous interactions with other agents, it has the following limitations: it does not distinguish, among potential cooperations, the requesters that are not able to reciprocate in the future; it does not consider the requesters which, as providers, perform the best service; and it assumes that a provider always prefers to interact with requesters with which it has a debt. However, in a potential cooperation situation, it may be useful to find out if the requester may actually provide a service that the provider may need in the future, so that reciprocation is achievable. In addition, the provider can consider, from among the requesters that are able to reciprocate, which provides the best quality service. Finally, the provider may prefer an interaction in which it will earn a credit (when it does not have a debt) over one in which it will pay a debt; for example, if a requester without a credit usually provides a better service than another requester with credit.

To address these limitations, we define *combined selection strategies*, which consider: information on *dependence* separately from *exchange values*, to find requesters that are able to reciprocate; and, from the exchange values information, consider (in addition to debts and balances of exchange values) the satisfaction of the provider in previous interactions in which it received a service, to find those requesters able to reciprocate with the best quality service from the provider's perspective. These combined strategies also provide alternative preference orderings for group selection so that others are possible in addition to always preferring to pay debts.

Therefore, for the combined strategies, the preference criteria to be considered by a provider are: debt, balance of exchange values, satisfaction (which is related to quality of service), and dependence. The possible causes of negative balances of exchange values are used as refusal conditions, as in the *analysing cooperative situations* strategy. The dependence criterion is used for preference when the provider has no information about exchange values with other agents or when there is no distinction between two requesters regarding the provider's exchange values. Thus, the remaining criteria, debt and satisfaction, are those considered for the main selection. Based on this, we define two different combined strategies, such that one strategy is to prefer *paying debts* over gaining credits (with requesters with which the provider has debts first), and the other is to prefer *gaining credits* over paying debts (with requesters with which the provider has higher satisfaction first). These are described in the next sections.

8.3.4.1 Preference for Paying Debts

The combined strategy with preference for paying debts instantiates the general decision-making in Algorithm 15 as below.

- There are three *refusal conditions*, which represent possible causes of unsuccessful interactions, such that $RC = \{rc_1, rc_2, rc_3\}$. They are described below.

  1. A devaluation (rc_1), represented as:

  $$rc_1 = \mathit{Devalued}(\beta_i)$$

  2. A low valorisation (rc_2), represented as:

  $$rc_2 = \mathit{Low\_Valorisation}(srv, \beta_i) \wedge \neg\ \mathit{Only\_Provider}(\beta_i)$$

  3. A poor compensation (rc_3), represented as:

  $$rc_3 = \mathit{Poor\_Compensation}(\beta_i) \wedge \neg\ \mathit{Only\_Provider}(\beta_i)$$

  These are the same as in the *analysing cooperative situations* strategy. The function $\mathit{Devalued}(\beta_i)$ is defined in Algorithm 11, $\mathit{Low\_Valorisation}(srv, \beta_i)$ in Algorithm 16, $\mathit{Only\_Provider}(\beta_i)$ in Algorithm 17, and $\mathit{Poor\_Compensation}(\beta_i)$ in Algorithm 18.

- When analysing an *individual request*, service provision is restricted to those requesters with which α has a debt or that α depends on. A dependence of α in relation to β is identified if β is in the set D_{on}. Thus, the individual selection criterion for this strategy is $\mathit{totalt}_{\alpha\beta} > 0 \vee \beta \in D_{on}$. Note that this is a combination of the *individual selection criteria* for the dependence-based and exchange values-based strategies.

- A *group of requests* in Q is first ordered with those with which α has higher debts first. For consecutive requesters with the same debt, the one that α depends on is preferred, resulting in sequence Q_m. This is represented as:

  $$Q_m = \{\langle \beta_1, ..., \beta_n \rangle : Q|\ ((\mathit{totalt}_{\alpha\beta_i} > \mathit{totalt}_{\alpha\beta_{i+1}}) \vee \\ (\mathit{totalt}_{\alpha\beta_i} = \mathit{totalt}_{\alpha\beta_{i+1}}\ \wedge\ \beta_i \in D_{on}\ \wedge\ \beta_{i+1} \notin D_{on})) \Rightarrow \beta_i > \beta_{i+1}\}$$

  Finally, if α has the same debt with, or the same dependence with, two consecutive requesters, α's total satisfaction value with each of these requesters is used to decide which is preferred over the other, resulting in sequence Q_o. This is represented below:

$$Q_o = \{\langle \beta_1, ..., \beta_n \rangle : Q_m | \; ((totalt_{\alpha\beta_i} = totalt_{\alpha\beta_{i+i}}) \; \wedge \\ ((\beta_i, \beta_{i+1} \in D_{on}) \; \vee (\beta_i, \beta_{i+1} \notin D_{on})) \; \wedge \; (avs_{\alpha\beta_i} > avs_{\alpha\beta_{i+1}})) \Rightarrow \beta_i > \beta_{i+1}\}$$

8.3.4.2 Preference for Gaining Credits

An alternative strategy for selecting requests is to prefer gaining credits over paying debts. The combined strategy with a preference for gaining credits has the same *refusal conditions* as the previous strategy, based on the causes of negative balances of exchange values, but different criteria for individual and group selection. This strategy instantiates the general decision-making in Algorithm 15 as follows.

- When analysing an *individual request*, service provision is restricted to those requesters with which the provider has higher satisfaction values or to those requesters on which α depends (so that the credit gained can be used to request a service in the future). Thus, the *individual selection criterion* is represented as $avs_{\alpha\beta} > avs_{\alpha_{all}} \vee \beta \in D_{on}$, where $avs_{\alpha_{all}}$ is α's average satisfaction in all previous interactions with other agents.
- A *group of requests* in Q is ordered in sequence Q_m, with those with which α has higher satisfaction first. For consecutive requesters with which the provider has the same satisfaction, α prefers the one with which it has a debt. The ordering of Q_m is given below:

  $$Q_m = \{\langle \beta_1, ..., \beta_n \rangle : Q | \; ((avs_{\alpha\beta_i} > avs_{\alpha\beta_{i+1}}) \vee \\ (avs_{\alpha\beta_i} = avs_{\alpha\beta_{i+1}} \; \wedge \; totalt_{\alpha\beta_i} > totalt_{\alpha\beta_{i+1}})) \Rightarrow \beta_i > \beta_{i+1}\}$$

  Finally, if for consecutive requesters, α has the same satisfaction and the same debt, the one with which α has a dependence is preferred, resulting in sequence Q_o. This is given below:

  $$Q_o = \{\langle \beta_1, ..., \beta_n \rangle : Q_m | \; ((avs_{\alpha\beta_i} = avs_{\alpha\beta_{i+1}} \wedge totalt_{\alpha\beta_i} = totalt_{\alpha\beta_{i+1}}) \wedge \\ (\beta_i \in D_{on} \; \wedge \; \beta_{i+1} \notin D_{on})) \Rightarrow \beta_i > \beta_{i+1}\}$$

8.4 Conclusions

In this chapter, we have presented a requester selection mechanism for autonomous agents that provide resource-consuming services in non-monetary cooperative applications. The analysis of incoming requests is necessary not only to support autonomous decision-making about whether to accept requests, but also to allow agents to sensibly manage their resources so as not to be overwhelmed with services to perform for other agents. Although agents are autonomous and are not obliged to accept requests, in a non-monetary cooperative environment they need to provide services to improve their chances of receiving services from others in the future. The challenge here is thus to balance the number of accepted requests with the availability of resources, so that chances of future interactions are improved and beneficial cooperations are maintained. Without a good balance, an agent may be continuously overwhelmed by services to perform for others while not being able to find available providers to perform the services it needs. An effective way to achieve this balance is to consider information about cooperative and reciprocal relationships between requesters and providers, such as the exchange values resulting from their interactions and service dependence.

Exchange values here motivate service provision in the sense that they form a system of credits and debts that the provider can draw upon to receive services in the future. They can also be used to analyse reciprocal relationships and their quality in order to decide whether to accept requests. Such an analysis is possible through the observation of the balance of exchange values in each cooperative situation in which an agent engages, and indicates whether an agent is getting what it expects from a cooperation in terms of compensation of provided and received services. Based on this, autonomous agents can reason about maintaining or terminating cooperations with others. Finally, information about service dependence between requesters and providers complements the information derived from exchange values, since it helps to determine if a mutual cooperation is possible — the provider may gain a credit after providing a service to a requester, but this credit can only be spent if the requester is able to perform some service that the provider needs in return.

Based on this, we have proposed five different requester selection strategies, which consider either information on service dependence and exchange values separately (to explore their individual advantages), or a combination of both (to gain the benefits of each in a single strategy). Different requester selection strategies or even different preference orderings for criteria in the same strategy (as in the case of the combined strategies) may influence the number and outcomes of interactions that the agent taking the decision engages in. This is because strategies that consider information on reciprocation only, like the *dependence-based* and *simple reciprocation* strategies, focus on improving the chances of future interactions, while strategies that consider additional information on the quality of interactions, like the *analysing cooperative situations* strategy and the *combined* strategies, focus not only on improving the chances of future interaction but also on receiving better services in return.

Chapter 9

Experiments

9.1 Introduction

In the previous chapters we have argued that participants in cooperative and open systems, such as those in the bioinformatics domain, need mechanisms for partner selection, when both requesting and providing services. In particular, in cooperative systems in which there is service variety, participants requesting services are likely to find many alternatives with similar functionality but different quality. Moreover, service requesters may need to compete for busy and resource-bounded providers. In this context, the aim of the provider selection mechanism is to find providers that perform good quality services while being more likely to accept requests.

Different strategies were proposed in Chapter 7 to achieve this aim, some of which focus on just service quality or just the likelihood of achieving cooperation, while others take into account both characteristics during selection. Thus, there may be differences in the efficiency of each strategy in finding providers, and better provider selection strategies are those that allow requesters to find available providers in less time, and to find providers that are able to perform their services to a high quality (according to the requesters' perspective).

Conversely, from the provider's perspective, in a cooperative system in which provided services are required by a wide range of service users, resource-bounded providers must limit the number of services they provide. At the same time, since cooperation is based on reciprocation, providers must select which requests to accept based on existing or potential reciprocal relationships, representing the chance of receiving services in the future. In this context, the aims of the requester selection strategy are to manage the provider's resources by limiting request acceptance and to improve the provider's chances of future interactions by analysing reciprocal relationships with requesters.

We have proposed in Chapter 8 different requester selection strategies to achieve these goals. Although they all consider reciprocation, they use different selection criteria to identify reciprocal relationships and to choose those requesters that are more likely to provide good quality services in the future. If the requester selection strategy to limit requests is efficient, a provider will make less effort in finding available partners to perform services and in finding those with high quality. Requester selection strategies thus influence the performance of provider selection strategies, since selecting requesters efficiently can actually improve the chance of selecting good providers more quickly. Agents therefore depend on both selection processes for effective cooperation.

To test our proposed requester and provider selection strategies in relation to the aspects described above, we undertook an empirical evaluation through several experiments, described in the next sections. These experiments were performed through a Java-based system that simulates the bioinformatics service domain. We start by describing the strategies tested in Section 9.2, and the experimental set-up in Section 9.3. In Section 9.4, we explain the performance measures that we use to compare different provider and requester selection strategies. We then present two sets of experiments in Sections 9.5 and 9.6, and conclude in Section 9.7.

9.2 Experimental Strategies

We have proposed *five* different provider selection strategies that take into account different selection criteria, in Chapter 7. In this section, we provide an overview of these strategies, highlighting the differences and similarities in their selection criteria, to allow comparison of both individual strategies and groups of strategies.

In addition to the strategies defined in earlier chapters, in our experiments we also consider a random strategy as a baseline for comparison. All strategies are listed in Tables 9.1 and 9.2 in which provider and requester selection strategies are indicated using prefixes 'p-' and 'r-' respectively (such as p-SR for the simple reciprocation provider selection strategy or r-SR for the corresponding requester selection strategy). Here, the exchange values-based provider selection strategies are p-SR and p-ACS, which differ in that p-SR uses the requester's credits with each possible provider as the selection criterion, while p-ACS uses, in addition to credits, the balances of exchange values in previous interactions. The evaluation-based provider selection strategy, p-EB, uses evaluations of services received in previous interactions as the selection criterion, and the dependence-based strategy, p-DB, uses service dependence between requester and providers. The combined criteria strategy, p-CC, uses the requester's exchange values, previous service evaluations, and dependence. Finally, the random strategy is p-RD, which selects providers at random.

Similarly, to select requesters, we proposed *five* different requester selection strategies, in Chapter 8. All of these strategies take into account reciprocal relationships in the

Abbrev.	**Provider-Selection Strategy**	**Class**	**Criteria**
p-SR	Simple reciprocation	Reciprocation	Exchange values
p-ACS	Analysis of cooperative situations	Reciprocation	Exchange values
p-EB	Evaluation-based	Non-reciprocation	Service evaluation
p-DB	Dependence-based	Reciprocation	Dependence
p-CC	Combined criteria	Reciprocation	Exchange values, Service evaluation, Dependence
p-RD	Random	-	-

Table 9.1: List of Provider Selection Strategies.

Abbrev.	**Requester-Selection Strategy**	**Class**	**Criteria**
r-SR	Simple reciprocation	Reciprocation	Exchange values
r-ACS	Analysis of cooperative situations	Reciprocation	Exchange values
r-DB	Dependence-based	Reciprocation	Dependence
r-CCP	Combined criteria for paying debts	Reciprocation	Exchange values, Dependence
r-CCG	Combined criteria for gaining credits	Reciprocation	Exchange values, Dependence

Table 9.2: List of Requester Selection Strategies.

form of different selection criteria, as listed in Table 9.2, and some have a corresponding strategy for provider selection (using the same set of criteria, such as r-ACS and p-ACS, or both r-CCP and r-CCG and p-CC). However, since we assume that agents operate in a non-monetary cooperative system in which the motivation to provide services is based on reciprocation, we do not consider requester selection strategies that do not take into account reciprocation, such as evaluation-based and random strategies (so that p-EB and p-RD do not have corresponding requester selection strategies). Importantly, a random requester-selection strategy is not considered in Table 9.2 because when providers select their requesters at random, the functionalities of the strategies used by these requesters (the provider-selection strategies), such as finding providers more likely to reciprocate through dependence or exchange values, do not work (since providers are not reciprocal when selecting at random). Therefore, since providers selecting at random invalidate the functionality of the strategies used by requesters, a random requester-selection strategy (for providers) does not provide a base for comparison among selection strategies. Note that the opposite does not happen, since reciprocity depends on the provider, and requesters selecting at random (with a random provider-selection strategy) do not invalidate the functionality of the strategies used by providers.

The exchange values-based r-SR strategy uses the provider's debts as the selection criterion; the r-ACS strategy uses, in addition to debts, the provider's balances of exchange values and, when these balances are negative, their possible causes. The r-DB strategy uses service dependence between provider and requesters. Finally, the r-CCP and r-CCG strategies use the requester's exchange values, previous service evaluations, and

Strategies		**Selection Criteria**			
Provider	**Requester**	**Reciprocity**		**Service quality**	
		Credits or Debts	Dependence	Balance	Service evaluation
p-SR	r-SR	•			
p-ACS	r-ACS	•		•	
p-EB					•
p-DB	r-DB		•		
p-CC	r-CCP,r-CCG	•	•	•	•
p-RD					

Table 9.3: Selection Criteria for Provider and Requester Selection Strategies.

dependence as selection criteria, the difference between them being that r-CCP uses exchange values with debts preferred over credits, and r-CCG uses exchange values with credits preferred over debts. A summary of the kind of selection criteria used by the various provider and requester selection strategies is shown in Table 9.3.

9.3 The Experimental Set-Up

9.3.1 Providers and Requesters

In our experiments, we simulate a cooperative system in which agents exchange free services, and providers are motivated to provide services to others on the basis of *reciprocation.* Agents are heterogeneous in terms of their abilities to perform a service (so that the same service may be delivered with a different quality by distinct providers), and have different perspectives over evaluation, so that the same service result may also be seen as having different quality by distinct requesters. In addition, more than one agent provides the same service and, in their role as providers, agents have limited resources. This scenario is illustrated in Figure 9.1, in which agents provide and request different services.

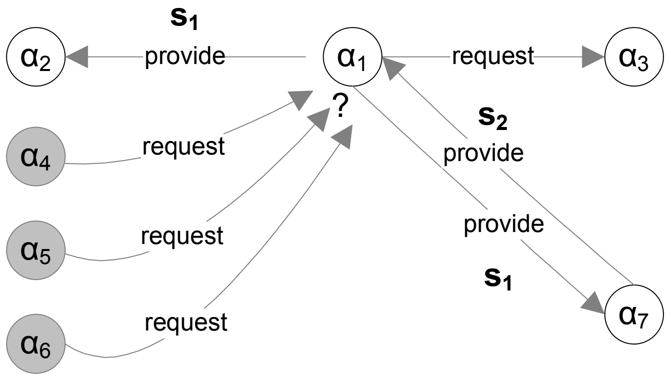


Figure 9.1: Cooperative Scenario.

To simulate a population of agents that provide services of various qualities, each provider is assigned a skill level in the range of $(0, 1]$, which represents the ability

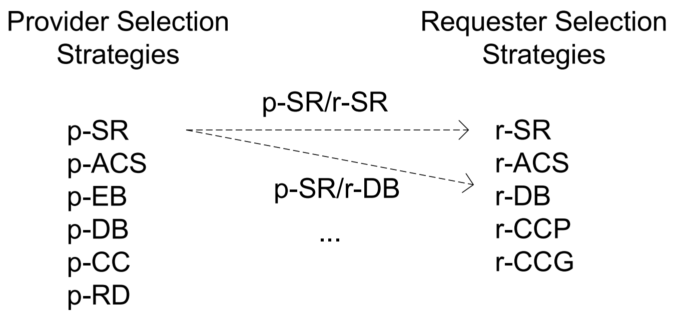


Figure 9.2: Combining provider and requester selection strategies.

of providers to perform a service. To simplify the experiments, we have divided the providers' skill levels into three groups (resulting in roughly similar intervals sizes of 0.4, 0.3, and 0.3), as follows:

- low-skilled providers, with skill levels in the $(0, 0.4]$ range;
- medium-skilled providers, with skill levels in the $(0.4, 0.7]$ range; and
- highly-skilled providers, with skill levels in the $(0.7, 1]$ range.

Similarly, to simulate a population of agents with different perspectives over service evaluation, each requester is assigned an evaluation strictness in the range of $[0.5, 1]$[1], which it uses to determine the evaluation of a received service, as described in our *evaluation method* in Section 5.3.3. We have also divided the requesters' strictness levels in three groups (and using roughly similar interval sizes of 0.2, 0.15, and 0.15), as follows:

- low-strictness requesters, with an evaluation strictness in the $(0.8, 1]$ range;
- medium-strictness requesters, with an evaluation strictness in the $[0.65, 0.8]$ range; and
- high-strictness requesters, with an evaluation strictness in the $[0.5, 0.65)$ range.

As a result, agents with different skill levels and evaluation strictness levels make up the population. In addition, in order to allow reciprocal behaviour, we assume that agents are both providers and requesters, so that each agent is assigned one provider selection strategy and one requester selection strategy, as shown in Figure 9.2. As stated earlier, provider and requester selection strategies are indicated by appropriate prefixes so that combinations of strategies can be similarly represented (for example, as p-DB/r-DB or p-ACS/r-SR). Although many combinations of provider and requester selection strategies are possible, when comparing provider selection strategies we take all agents to have the same requester selection strategy, and *vice versa*.

[1]Values smaller than 0.5 result in extremely low evaluations, so these values are not considered.

Parameter	**Value**	**Description**
TotalAgents	70	Number of agents in the system.
ProvidersPService	8	Number of providers for each available services.
TotalCycles	600	Number of cycles in the simulation.
Tolerance	0.15	Tolerance value for comparing exchange values that are in the $[0, 1]$ range.
MaxCapacity	2	Number of services that can be performed at the same time.
Lskill	10	Proportion of low-skilled providers.
Mskill	80	Proportion of medium-skilled providers.
Hskill	10	Proportion of highly-skilled providers.
Lstric	10	Proportion of agents with low result expectations.
Mstric	80	Proportion of agents with medium result expectations.
Hstric	10	Proportion of agents with high result expectations.

Table 9.4: Default simulation configuration.

Selection strategies are compared through a number of *experiments*, each of which comprises a set of what we call *sub-experiments*. In a sub-experiment, the performance of a *given pair of strategies* (for example, p-DB/r-DB and p-SR/r-DB) is compared under a number of different settings, each defining a *simulation run*. In all our experiments, there are equal numbers of agents using each strategy (with 50% of the agents using one strategy and 50% using the other), and skill levels are also divided equally among agents so that both strategies have the same number of representative agents with low, medium, and high skills (otherwise, if one strategy has more low-skilled providers than the competitor strategy, this may bias the final comparison).

A simulation run is made up of a fixed number of cycles, in each of which every agent in the population takes an action such as requesting a service, providing a service, evaluating results, and so on. If a request is accepted and a service is provided, we say that there was an *interaction* between the two agents. Thus, for an interaction to occur, a request must be accepted (the act of sending a request or receiving a refusal for a request are not considered an interaction). Once the number of cycles is reached, the simulation run finishes. After all sub-experiments finish, the results are analysed.

To increase the number of requests received by providers, so that we can test the requester selection strategies, some fixed proportion of the total population of agents only send requests but do not provide any service (so that the load of requests on providers increases).

9.3.2 Simulation Configuration

A basic simulation configuration is shown in Table 9.4. Since agents can have three different skill levels (that is, low-skilled, medium-skilled and highly skilled), and these are divided equally among agents of each strategy, the minimum number of agents in the simulation (TotalAgents) is six (three for each strategy, one for each skill level), and to simulate a system in which several requests are sent and received the default number of agents is set to 70.

The number of providers per service (ProvidersPService) must be greater than 1, so that agents have alternatives to choose from and the provider selection strategies can be tested. The total number of cycles (TotalCycles) indicates the duration of the simulation run: the longer the simulation, the better the choices made by agents since their strategies will have more information available during selection. The tolerance value (Tolerance), used only for exchange values-based selection strategies, as specified in Tables 9.1 and 9.2, enables comparisons between exchange values on the $[0, 1]$ scale. Differences below this value are considered insignificant, so that roughly similar values are deemed equal. Based on the results in Section 9.4.1, the tolerance should be less than or equal to 0.15 so that exchange values with different quantities are discriminated.

The provider's maximum capacity (MaxCapacity) must be greater than 1 to select from among groups of requests as part of the requester selection strategies, but should be less than the number of requests received in order to generate competition for resources. High-skilled and low-skilled providers (Hskill and Lskill) are set to be 10% of the population, and the medium-skilled providers (Mskill) are set to be 80% in the basic configuration. These values are somewhat arbitrary, but reflect what we take to be a common distribution, and the same proportions are used for low, medium and high strictness requesters (Lstric, Mstric and Hstric).

Different strategies are compared pairwise using the same configuration. For example, if the aim of an experiment is to compare the random strategy with our five proposed provider strategies using the r-SR requester selection strategy, we must then perform five sub-experiments, one for each pair of strategies (for example, p-SR/r-SR versus p-RD/r-SR, p-ACS/r-SR versus p-RD/r-SR, p-CC/r-SR versus p-RD/r-SR, p-DB/r-SR versus p-RD/r-SR, and p-EB/r-SR versus p-RD/r-SR), using the same configuration, and compare the results.

All results are tested for statistical significance, as shown in Appendix B.

9.4 Performance Measures

To compare the efficiency of different selection strategies, given the aims of both provider and requester selection mechanisms, we consider two performance measures, as described below.

- The *total interactions* is defined as the percentage of *accepted* requests (from those sent), resulting in an interaction between requester and provider. It provides a measure of the ability of requesters to find available interaction partners even though providers may limit request acceptance due to individual preferences and resource constraints.

- The *average satisfaction* with received services provides a measure of whether agents can identify good providers even though there are providers with different skills and good providers may not always be available.

The *total interactions* achieved by agents is used to compare both provider and requester selection strategies. The *average satisfaction* indicates the efficiency not only of the provider selection strategies alone, but also of their use with different requester selection strategies. Since exchange values are used as a selection criterion by some of the strategies, and are compared through a tolerance threshold, we consider how this tolerance value influences the performance measures described above.

9.4.1 Determining the Tolerance Threshold

Exchange values-based strategies that use the balance of exchange values as a criterion to select providers and requesters may be affected by the tolerance threshold used to compare two exchange values. If the difference between two exchange values is smaller than or equal to the tolerance value, they are considered to be equal. To determine an appropriate tolerance threshold for use in our later experiments, we undertook sub-experiments with a number of simulation runs, with a population of 70 agents, with services to request and services to provide.

Since the impact of the tolerance value is on selection criteria, to sensibly analyse this impact, agents take a combination of provider and requester selection strategies that use corresponding selection criteria (that is p-ACS/r-ACS, p-CC/p-CCP, and p-CC/r-CCG, as detailed in Table 9.3). In each simulation a different tolerance threshold was used,[2] varying from 0 to 1, and different selection strategies were compared in terms of the two performance measures.

As shown in Figure 9.3(a), the tolerance threshold has no impact on the *total interactions*, only on the *average satisfaction* (shown in Figure 9.3(b)), which is mainly affected when agents use the p-ACS/r-ACS selection strategies, which achieve lower average satisfaction for tolerance values higher than 0.15. This is because for this strategy the balance of exchange values (to which tolerance applies) is the only criterion that indicates service quality, so assessment of service quality also is affected. Although the combined criteria strategies p-CC/r-CCP and p-CC/r-CCG also use the balance of exchange values, they include consideration of satisfaction with received services, which is determined from service evaluation and is not influenced by the tolerance value.

[2]The tolerance value is defined in a $[0, 1]$ range, as in Section 6.5.

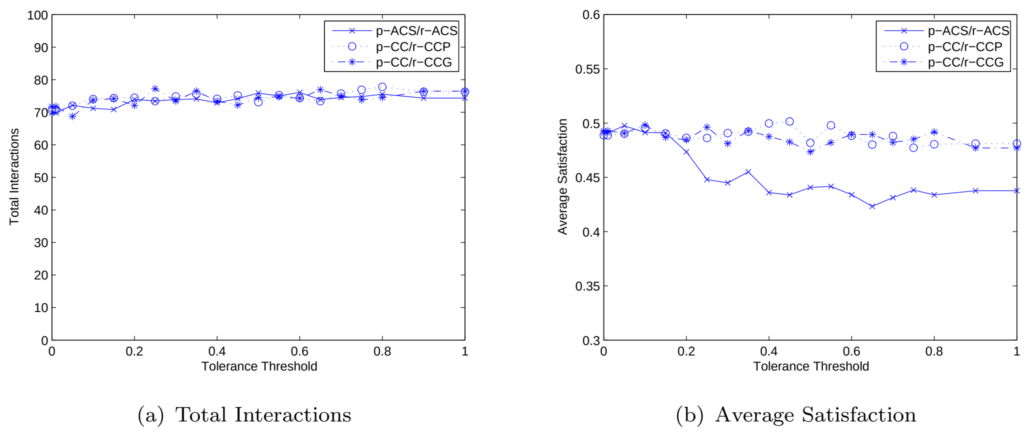


FIGURE 9.3: Impact of the tolerance value on performance measures.

9.5 Busy Providers and Reciprocity

When service providers are busier, requesters must compete for them, possibly taking too long for their requests to be accepted. Conversely, busy providers must cope with bounded resources by limiting the number of requests they accept, potentially reducing their chances of future interactions. In the first of our experiments, therefore, we tested the effect of busy providers on the performance of selection strategies in terms of the *total interactions* achieved by agents.

Hypothesis 1: *when providers are busier, due either to an increase in the number of requests that they receive or to a decrease in their resource capacity, provider selection strategies that use reciprocity to find providers more likely to accept their requests achieve higher total interactions than non-reciprocation based strategies (since they take less time for their requests to be accepted).*

To test this hypothesis, we compared those strategies using reciprocity — the exchange values-based (p-SR and p-ACS), dependence-based (p-DB), and combined criteria (p-CC) strategies — with those that do not — the evaluation-based (p-EB) strategy, as specified in Table 9.1. The demand on providers was increased by increasing the number of agents (and requests) in the population in Experiment 1, and by decreasing the capacity of providers in Experiment 2. Results for both experiments are presented next.

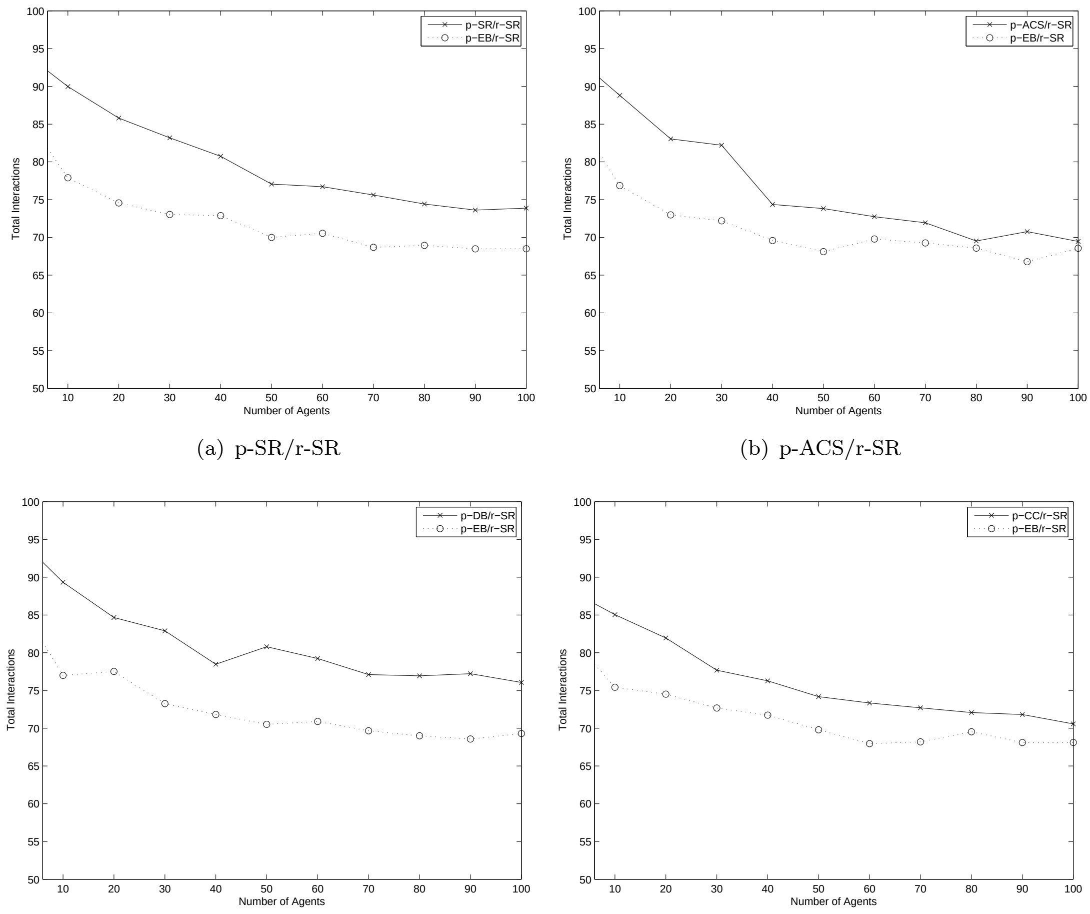
9.5.1 Experiment 1: Increasing Numbers of Agents (and Requests)

In the first experiment, we used the default simulation configuration in Table 9.4, but varied the number of agents in the population (TotalAgents) from 6 to 100 (in intervals[3] of 10), with each simulation being undertaken with a different population size.

[3]Note that the first interval is 4 (from 6 to 10), since we start with 6 agents, given the minimum of 3 agents per strategy.

Figure	**Requester**	**Provider** a	**Provider** b	**Provider** c	**Provider** d
Figure 9.4	**r-SR**	p-SR	p-ACS	p-DB	p-CC
Figure 9.5	**r-DB**	p-SR	p-ACS	p-DB	p-CC
Figure A.1	**r-ACS**	p-SR	p-ACS	p-DB	p-CC
Figure A.2	**r-CCP**	p-SR	p-ACS	p-DB	p-CC
Figure A.3	**r-CCG**	p-SR	p-ACS	p-DB	p-CC

Table 9.5: Combinations of provider selection strategies for each requester selection strategy, in comparison to p-EB, in Experiment 1.



(a) p-SR/r-SR (b) p-ACS/r-SR

(c) p-DB/r-SR (d) p-CC/r-SR

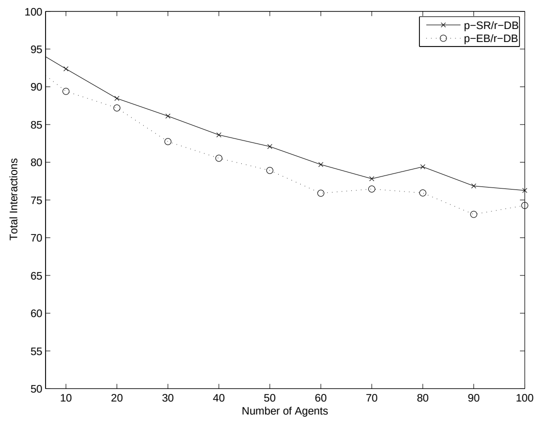
Figure 9.4: Total interactions of provider selection strategies using r-SR.

Then, each of the four reciprocation-based provider selection strategies was compared *pairwise* against the evaluation-based provider selection strategy p-EB. However, since the requester selection strategy is also relevant in determining the *total interactions*, we ran this combination of strategies five times, once for each different requester selection strategy. Table 9.5 indicates the combinations of strategies run (grouped by requester selection strategy), and identifies the figures in which they are shown: Figure 9.4 shows the results of using the *simple reciprocation* requester selection strategy, such that the combinations of provider and requester selection strategies being compared against p-EB/r-SR are p-ACS/r-SR, p-SR/r-SR, p-DB/r-SR, and p-CC/r-SR. The other figures are similar, but for the r-DB, r-ACS, r-CCP and r-CCG strategies.

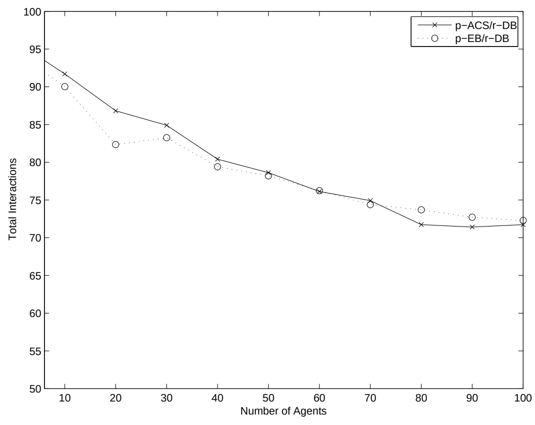
According to the results, with r-SR, all reciprocation-based provider selection strategies, p-ACS, p-CC, p-SR and p-DB, achieve *more total interactions* than the evaluation-based strategy (p-EB), except for p-ACS in a population of more than 80 agents, as shown in Figure 9.4. With the *dependence-based* requester selection strategy (r-DB), shown in Figure 9.5, p-DB and p-CC achieve *more total interactions* than p-EB as the number of agents increases (above 10), but the difference in performance is less substantial than with r-SR, in particular for p-SR and p-ACS. For the p-ACS and p-SR strategies against p-EB, however, there is now little difference in performance, with similar total interactions. This is because the r-DB strategy uses dependence to select requests, unlike the pure exchange values-based provider selection strategies, so that the latter are not successful in *predicting* the requesters more likely to cooperate. Conversely, the dependence-based p-DB strategy performs better against the p-EB strategy when using the r-DB strategy, since they both use dependence to select requesters. Results for the remaining three requester selection strategies, which use the quality of interactions in addition to reciprocity (r-ACS, r-CCP and r-CCG), are shown in Figures A.1, A.2, and A.3 in Appendix A.1. Here, p-DB and p-SR achieve *more total interactions* than p-EB as the number of agents increases, and p-ACS and p-CC outperform p-EB in some cases (with similar performance in others).

A summary of the performance of all reciprocation-based provider selection strategies against the evaluation-based strategy, for each requester selection strategy, is shown in Figure 9.6. Here, among the reciprocation-based strategies, the simple reciprocation provider selection strategy (p-SR) and the dependence-based provider selection strategy (p-DB) achieve the highest total interactions in most cases and are thus best at finding providers more likely to reciprocate. This shows that provider selection strategies that take into account service quality in addition to reciprocity (p-ACS and p-CC, as specified in Table 9.3) are less efficient in finding providers more likely to reciprocate than those that use reciprocity alone (p-SR and p-DB, as specified in Table 9.3).

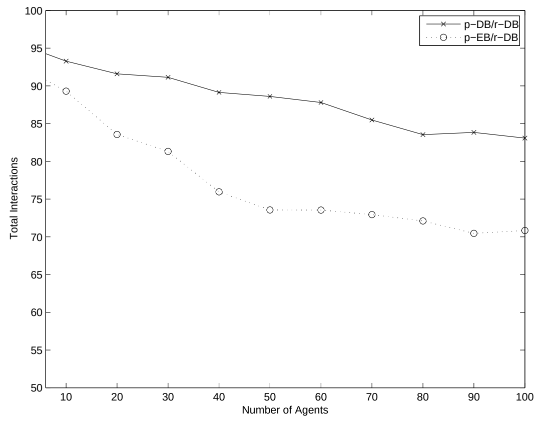
Conversely, in Figure 9.7, we show the results by provider selection strategy, to compare the impact of requester selection strategies on performance. It is clear that provider selection strategies p-EB, p-DB and p-CC perform best when using r-DB, but for strategies p-SR and p-ACS it is not significantly better than using r-CCP and r-CCG (because p-SR and p-ACS use selection criteria that are different to those used by r-DB). In addition, although r-CCP and r-CCG improve the performance of provider selection strategies (except p-DB) in most cases, as shown in Figures 9.7(a), 9.7(b), 9.7(c), and 9.7(e), this is also not significant. Note that all of r-DB, r-CCP and r-CCG use dependence as selection criteria, indicating that by using this for requesters providers are able to choose those requesters that are likely to reciprocate in the future.



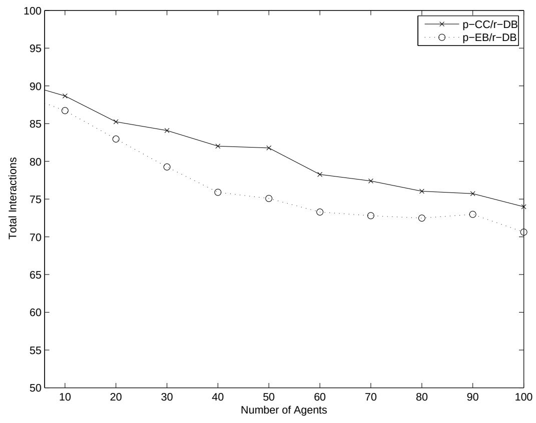
(a) p-SR/r-DB



(b) p-ACS/r-DB



(c) p-DB/r-DB



(d) p-CC/r-DB

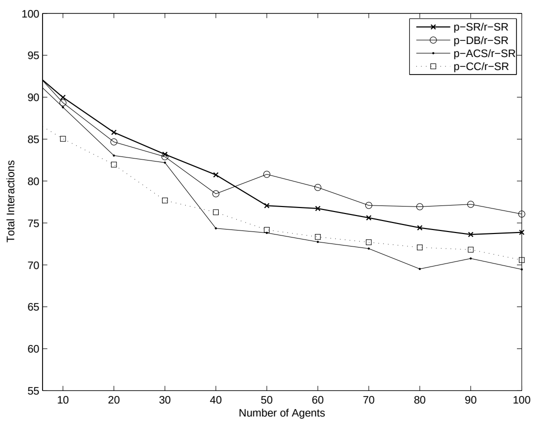
FIGURE 9.5: Total interactions of provider selection strategies using r-DB.

9.5.2 Experiment 2: Decreasing Resource Capacity

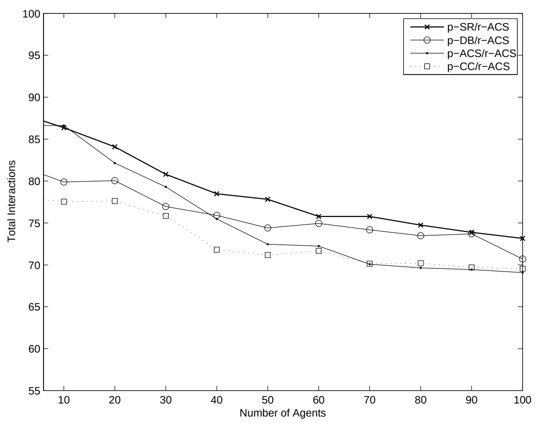
Figure	**Requester**	**Provider** a	**Provider** b	**Provider** c	**Provider** d
Figure 9.8	**r-SR**	p-SR	p-ACS	p-DB	p-CC
Figure 9.9	**r-DB**	p-SR	p-ACS	p-DB	p-CC
Figure A.4	**r-ACS**	p-SR	p-ACS	p-DB	p-CC
Figure A.5	**r-CCP**	p-SR	p-ACS	p-DB	p-CC
Figure A.6	**r-CCG**	p-SR	p-ACS	p-DB	p-CC

TABLE 9.6: Combinations of provider selection strategies for each requester selection strategy, in comparison to p-EB, in Experiment 2.

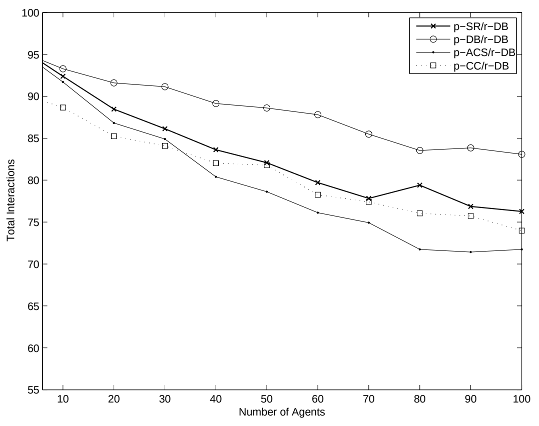
In a complementary experiment, we decreased the *providers' resource capacity*, reducing the number of requests that the providers can satisfy simultaneously. Again, the default simulation configuration in Table 9.4 was used, but varying the number of services that a provider can perform at the same time (MaxCapacity) from 4 to 1 (in intervals of 1). Each of the four reciprocation-based provider selection strategies (p-ACS, p-SR, p-DB, and p-CC) was compared *pairwise* against the evaluation-based provider selection strategy (p-EB), using different requester selection strategies. The combination



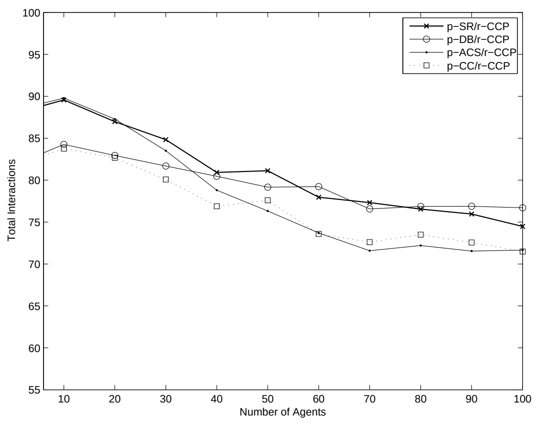
(a) r-SR



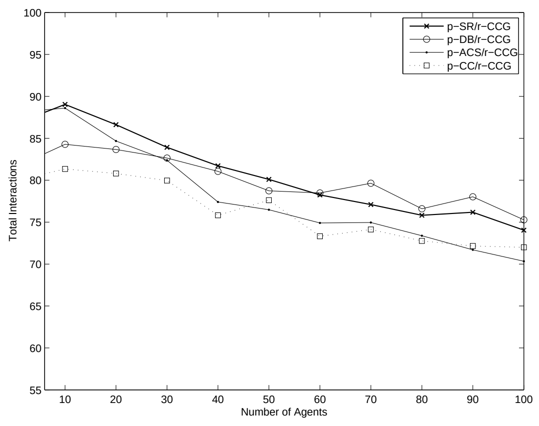
(b) r-ACS



(c) r-DB



(d) r-CCP

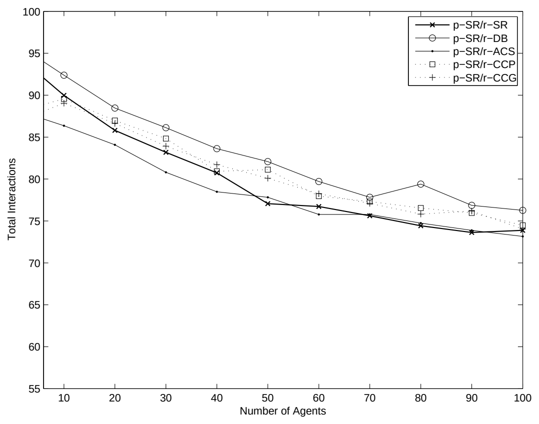


(e) r-CCG

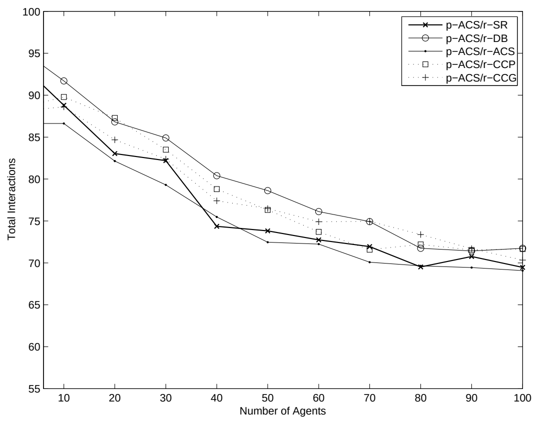
FIGURE 9.6: Summary of results for all reciprocation-based provider selection strategies in terms of total interactions, for each requester selection strategy.

of strategies and the figures in which they are shown are presented in Table 9.6.

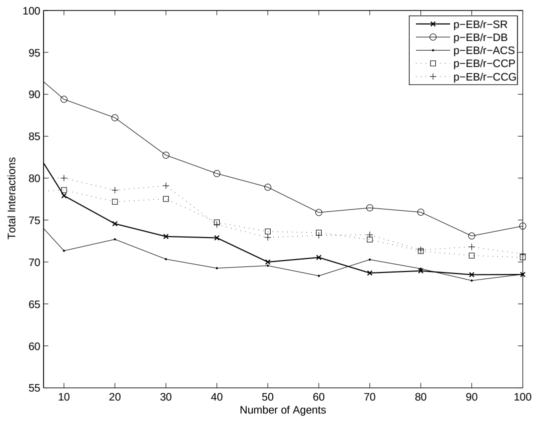
Results for r-SR are shown in Figure 9.8, in which the total interactions decrease with the provider's capacity. Here, the p-EB strategy achieves *fewer total interactions* when competing with *any other reciprocation-based provider selection strategy.* This difference



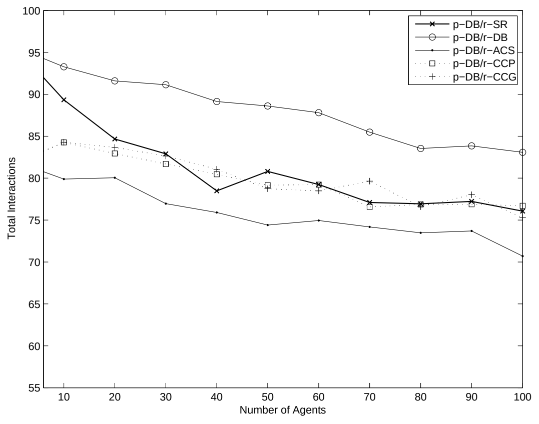
(a) p-SR



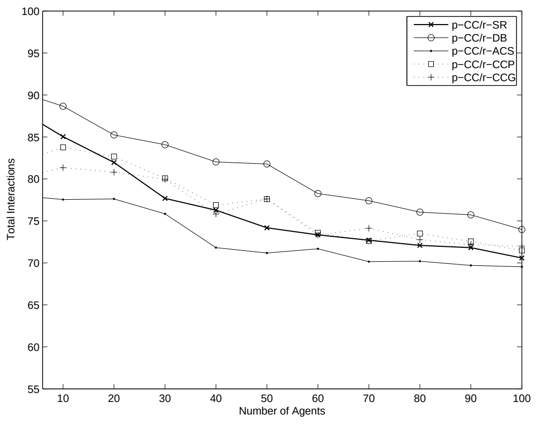
(b) p-ACS



(c) p-EB



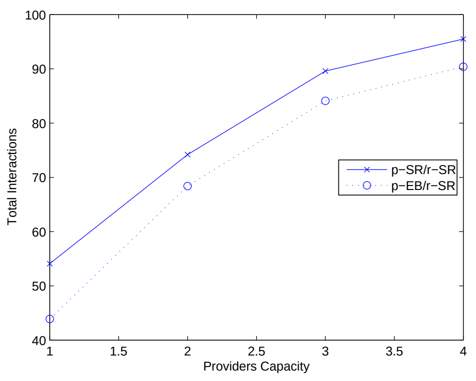
(d) p-DB



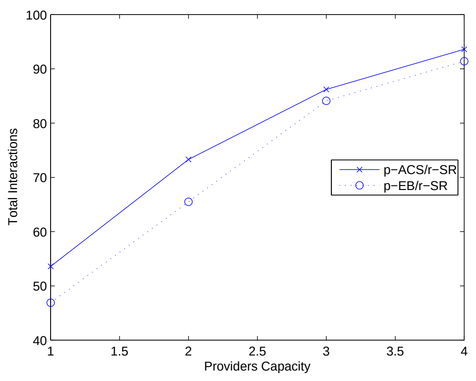
(e) p-CC

FIGURE 9.7: Influence of requester selection strategies on the total interactions.

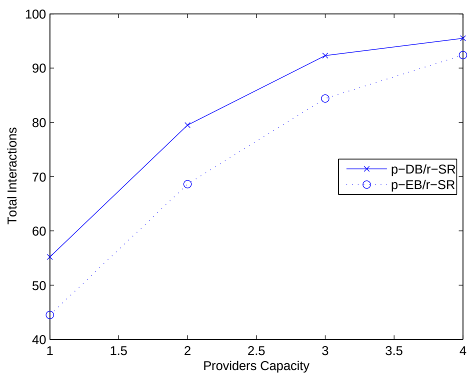
is more substantial when the provider's capacity is low, which indicates that finding providers more likely to reciprocate becomes relevant when there is greater competition for request acceptance. Reciprocation-based provider selection strategies also achieve *more total interactions* than p-EB with all other requester selection strategies (r-DB, r-ACS, r-CCP and r-CCG), as shown in Figures 9.9, A.4, A.5, and A.6. Note that the



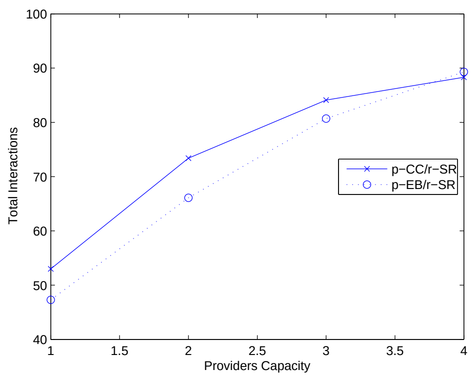
(a) p-SR/r-SR



(b) p-ACS/r-SR



(c) p-DB/r-SR

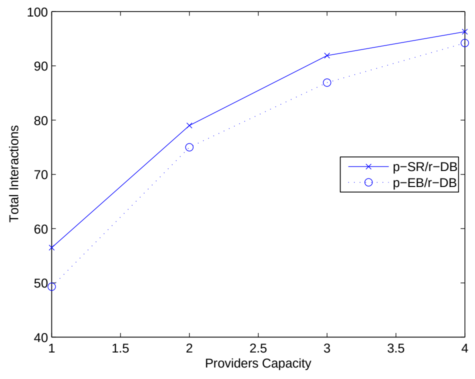


(d) p-CC/r-SR

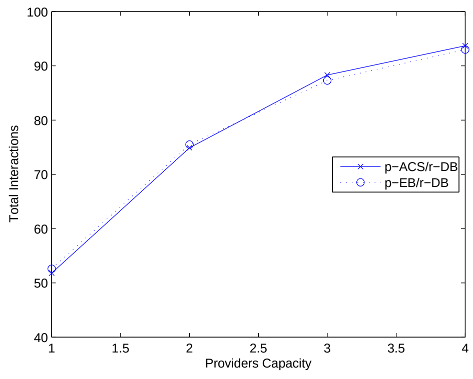
FIGURE 9.8: Total interactions for provider selection strategies using the r-SR requester selection strategy when varying provider capacity.

only case in which p-EB performs similarly to a reciprocation-based strategy is against p-ACS, when used with the r-DB requester selection strategy.

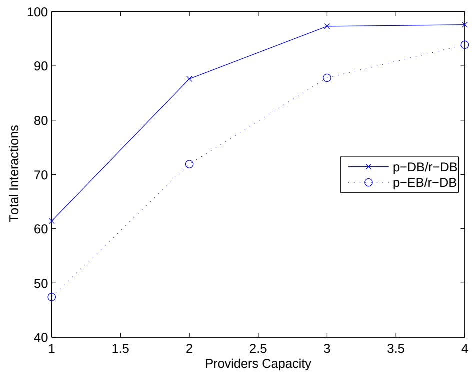
In addition, to determine the impact of requester selection strategies on provider selection, we have grouped these results by provider selection strategy, in Figure 9.10. Figure 9.10(a) shows that the exchange values-based p-SR achieves more total interactions when used with any of the combined criteria requester selection strategies (r-CCP and r-CCG), or with the dependence-based r-DB. However, in the extreme case in which available resources are minimal (that is, a capacity of 1), r-CCP and r-CCG outperform r-DB. Similarly, in Figure 9.10(b), higher total interactions are achieved for the exchange values-based p-ACS when it is used with both r-CCP and r-CCG. The evaluation-based p-EB also has better performance in terms of the total interactions when used with r-CCP, r-CCG and r-DB, as shown in 9.10(c) but, with low capacity, the performance of p-EB is higher with r-CCP than with r-DB and r-CCG. Conversely, for the dependence-based p-DB, the best requester selection strategy is the corresponding r-DB, which also uses dependence, as shown in 9.10(d) but, with low capacity, the performance of p-DB is higher with r-DB and r-CCG than with r-CCP. Finally, the combined criteria p-CC



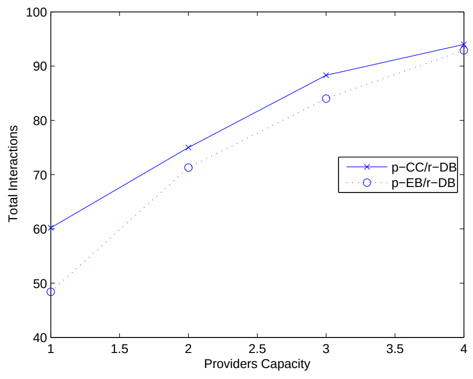
(a) p-SR/r-DB



(b) p-ACS/r-DB



(c) p-DB/r-DB

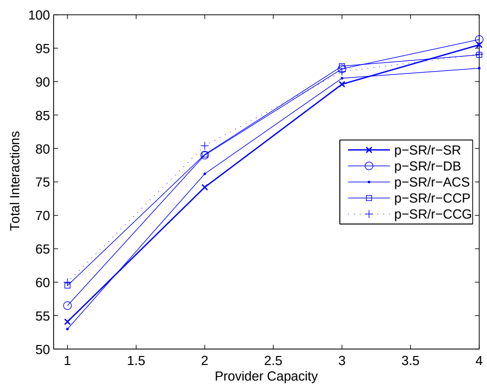


(d) p-CC/r-DB

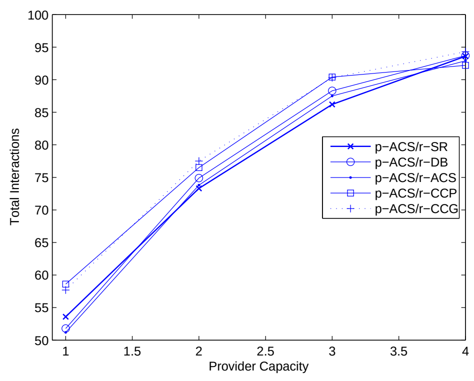
FIGURE 9.9: Total interactions for provider selection strategies using the r-DB requester selection strategy when varying provider capacity.

performs best with r-CCP and r-CCG, which use similar selection criteria as p-CC, and with the dependence-based r-DB, as shown in Figure 9.10(e).

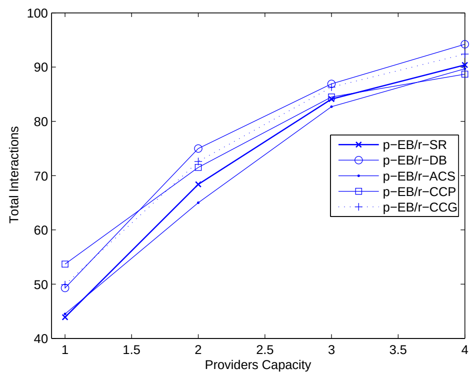
From these results we can conclude that, as in the previous experiment, requester selection strategies using service dependence have the best performance. Here, in all cases, the requester selection strategies using combined criteria, r-CCP and r-CCG, *improve the performance of all provider selection strategies*, in comparison to the pure exchange values-based r-SR and r-ACS. In addition, although the dependence-based r-DB outperforms the combined criteria strategies r-CCP and r-CCG in some cases in which providers have high capacity, r-CCP and r-CCG enable provider selection (except p-DB) to achieve equal or more interactions than r-DB (as shown in Figures 9.10(a), in situations in which resource capacity is low 9.10(b), 9.10(c) and 9.10(e)).



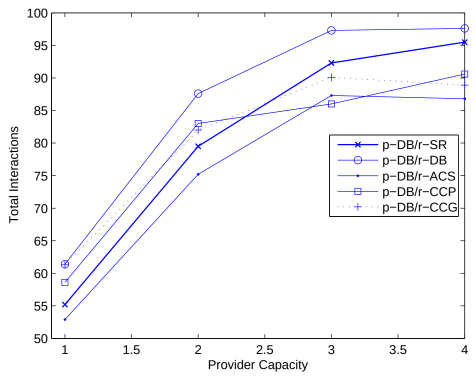
(a)



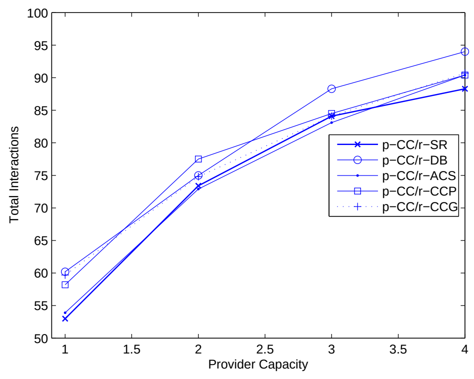
(b)



(c)



(d)



(e)

FIGURE 9.10: Comparing requester selection strategies when varying provider capacity.

9.6 Low-skilled Providers and Received Service Quality

In addition to the time taken to find an available provider, quality of service is also relevant in determining whether agents are able to achieve their goals, in particular if they require service results of a certain quality level, or if they are in danger of not selecting good quality services. As mentioned above, some provider selection strategies

Figure	**Requester**	**Provider** a	**Provider** b	**Provider** c	**Provider** d	**Provider** e
Figure 9.11	**r-SR**	p-SR	p-ACS	p-DB	p-CC	p-EB
Figure A.7	**r-DB**	p-SR	p-ACS	p-DB	p-CC	p-EB
Figure A.8	**r-ACS**	p-SR	p-ACS	p-DB	p-CC	p-EB
Figure A.9	**r-CCP**	p-SR	p-ACS	p-DB	p-CC	p-EB
Figure A.10	**r-CCG**	p-SR	p-ACS	p-DB	p-CC	p-EB

Table 9.7: Combinations of provider selection strategies for each requester selection strategy, in comparison to p-RD in terms of average satisfaction, in Experiment 3.

take into account just service quality or just reciprocity, while others focus on both, as detailed in Table 9.3. Requesters using an evaluation-based provider selection strategy (such as p-EB) are likely to be able to select good quality providers, since service quality is the only criterion used, but other strategies based on reciprocation alone (such as p-SR and p-DB) may not be able to do so. However, we believe that provider selection strategies using service quality in *combination* with reciprocity either *indirectly* through the balance of exchange values (such as p-ACS), or *directly* through service evaluation (such as p-CC), are able to choose good quality providers as well as a pure evaluation-based strategy. We can state this hypothesis more clearly, as follows.

Hypothesis 2: *provider selection strategies that use the balance of exchange values (p-ACS) or a combination of selection criteria (p-CC) can maintain* higher average satisfaction *than a strategy using pure service evaluation, even when the number of low-skilled providers increases.*

In addition, we argue that requester selection strategies that use service quality (as specified in Table 9.3) enable providers to build relationships with agents providing good quality services so that, subsequently, when providers become requesters, not only will they find agents willing to reciprocate, but agents providing better services willing to reciprocate. Thus, for example, agents using the p-ACS provider selection strategy when faced with the r-CCP requester selection strategy may achieve higher *average satisfaction* than agents using p-ACS when faced with r-SR, which does not take service quality into account.

Hypothesis 3: *requester selection strategies that consider either the satisfaction of the receiver or the balance of exchange values in addition to reciprocity (r-ACS, r-CCP and r-CCG) improve the performance of provider selection strategies in relation to the* average satisfaction *when the number of low-skilled providers increases.*

In this section, therefore, we test the performance of selection strategies, focussing on p-ACS and p-CC, in terms of their *average satisfaction* with received services, when there is an increasing number of low quality services in the system.

9.6.1 Experiment 3: Provider Selection Strategies Against Baseline

More specifically, we fix the number of agents in the population and the capacity of providers to 70 and 2, respectively, according to the default configuration described in Table 9.4, and vary the proportion of low-skilled providers in the population from 0 to 100% (in intervals of 10). In each simulation, for each interaction in which an agent participates, we record the satisfaction with the received service (or the service evaluation), and use these values to calculate the *average satisfaction*.

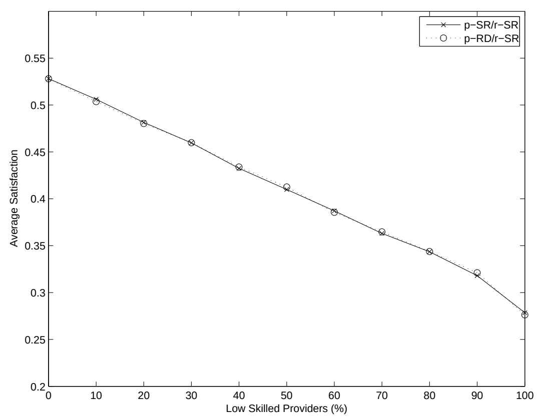
Since we do not distinguish between reciprocity or otherwise here, we compare all provider selection strategies (p-SR, p-ACS, p-EB, p-DB, and p-CC) with the baseline random strategy (p-RD). As previously, these strategies are used with different requester selection strategies to assess the performance of all. The combination of strategies and the figures in which they are shown are specified in Table 9.7.

We start by comparing all provider selection strategies against the baseline strategy (p-RD) when using the r-SR requester selection strategy. As expected, the results in Figure 9.11 show that p-ACS, p-EB, and p-CC achieve *higher average satisfaction* than p-RD, and that p-SR and p-DB are similar in performance to p-RD.

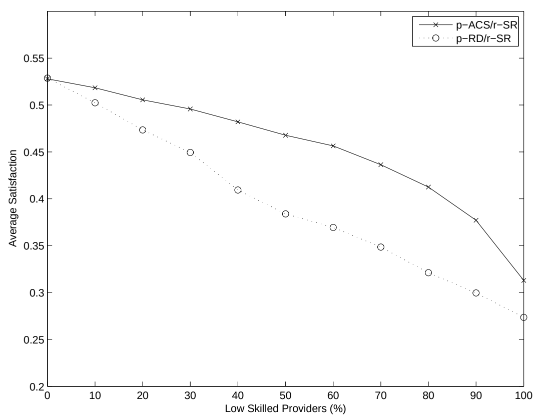
Similar results are produced when these strategies are used with other requester selection strategies, r-DB, r-ACS, r-CCP and r-CCG, as shown in Appendix A (in Figures A.7, A.8, A.9, A.10), and summarised in Figure 9.12 (for all provider selection strategies against p-RD when using different requester selection strategies). When there are few low-skilled providers (less than 10%), provider selection strategies perform similarly, since it is easier to find good providers.

However, when the number of low-skilled providers increases, provider selection strategies that use service quality (p-ACS, p-EB, and p-CC) perform best, achieving *higher average satisfaction* than the others (p-SR and p-DB). Finally, among the best performing strategies, p-ACS and p-CC achieve *similar* average satisfaction to the p-EB strategy. Thus, although p-ACS and p-CC both perform well, they do not clearly outperform p-EB, in contrast to Hypothesis 2.

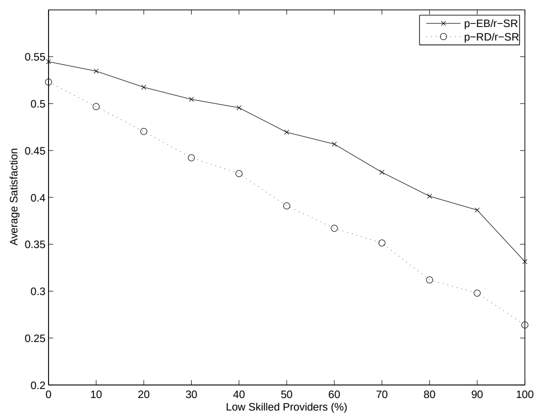
We also analyse the performance of different *requester selection strategies*, as they influence the performance of provider selection strategies in terms of average satisfaction (as in Hypothesis 3). In Figure 9.13, we show the results by provider selection strategy, according to which there is little difference in the average satisfaction achieved by all provider selection strategies when using different requester selection strategies. This suggests that, when strategies compete with the baseline strategy, performance in terms of average satisfaction depends on the provider selection strategy alone. This is because provider selection strategies generally do not compete with the random strategy for service providers.



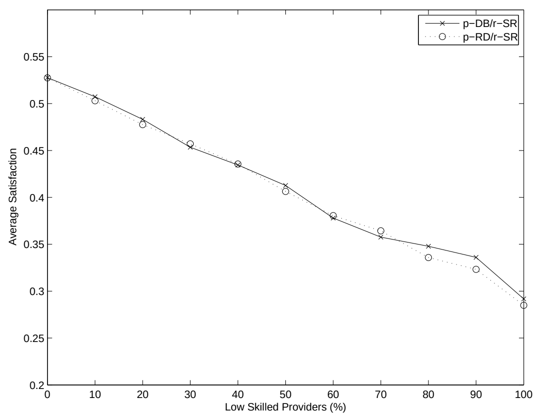
(a) p-SR/r-SR



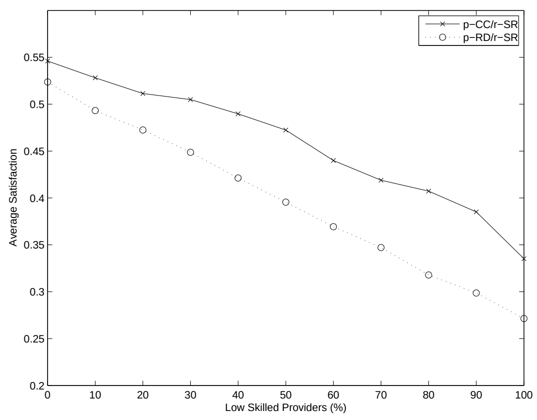
(b) p-ACS/r-SR



(c) p-EB/r-SR



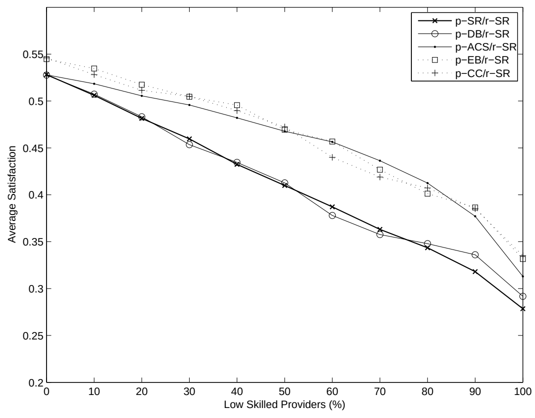
(d) p-DB/r-SR



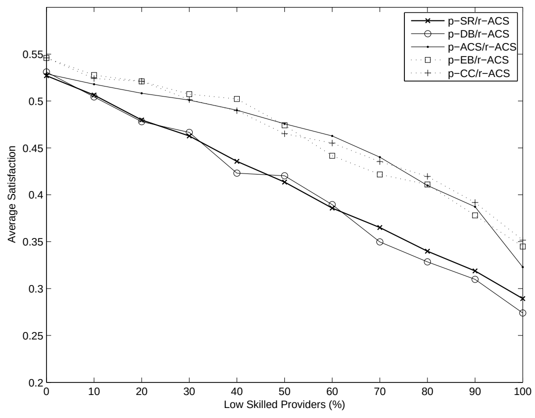
(e) p-CC/r-SR

FIGURE 9.11: Average satisfaction for all provider selection strategies against the baseline with r-SR requester selection strategy.

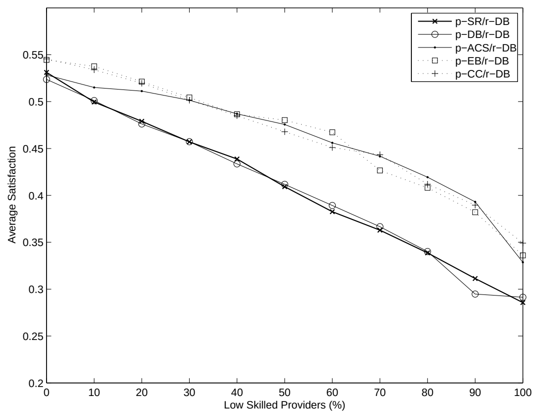
In summary, when competing in the same population as the baseline strategy, the pure exchange values-based p-ACS and combined criteria p-CC achieve similar average satisfaction to the evaluation-based p-EB (not supporting Hypothesis 2). The results also do not support Hypothesis 3, since there is no evidence to suggest that requester selection strategies influence the performance of provider selection strategies in terms of



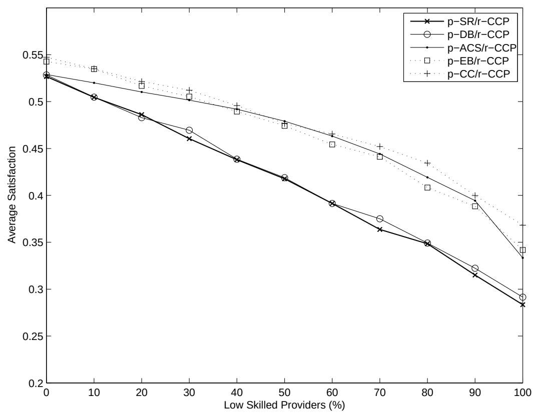
(a) r-SR



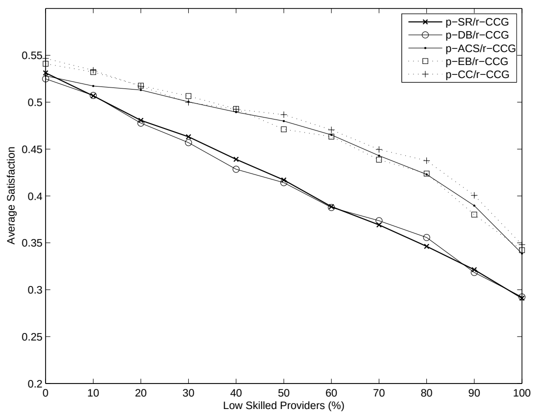
(b) r-ACS



(c) r-DB



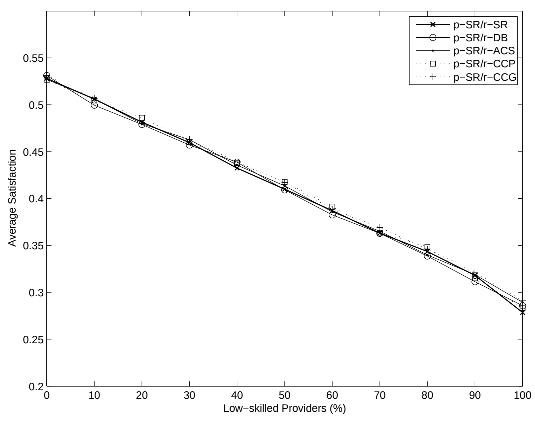
(d) r-CCP



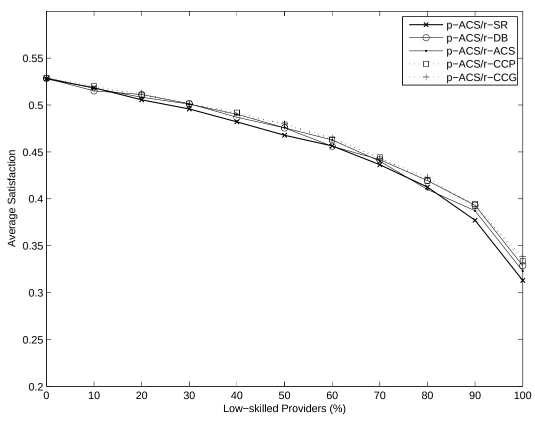
(e) r-CCG

FIGURE 9.12: Comparing provider selection strategies against the random strategy regarding the agents' average satisfaction.

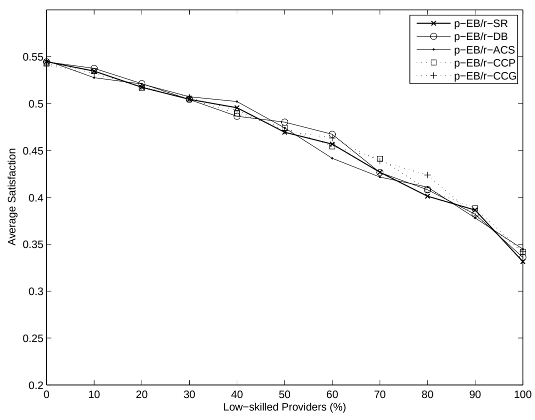
average satisfaction. Yet, while these results show the performance of p-ACS, p-CC, and p-EB strategies individually against a baseline strategy, further testing is required for Hypotheses 2 and 3, with strategies p-ACS and p-CC compared pairwise against p-EB (so that they compete in the *same* population of agents). This experiment, Experiment 4, is described next.



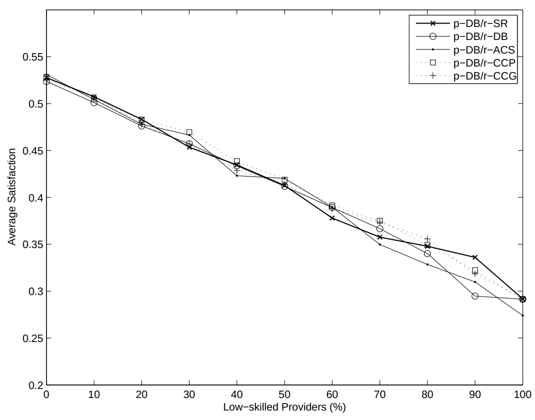
(a)



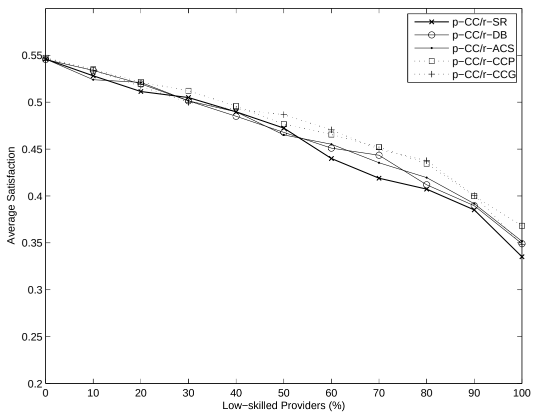
(b)



(c)



(d)

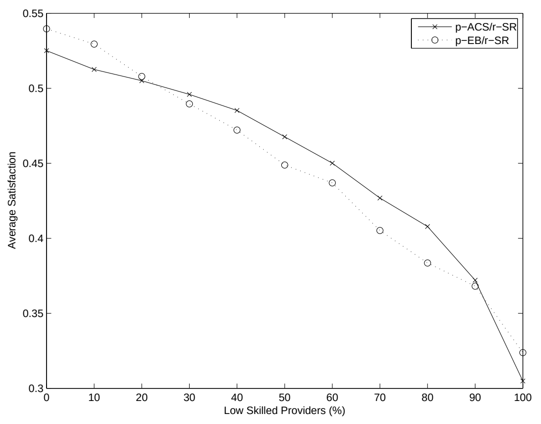


(e)

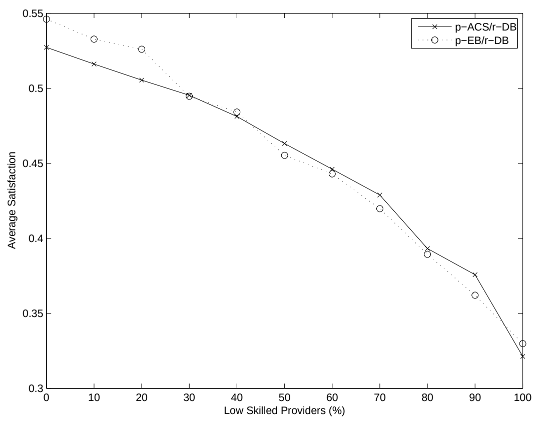
FIGURE 9.13: Comparing requester selection strategies in terms of average satisfaction when varying low-skilled providers.

9.6.2 Experiment 4: Pairwise Analysis of Provider Selection Strategies

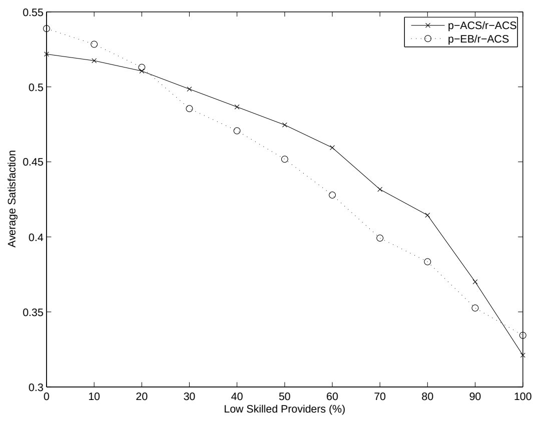
In this additional experiment, we use the same set-up as previously, but compare different pairs of provider selection strategies, so that the pairs are p-ACS/r-SR and p-EB/r-SR, and p-CC/r-SR and p-EB/r-SR (and the same pairs with all other requester selection



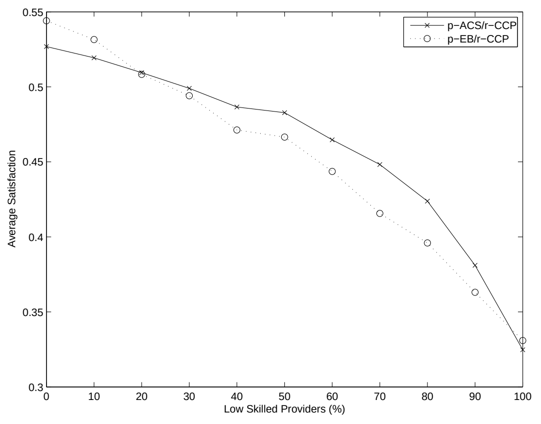
(a) p-ACS/r-SR and p-EB/r-SR



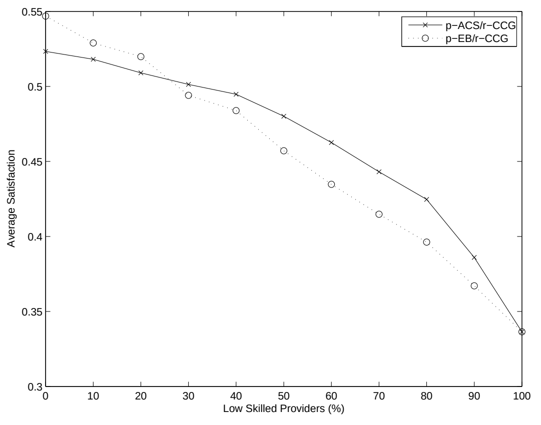
(b) p-ACS/r-DB and p-EB/r-DB



(c) p-ACS/r-ACS and p-EB/r-ACS



(d) p-ACS/r-CCP and p-EB/r-CCP

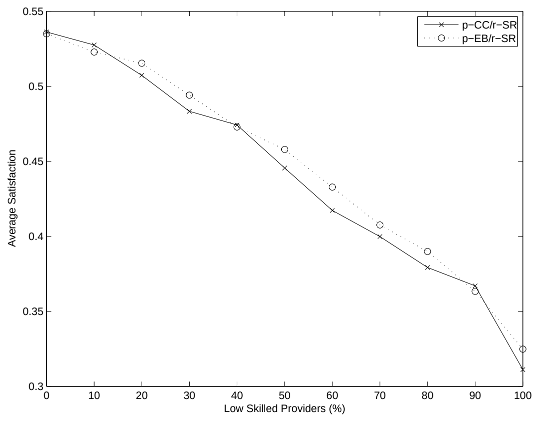


(e) p-ACS/r-CCG and p-EB/r-CCG

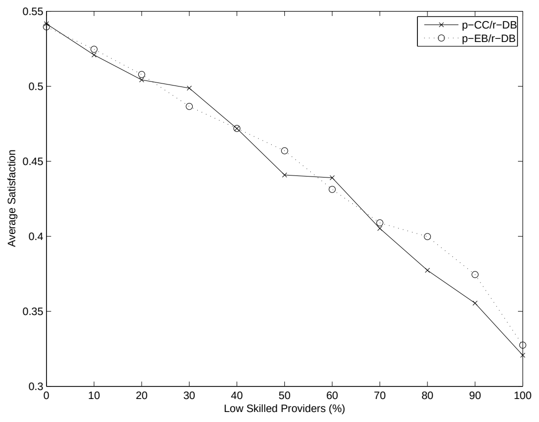
FIGURE 9.14: Pairwise comparison of p-ACS with p-EB using all requester selection strategies in terms of the agents' average satisfaction.

strategies).

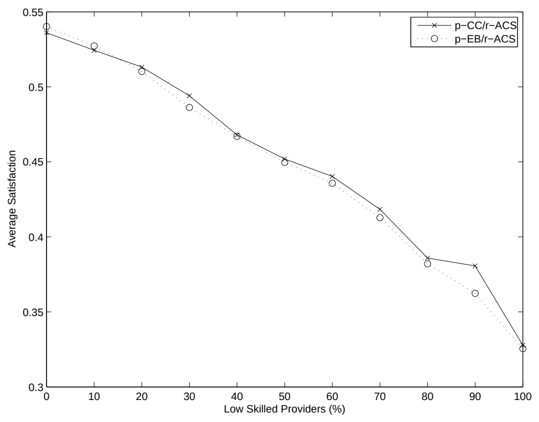
Comparing the performance of p-ACS and p-EB in terms of average satisfaction, p-ACS achieves *higher average satisfaction* when competing in the same population as p-EB, when low-skilled providers are more than 30% of the population, using all requester



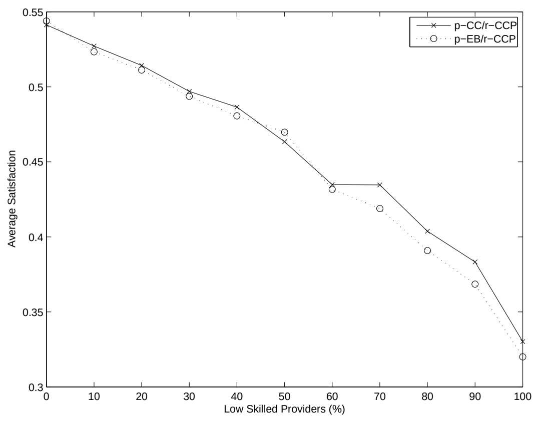
(a) p-CC/r-SR and p-EB/r-SR



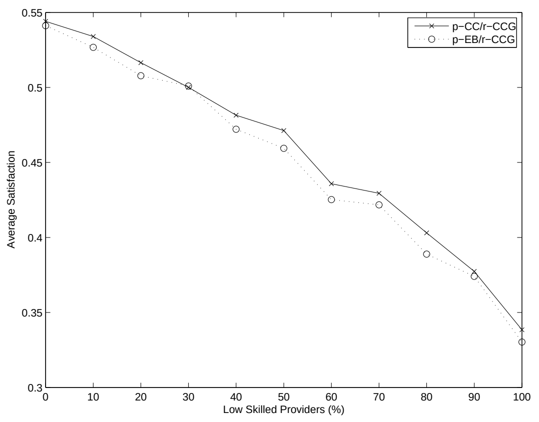
(b) p-CC/r-DB and p-EB/r-DB



(c) p-CC/r-ACS and p-EB/r-ACS



(d) p-CC/r-CCP and p-EB/r-CCP

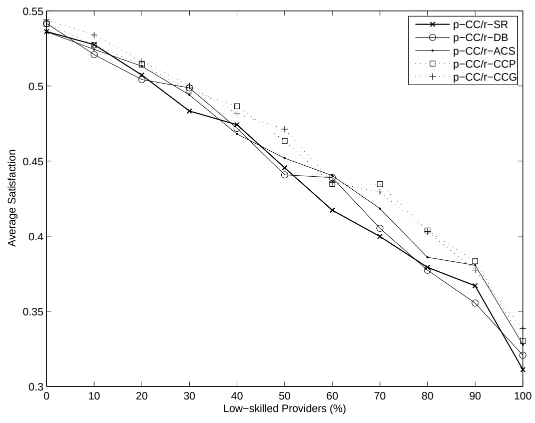


(e) p-CC/r-CCG and p-EB/r-CCG

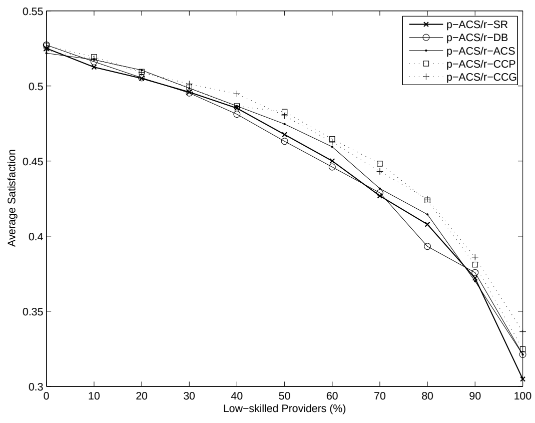
FIGURE 9.15: Pairwise comparison of p-CC with p-EB using all requester selection strategies in terms of the agents' average satisfaction.

selection strategies except r-DB (with which p-ACS achieves similar performance to p-EB), as shown in Figure 9.14. For the pair of strategies p-CC and p-EB, p-CC has similar performance to p-EB across all requester selection strategies, as shown in Figure 9.15.

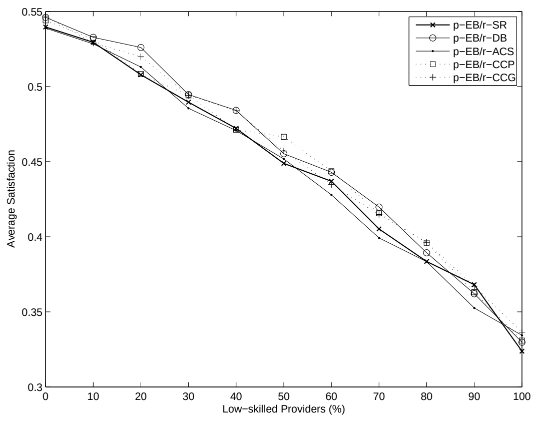
Therefore, of the provider selection strategies in Hypothesis 2, p-ACS performs better than p-CC in maintaining higher average satisfaction than a strategy using pure service evaluation, p-EB. Even though p-ACS does not consider the quality of service directly when selecting providers, but indirectly through the balance of exchange values, it is able to select those agents providing better quality services.



(a) Average Satisfaction for p-CC



(b) Average Satisfaction for p-ACS



(c) Average Satisfaction for p-EB

FIGURE 9.16: Influence of requester selection strategies on each provider selection strategy regarding average satisfaction.

The influence of requester selection strategies on the performance of provider selection strategies p-ACS, p-CC and p-EB is detailed in Figure 9.16. It shows that, as the number of low-skilled providers increases, provider selection strategies p-ACS and p-CC have better performance in terms of average satisfaction when using requester selection strategies r-CCP and r-CCG, which consider service quality. Importantly, p-EB is not influenced by requester selection strategies because it uses service evaluation as the selection criterion, which is not influenced by the choices of a requester selection strategy, as opposed to exchange values which are used by p-ACS and p-CC. Thus, of the requester selection strategies in Hypothesis 3, only r-CCP and r-CCG, which take into account service quality in addition to reciprocity, facilitate the selection of available providers offering good quality services, bringing higher average satisfaction for requesters, but only for the provider selection strategies p-ACS and p-CC.

9.7 Conclusions

In this chapter we have presented an empirical study to evaluate and compare the efficiency of provider and requester selection strategies used by agents in a cooperative system with service variety and resource constraints. The efficiency of provider selection strategies is related to their ability to find providers that perform good quality services, and that are more likely to accept requests. Similarly, the efficiency of requester selection strategies is related to their ability to improve chances of future interactions and their quality when there are resource constraints. These abilities were evaluated through two performance measures: the total number of interactions achieved; and the average satisfaction with received services.

We have argued, in Chapters 7 and 8, that in a non-monetary cooperative system requesters must take into account reciprocal relationships with providers to find those more likely to accept requests, since when providers are busier or have few available resources, agents requesting services need to compete for available providers. By comparing the total interactions achieved by agents using provider selection strategies considering reciprocation with those not taking into account reciprocation, in Experiments 1 and 2 (Sections 9.5.1 and 9.5.2), we showed that, indeed, provider selection strategies that consider reciprocation allow agents to find available providers easily (since they achieve a higher number of total interactions). This ability is more evident when providers have limited resources, as shown in Experiment 2.

In addition to finding providers more likely to accept requests, provider selection strategies must identify those providers offering good quality services. Strategies that consider quality of service are more likely to have higher average satisfaction. However, the relative emphasis on the balance of exchange values (represented by p-ACS) as opposed to both service evaluation and reciprocal relationships (represented by the p-CC strategy) is important to understand and determine in order for cooperation to be effective. Results in Experiments 3 and 4, in Sections 9.6.1 and 9.6.2, show that the combined criteria p-CC strategy performs well in terms of average satisfaction, and no worse than a pure evaluation-based strategy. More importantly, the exchange values-based p-ACS performs *better* than the purely evaluation-based strategy (p-EB) because, by balancing reciprocity and service quality, p-ACS can identify providers that simultaneously provide good services and are more likely to cooperate.

Moreover, achieving more efficient and effective cooperation depends not only on the provider selection strategy but also on the requester selection strategy. For example, the pure reciprocation *dependence-based* requester selection strategy (p-DB) has a positive influence on provider selection strategies, improving their performance in terms of total interactions. Conversely, the *combined criteria* requester selection strategies (r-CCP and r-CCG) improve the performance of corresponding provider selection strategy p-CC in terms of average satisfaction. This demonstrates that it is possible to balance reciprocal interactions with the analysis of the quality of those interactions when providers need to limit requests, in order for agents to operate effectively in heterogeneous cooperative systems.

In summary our experiments have demonstrated the following key aspects.

- Provider selection strategies that identify reciprocal relationships are able to find providers more likely to reciprocate than those strategies using service quality alone, and are necessary for busy providers with resource limits or an increasing number of requests.
- Provider selection strategies that identify reciprocal relationships, in addition to service quality, through exchange values, are able to find providers that are, at the same time, more likely to reciprocate and perform the best services, resulting in agents with high satisfaction with received services. Such strategies are necessary when agents operate in systems with many low quality services.
- Requester selection strategies that use dependence alone increase the number of interactions in which agents engage, thus receiving more services. However, in extreme situations in which providers have minimum resource capacities, dependence alone is not enough; an efficient requester selection strategy requires a combination of dependence and exchange values as selection criteria.

Chapter 10

Conclusions

In this thesis, we have proposed models, methods and mechanisms with the aim of supporting effective cooperation and partner selection for computational entities operating in open cooperative systems with free services. In this chapter, we summarise the key characteristics of each proposed model, method and mechanism, and highlight the research contributions resulting from this thesis. We also discuss the limitations of our approaches, and explore possible extensions and alternative applications. With this objective, the chapter starts with a thesis summary in Section 10.1, followed by research contributions in Section 10.2, and limitations in Section 10.3. Future work is presented in Section 10.4, and we finish with concluding remarks in Section 10.5.

10.1 Thesis Summary

The work in this thesis is contextualised in the area of open cooperative distributed systems, in particular, systems in which services are provided free of charge. Computational entities operating in such systems often have different capabilities and interests, and need to cooperate with each other to achieve individual or common goals. Here, the challenges imposed on cooperations are that services are dynamic and provide different levels of quality, and service providers often have bounded resources and may not always be willing to cooperate.

For cooperation to be efficient in this context, entities requesting services must select among providers, first so that the services they receive have good quality, and second so that they find available providers quickly. In addition, entities providing services must have a non-monetary incentive for cooperation, and decide whether to cooperate with others, such that they can manage their computational resources sensibly, improve their chances of future interactions, and terminate cooperations that do not bring any benefit. Given this, we can consider the computational entities that make up such distributed systems as agents interacting in a multi-agent system.

Although there are existing mechanisms in the multi-agent systems and distributed systems literature that address cooperation and partner selection, as discussed in Chapter 2, they have limitations in dealing with specific characteristics of domain applications and of open systems, or do not provide a combined solution for achieving both efficient and cooperative behaviour in the context of open non-monetary cooperative systems. Therefore, we propose in this thesis a set of models, methods and mechanisms that are combined to support effective cooperation in open systems with free services. More specifically, we address the problems of incentivising non-monetary cooperation and of partner selection in open cooperative contexts.

We started by identifying, in Chapter 3, an application scenario in the bioinformatics domain, aiming at a concrete case study. The choice for bioinformatics in particular has dual sides. First, due to the inherent characteristics of service variety in bioinformatics, in which most services are available free of charge and data is inter-related, the benefits that open cooperative distributed systems can bring to this domain are promising (Stein, 2002; Foster, 2005). However, these same characteristics offer additional challenges for developing agents capable of efficient cooperative behaviour.

Indeed, in Chapter 3 we identified in more detail the specific issues that need to be address so that agents achieve efficient cooperative behaviour in open cooperative systems with free services, and presented an architecture with four components: a *framework for non-monetary cooperative interactions*, which provides non-monetary incentives for service provision and a means to analyse cooperations; an *evaluation method*, which is responsible for evaluating dynamic services; a *requester selection mechanism*, which is responsible for decision-making over service provision, and uses information on cooperative relationships and their properties; and a *provider selection mechanism*, which is responsible for decision-making over service requests.

The evaluation method for dynamic services, proposed in Chapter 5, aims at measuring the properties of different services, so that agents have a means to identify those with better properties according to their preferences. To achieve this, we developed a general evaluation method using evaluation attributes, result measures for those attributes, and evaluation functions. Here evaluation must be undertaken from the different perspectives of providers and requesters, with agents using the evaluations generated by our method to select among alternative interaction partners. The proposed evaluation method was demonstrated through application in real bioinformatics services (MS/MS services) used for protein identification.

In addition to an awareness of service quality, efficient cooperative behaviour in open systems also requires that providers have non-monetary incentives to cooperate and that agents have a means to analyse the cooperations in which they engage. In response, in Chapter 6, we described a computational framework for non-monetary interactions among agents, based on Piaget's theory of exchange values, but addressing the limitations of Piaget's theory to provide a computational representation of exchange values, and a computational model that determines how exchange values are acquired, stored, and spent by agents over interactions. As part of the computational model, we use a valorisation system to determine credits and debts, combining an agent's objective evaluation of a service (through the evaluation method) with subjective evaluations of that service (through subjective influences). Practical examples of cooperative interactions that show how exchange values can be determined from service evaluation, with agents providing and requesting protein identification services, demonstrated the applicability of the framework.

Based on this, in Chapter 7 we described a provider selection mechanism for agents requesting services. The objective here is to select providers that are, at the same time, more likely to provide good services, and more likely to accept a request, given that they are not obliged to do so. As part of the provider selection mechanism, therefore, we developed an algorithm to select among alternative service providers, and a group of strategies to instantiate it, using quality-related selection criteria (based on previous service evaluations and the balance of exchange values), and persuasion-related selection criteria (based on reciprocal relationships among requester and providers, including the credits, debts and service dependence among them). By combining information on service evaluation and reciprocal relationships, it is possible to develop a selection mechanism that can help service requesters to find providers of good quality services without spending too much time doing so.

Similarly, in Chapter 8, we developed a request selection mechanism, which aims not only to support flexible decision making over whether to accept requests, but also to allow agents to manage their resources. We developed an algorithm for dynamic decision-making among incoming requests, and a group of strategies to instantiate this algorithm, establishing preference orderings for requesters according to different combinations of the selection criteria defined earlier.

For both of these selection mechanisms, we carried out an empirical study to test their applicability and efficiency, in Chapter 9. In particular, we simulated a cooperative system, with heterogeneous services, and with resource-constrained providers. We compared our proposed provider and request selection strategies and showed that agents using reciprocation-based provider selection strategies (using exchange values or dependence) are able to find available providers much quicker than those not doing so (using evaluation alone or at random). Moreover, our results showed that, when considering both service quality, through the balance of exchange values, and reciprocal relationships, agents using the *exchange values-based* provider selection strategies not only achieve high proportions of successful interactions and high average satisfaction with received services, but they are also better than the purely evaluation-based provider selection strategy. This is because, by balancing reciprocal relationships and service quality, provider selection strategies can identify providers that are, at the same time, providing good services and more likely to cooperate. For requester selection, we showed that by considering reciprocal relationships, through a combination of dependence and exchange values, agents can indeed improve their chances of future interaction by increasing the performance of provider selection strategies in finding available providers quicker. Also, our proposed *combined* strategy with a preference for paying debts not only helps to improve the chances of future interactions but also improves the quality of those interactions. More importantly, we showed that it is possible to improve the quality of interactions even when providers need to limit requests.

10.2 Research Contributions

The key contribution of this thesis is in the efficient request and provision of services in open cooperative systems with unpaid services. This is achieved through a set of solutions ranging from the evaluation of services to non-monetary incentives for cooperation. In this context, the general contributions are: a novel computational framework to support and incentivise non-monetary cooperative interactions among self-interested agents; a provider selection mechanism based on social techniques and service evaluation; and a request selection mechanism for agents providing unpaid services and with resource constraints. In what follows, we examine these contributions in more detail.

10.2.1 A Computational Framework for Non-monetary Cooperative Interaction

To develop autonomous agents capable of operating in open cooperative systems with unpaid services, agents providing services need some *incentive to cooperate*, and agents requesting services need some means to avoid relying on altruism to guarantee service provision. Although there are approaches in the literature that use non-monetary incentives for cooperation, such as brownie points (Glass and Grosz, 2000) and reciprocation ensured by norms and contracts (Dignum and Dignum, 2001; Dignum, 2003), these do not apply to the kind of systems we focus on in this thesis.

In response, we have developed a framework for non-monetary exchanges among self-interested agents, through a computational model of Piaget's theory of exchange values, in which the motivation to cooperate comes from acquiring, accumulating, and spending (non-monetary) credits and debts that result directly from interactions. This work can be broken down into the following specific contributions.

- **A computational model of Piaget's theory of exchange values** to support non-monetary cooperative interactions between self-interested agents, and sufficiently general to apply to different domains. More than merely providing a means for motivating and modeling cooperations, the model enables agents to make decisions on whether to maintain or terminate cooperative interactions, through exchange values and their balances.

- **A system of credits and debts** that represents the provision-reciprocation cycle in which non-monetary credits and debts are accumulated and spent, yet is compatible with self-interested behaviour. This provides the basic incentive for non-monetary cooperative interactions among autonomous agents.
- **A valorisation system** that considers the evaluation of services provided or received during an interaction according to the individual objective and subjective perspectives. Here, objectivity captures possible differences in service results, while subjectivity captures possible influences on objective values caused by the individual agent behaviour or social relationships among agents. By capturing such individual perspectives over services, our valorisation system copes with diverse interaction partners that agents may encounter in open systems, and thus provides a means for agents to better select the interactions in which they engage.
- **A computational representation of subjective influences** that may alter an agent's objective evaluation of a service. In systems in which interactions depend on reciprocal relationships among heterogeneous agents, it is likely that different social relations will develop, influencing objective evaluations of services either to improve the chances of future interactions or to improve the value of an interaction. These subjective influences thus enable the modeling of heterogeneous social behaviour and the reasoning about such heterogeneous partners.

Importantly, the system of credits and debts and the valorisation system are *dynamic*, in the sense that they manage and provide dynamic information on reciprocal relationships, objective and subjective service evaluation, and on the outcome of interactions in terms of compensation of provided services.

10.2.2 An Evaluation Method for Dynamic Services

In the service-oriented literature, some methods that address the evaluation of dynamic services have been proposed (such as (Day and Deters, 2004; Casati *et al*, 2004; Sun *et al*, 2006)), but they typically evaluate only a single aspect of a service or they require pre-defined quality levels for service results. In this thesis, we have proposed an evaluation method for dynamic services that evaluates their properties without requiring any information on *expected* or *ideal* results, and generates consistent evaluations that can be compared at different points in time. In particular, our method provides a means for agents to evaluate dynamic services and to use these evaluations to subsequently select among alternative services. Moreover, it allows for personalised definition of evaluation functions, in the sense that agents can be designed to be more or less strict in their evaluations, according to need or purpose. Our technique is general enough to allow the evaluation of services from the perspective of both service providers, which evaluate services in terms of their cost, and service requesters, which evaluate services in terms of their satisfaction. It can thus be used to support the selection processes for both requesters and providers, in order to make better decisions over alternative, diverse interaction partners, and in domains in which expectations or idealisations are difficult to specify, such as services in the bioinformatics domain.

10.2.3 Provider Selection

In most open cooperative systems, selecting providers is important when agents requesting services find alternative providers for the same service but offering services with different levels of quality. At the same time, finding providers that are more likely to cooperate is indispensable when they are not obliged to accept requests and may have to limit the services they provide due to resource constraints. Existing mechanisms for provider selection consider either information on service properties (such as service evaluation and similarity) (Casati *et al*, 2004; Caverlee *et al*, 2004) or cooperative relationships between providers and requesters (Sichman *et al*, 1994; David *et al*, 2001), and there is no attempt to balance these alternatives in a single provider selection mechanism. Moreover, these mechanisms typically assume that providers always accept requests, and thus do not consider the possibility of not finding an available provider.

To address these limitations for agents operating in open cooperative applications, we have developed a provider selection mechanism that allows agents to take advantage, *at the same time*, of service variety, by identifying service providers with better properties, and of reciprocal relationships, as a way to find available providers quicker. Within this, we have also specified different provider selection strategies that focus on particular selection criteria or on a combination of different criteria, providing flexible decision-making for agents operating under different environmental conditions (such as the number of participants or low quality services in the system) and social conditions (such as the service dependencies among agents). These strategies have been tested and compared to demonstrate their effectiveness in different contexts.

10.2.4 Request Selection

Although autonomous agents providing services in open systems are not obliged always to accept requests, in a non-monetary cooperative environment they may need to provide services to improve their chances of receiving services from others in the future. Also, they may need to prioritise incoming requests in case of scarce resources. The challenge here is to balance the number of accepted requests against the availability of resources, so that the chances of future interactions are improved and beneficial cooperations are maintained, but without being overwhelmed by services to perform. However, existing decision-making mechanisms in the literature to decide whether to cooperate (Saha *et al*, 2003; Banerjee *et al*, 2005) do not consider resource limitations on the provider's side, nor the success of cooperative relationships in terms of the balance between provided and received services. Yet this is important if the environment has agents with different preferences and skills that reciprocate, but by providing low quality services.

In this thesis, we have developed a requester selection mechanism to support decision-making over service provision in resource-constrained cooperative applications to deal with the above problems. Experimental results show that our proposed requester selection strategies that combine information on exchange values and service dependence indirectly *help to improve the agents' performance* in terms of the number of interactions they achieve and the quality of those interactions. Thus, our request selection mechanism contributes to efficient and flexible decision-making for providers with resource constraints operating in a cooperative context, such that agents improve their chances of future interactions while managing their resources sensibly.

10.2.5 Metrics for Protein Identification Services

Outside the domain of computing, in bioinformatics, we have also made a distinct contribution. Specifically, although benchmarks for MS/MS services, used in protein identification experiments, have been presented in the literature (Chamrad *et al*, 2004; Kapp *et al*, 2005), they are limited in application to real protein identification experiments. In particular, they typically do not consider performance-related aspects of MS/MS services, are based on analyses of biological data with known identity (as opposed to unidentified data), and are static in that they only evaluate services once. We have addressed these limitations by identifying evaluation attributes and defining evaluation functions for MS/MS services, directed at unknown biological data and considering data and performance-driven aspects. Our evaluation method for MS/MS services thus contributes to the efficient automation of *in-silico* protein identification experiments, so that such experiments can improve in quality once the properties of alternative services are known (and the best can be selected).

10.3 Limitations

While we have made substantial contributions through the various models and mechanisms described above, there are still some limitations, as discussed below.

- **Evaluation of previously unused services**. Although the evaluation method proposed in Chapter 5 provides accurate information about services that were received in previous interactions, it only helps the selection process if there are previously used services among those available. In cases where there has been no prior interaction with (or use of) a service, some combination of our evaluation method with a reputation-based evaluation system, such as (Sabater and Sierra, 2001; Teacy *et al*, 2005), might be needed. Reputation essentially provides trusted information from third parties on services that they have used perviously, which is valuable in allowing others to evaluate services they have not yet encountered.

- **Extensive list of possible providers**. The provider selection mechanism loses performance when the list of possible providers is extensive. Since partner selection is based on information generated by an agent's direct experience with others, a large number of potential partners may excessively delay the ability to differentiate among them. In this case, identifying those providing better services may simply take too much time. As before, to acquire information on all potential providers (for both evaluation and reciprocal relationships), could be addressed by using trust and reputation mechanisms, such as (Sabater and Sierra, 2001; Teacy *et al*, 2005), for using information from third parties. Such reputation systems address the potential uncertainty and veracity of third party information and, in doing so, complement mechanisms based on direct experience, such as those proposed here. However, it is an entirely separate topic and is outside the scope of our work.

- **Multiple evaluation scales**. In our proposed framework for non-monetary cooperation we use a fixed scale of $[0, 1]$ to represent and determine exchange values, compatible with our evaluation method. However, agents may evaluate services and determine their exchange values according to different scales. If so, then they need to agree upon a common scale to use in the interaction, so that they make sense of the valorisations they receive from others. This would require that service evaluations (both objective and subjective) used to determine exchange values must be mapped to some common scale, with a means to reach agreements over which scale to use, possibly through negotiation and argumentation techniques. Independent of the solution, alternative scales must not interfere in the *perception* that an agent has of a service. For example, if an agent receives a service s_1 and the evaluation of this service on a scale of $[0, 1]$ indicates that s_1 is a poor quality service, the evaluation of s_1 on a scale of $[0, 10]$ should also indicate that the service has poor quality.

- **Cooperation among a group of agents**. The functionalities of our proposed framework for non-monetary interactions based on exchange values apply to pairs of agents rather than groups of more than two. Indeed, in our model, these values are not transferable from one agent to another. This means that, if agent A has a debt with agent B and agent B has a debt with agent C, agent B could not ask agent A to pay its debt by performing a service for agent C. Yet, in principle, such transferable credits and debts might be useful when agents work in a team or form a coalition to take advantage of complementary skills. To account for transferable credits and debts would require different ways of determining credits and debts and different ways of analysing cooperations through exchange values (since it would involve comparing exchange values acquired in interactions with different partners).

10.4 Further Work

In addition to the limitations above, there are also several possible extensions and alternative applications of our work.

1. **Adjustable evaluations for dynamic systems.** When agents analyse the result of their cooperations, they should decide not only to continue a successful cooperation and terminate an unsuccessful one, but also to adjust their evaluations if unsuccessful interactions result from strict application of evaluation requirements, or if successful interactions result from relaxed evaluation standards. Although our evaluation method allows for individual evaluation functions in this way, with more or less strictness on service evaluations, we assume that this is pre-defined by an agent to reflect its preference for standards of evaluation. However, in some cases it may be useful to determine this strictness dynamically. For example, if the quality of the services available in the system improves, and providers previously considered to be offering average services are subsequently considered to provide poor services in comparison to the majority of high quality services, agents might tune their evaluation strictness to reflect this change. This would be valuable for improving the accuracy of evaluations in a dynamic environment. Moreover, agents may evaluate average services as being too low or too high, if they are either too strict or to relaxed in their evaluation functions, and this can impact on future interactions. In particular, an agent might terminate a cooperation with a partner that actually provides good services as a result of too strict a level of evaluation. Addressing this would require the development of a *meta-evaluation* mechanism to operate on top of our proposed evaluation method, allowing agents to tune their evaluation standards when necessary.

2. **Enforcing reciprocation with normative systems**. At present, when agents do not pay their debts, they are devalorised by the creditor, decreasing their chances of future interactions with it. Reciprocation is thus not imposed, but results from individual agent decisions. However, it may be useful in some contexts to enforce the debtor's commitment by establishing explicit norms and rules for the reciprocation process. This would require combining our framework with a normative system, such as (Lopez y Lopez *et al*, 2005). Such a system would not

interfere in the process of determining exchange values, but only act on the transition from the provision to the reciprocation stage of the interaction by enforcing the commitments resulting from credits and debts.

3. **Improving cooperation in complex scenarios through negotiation.** In the current work, we do not consider any formal commitments among requester and provider regarding expected properties of the service to be delivered. However, more complex scenarios might demand the establishment of explicit constraints on the provided or reciprocated service through contracts and negotiation. For example, in a time-constrained scenario, interacting agents could establish a contract specifying the deadline by which a creditor should receive a reciprocated service, and additionally, the sanctions if a debtor does not meet the deadline (such as the debt doubling in value). Negotiation could also be used after an unsuccessful cooperation to address a perceived imbalance in values by, for example, negotiating for an extra service as a precondition for maintaining the cooperative relationship. In a different context, negotiation could be used to enable agents to reach agreement over a common scale for determining exchange values (as discussed above). Adding a negotiation mechanism to our current architecture does not require any changes to the framework, since exchange values and evaluations are still determined in the same way. However, the commitments and sanctions established in contracts, and whether they are violated, may influence credits and debts, in which case these influences must be represented just as other subjective influences.

4. **Pro-active behaviour.** In a reciprocation-based cooperative environment in which requesters need a persuasive element (such as service dependence, and credits and debts) to get priority in a provider's incoming requests, pro-active behaviour may be necessary if the selection mechanism does not identify any suitable provider. In this case, an agent could pro-actively offer services to potential providers in order to build up cooperative relationships and improve the subsequent chances of interacting with these providers. This would entail an extension to our agent architecture to include a mechanism to support such pro-active behaviour towards interactions. A mechanism of this kind would need to be efficient in identifying providers to send service offers, in order to avoid choosing those that do not reciprocate or that reciprocate by providing low quality services. Pro-activeness in this way would complement the functionalities of both the provider and request selection mechanisms, in support of greater opportunities for interaction in open cooperative systems.

10.5 Concluding Remarks

Many of today's important scientific discoveries have only become possible with the introduction of cooperative initiatives to make data and tools, generated by organisations and individuals for personal use, available to global communities through the Internet. One notable example is the domain of genetic research, in which the online availability of several databases of biological data and bioinformatics tools has contributed to the discovery of genes and proteins connected to biological process of living organisms. In turn, the understanding and analysis of the function of such genes and proteins is helping to design more effective drugs for disease control (The Wellcome Trust Case Control Consortium, 2007), and to prevent (and sometimes cure) genetic diseases (Campbell and Heyer, 2002).

As the number and diversity of tools and data that are made available to global communities continues to grow, it becomes increasingly difficult for human users to keep track of such resources manually, hence the need to automate user tasks (such as through *in silico* experiments), so that these tools and data can be used effectively and efficiently. The final goal here is that global cooperative communities are seen as open cooperative systems rich in tools and data, in which tasks are performed efficiently and free of charge to allow further discoveries and progress in the domain of expertise of such communities.

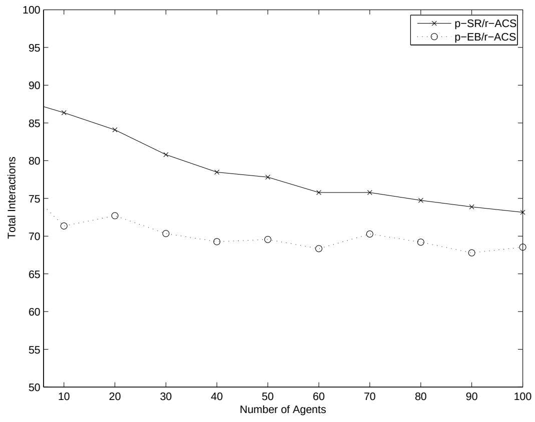
While initiatives for sharing private resources are the first step in creating such open cooperative systems, the second step is the development of infrastructures to support the interoperability and access of diverse data and tools (such as with service-oriented architectures, Semantic Web and Grid technologies). The current challenge is to address the dynamism and diversity of participants and services through the development of computational entities that can operate effectively in such systems. In this thesis, we have focussed on this third step towards developing open cooperative systems rich in data and services, through development of autonomous agents capable of efficient and effective cooperation with others in open systems with unpaid services.

Appendix A

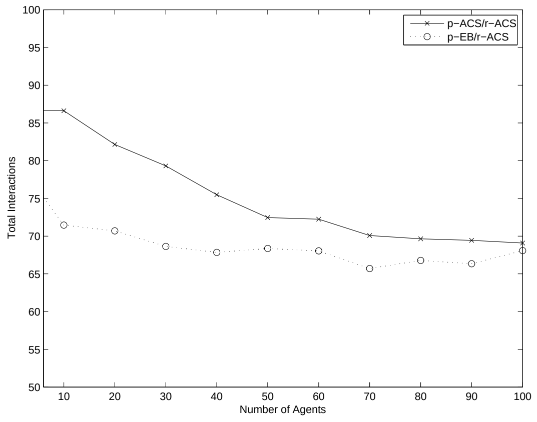
Additional Results

In this appendix, we provide full details of the experiments in Chapter 9, including the full set of graphs.

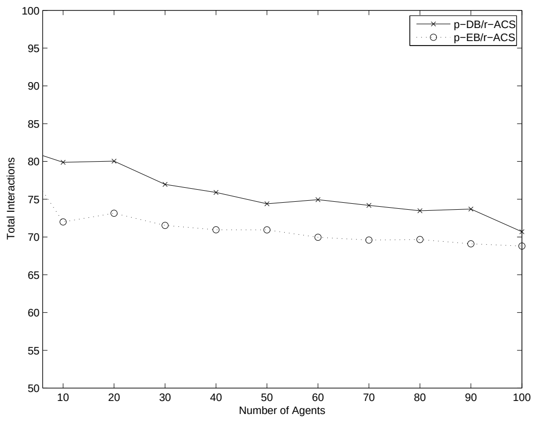
A.1 Experiment 1



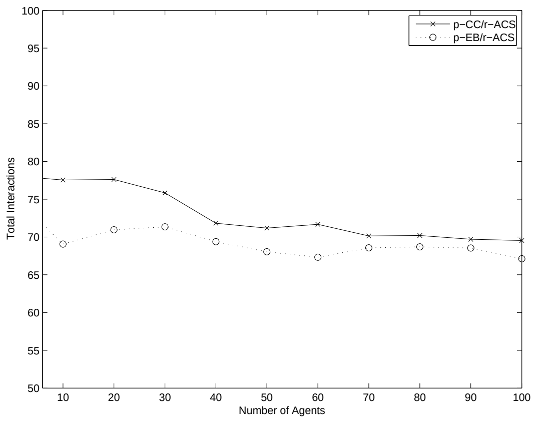
(a) p-SR/r-ACS



(b) p-ACS/r-ACS

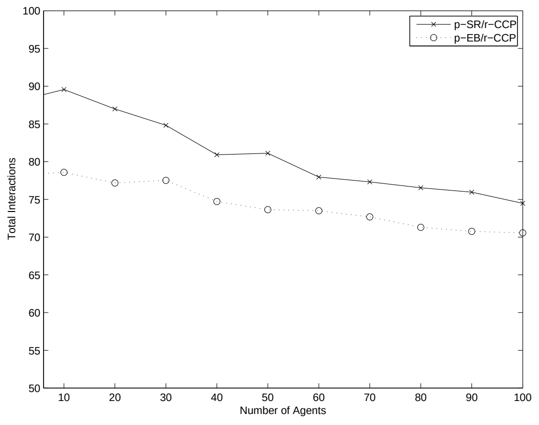


(c) p-DB/r-ACS

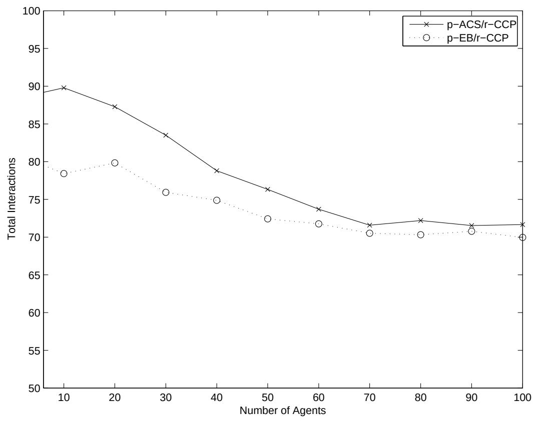


(d) p-CC/r-ACS

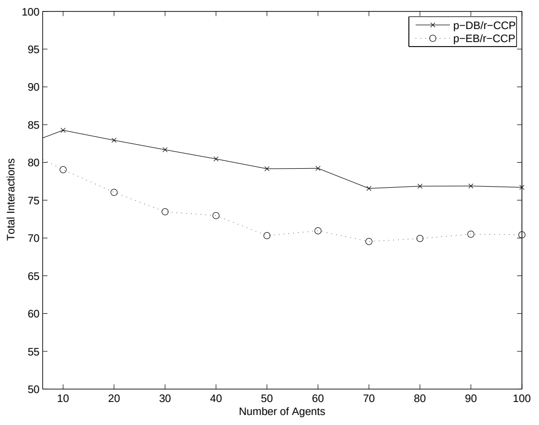
FIGURE A.1: Comparing provider selection strategies when using the r-ACS requester selection strategy.



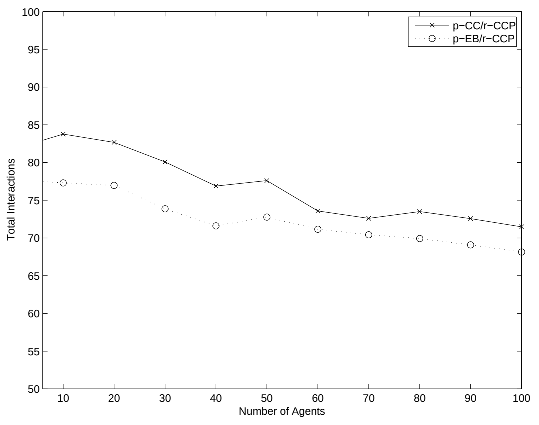
(a) p-SR/r-CCP



(b) p-ACS/r-CCP

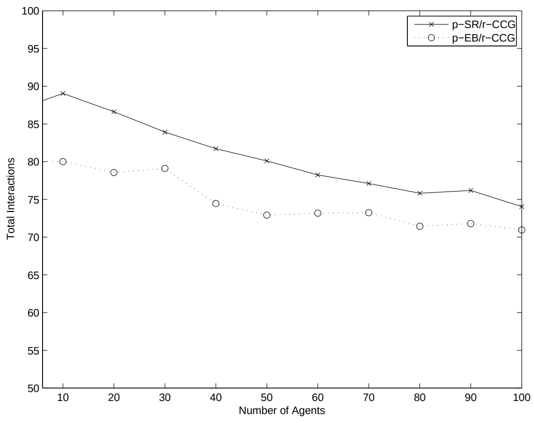


(c) p-DB/r-CCP

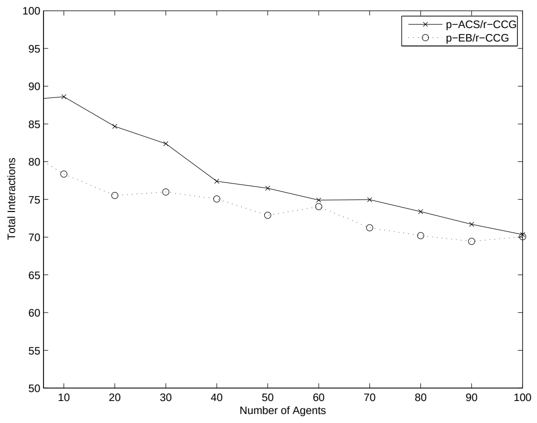


(d) p-CC/r-CCP

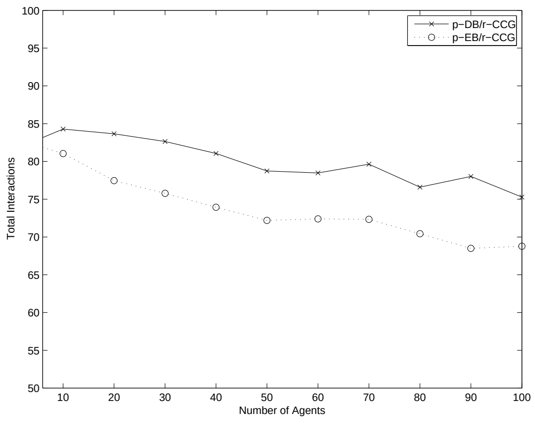
FIGURE A.2: Comparing provider selection strategies when using the r-CCP requester selection strategy.



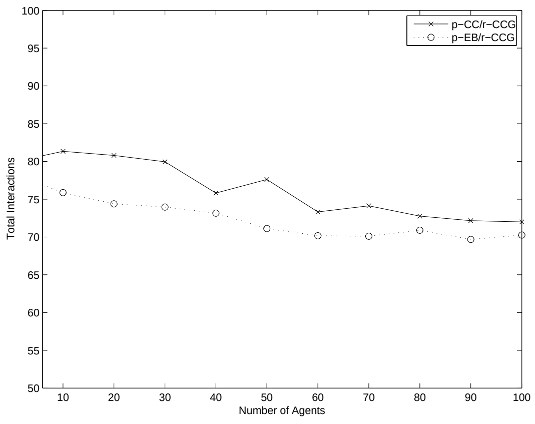
(a) p-SR/r-CCG



(b) p-ACS/r-CCG



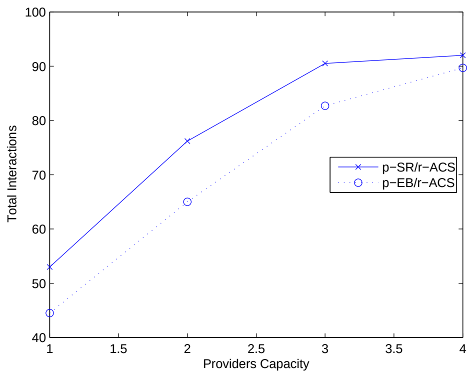
(c) p-DB/r-CCG



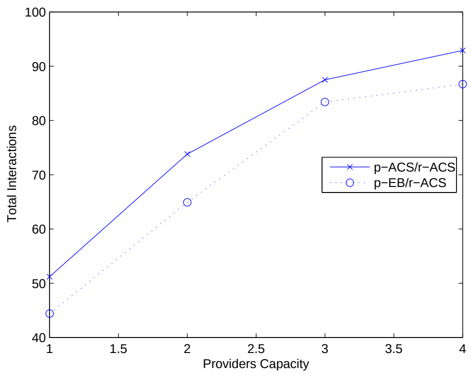
(d) p-CC/r-CCG

FIGURE A.3: Comparing provider selection strategies when using the r-CCG requester selection strategy.

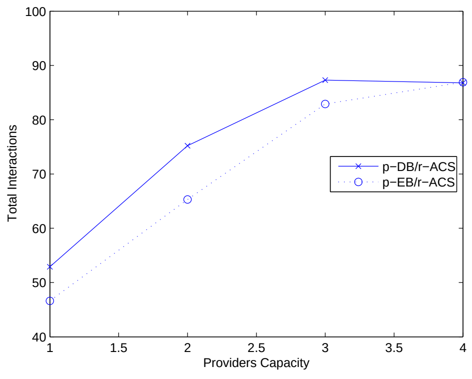
A.2 Experiment 2



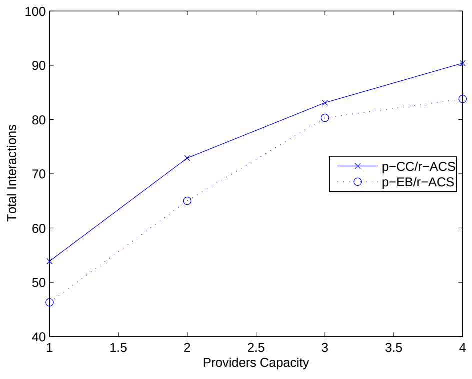
(a) p-SR/r-ACS



(b) p-ACS/r-ACS

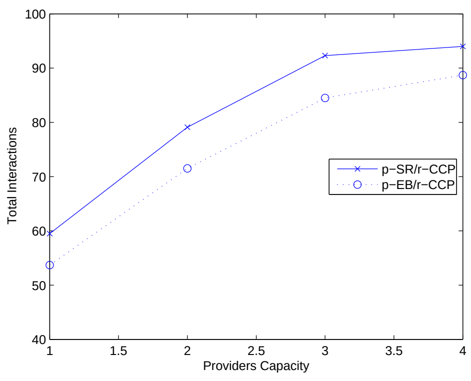


(c) p-DB/r-ACS

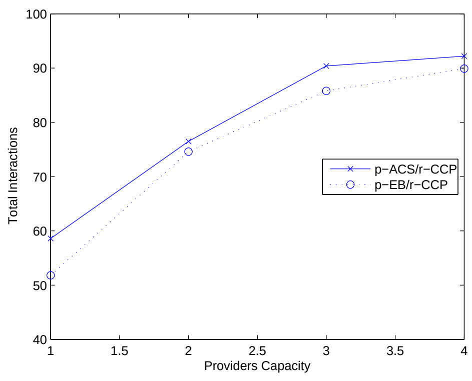


(d) p-CC/r-ACS

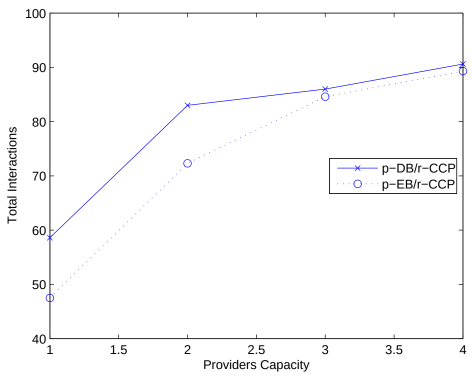
FIGURE A.4: Total interactions for provider selection strategies using the r-ACS requester selection strategy when varying the providers capacity.



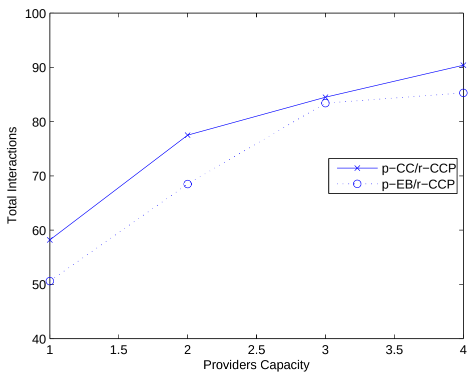
(a) p-SR/r-CCP



(b) p-ACS/r-CCP

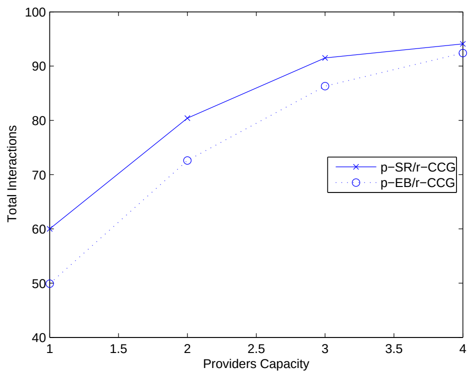


(c) p-DB/r-CCP

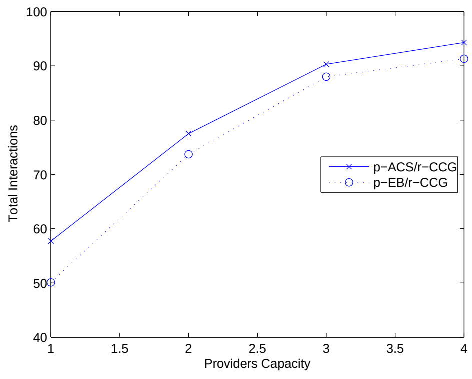


(d) p-CC/r-CCP

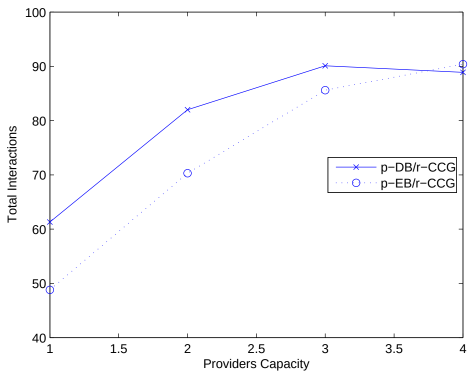
FIGURE A.5: Total interactions for provider selection strategies using the r-CCP requester selection strategy when varying the providers capacity.



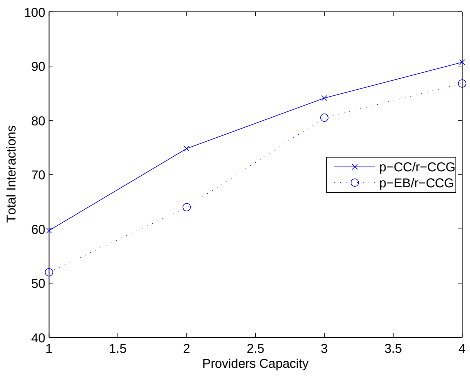
(a) p-SR/r-CCG



(b) p-ACS/r-CCG



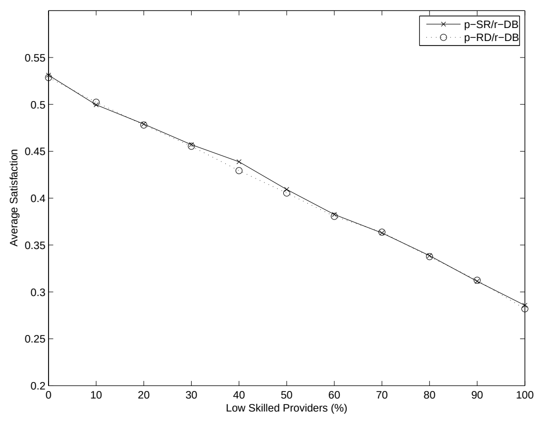
(c) p-DB/r-CCG



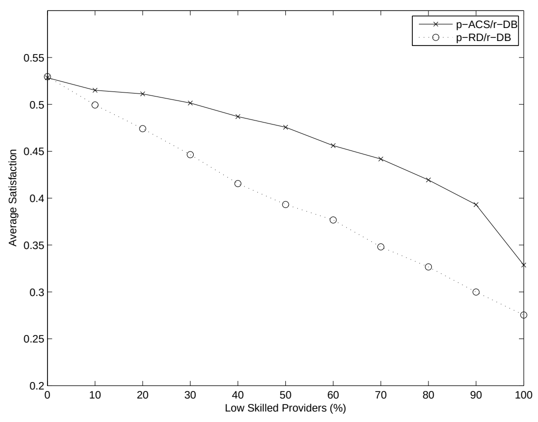
(d) p-CC/r-CCG

FIGURE A.6: Total interactions for provider selection strategies using the r-CCG requester selection strategy when varying the providers capacity.

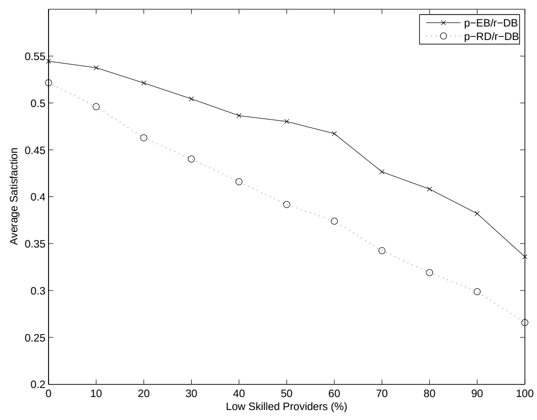
A.3 Experiment 3



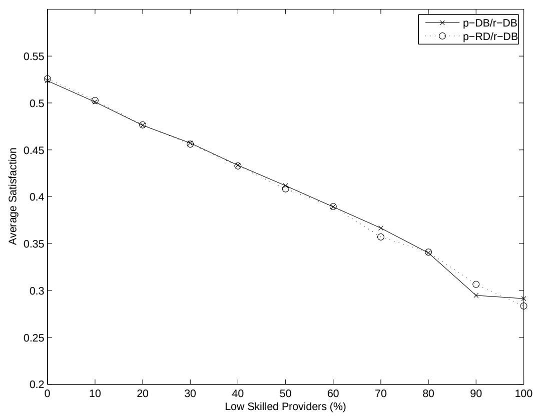
(a) p-SR/r-DB



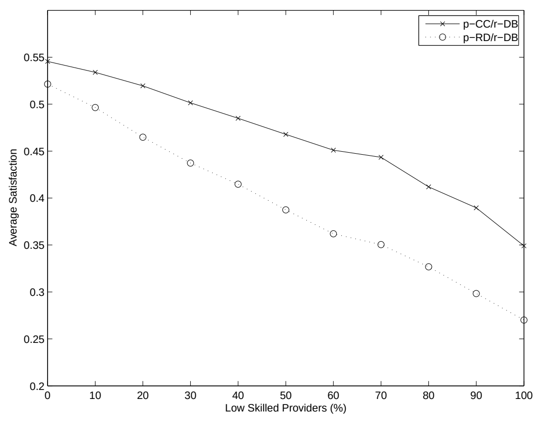
(b) p-ACS/r-DB



(c) p-EB/r-DB

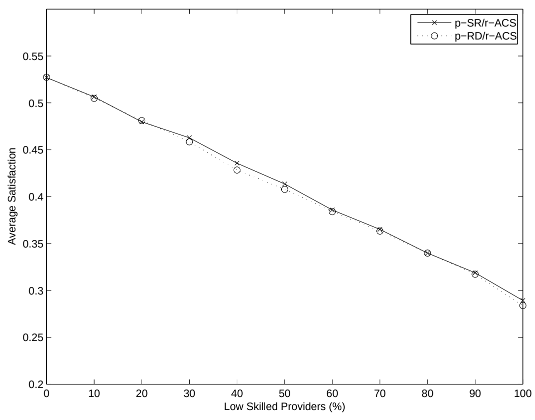


(d) p-DB/r-DB

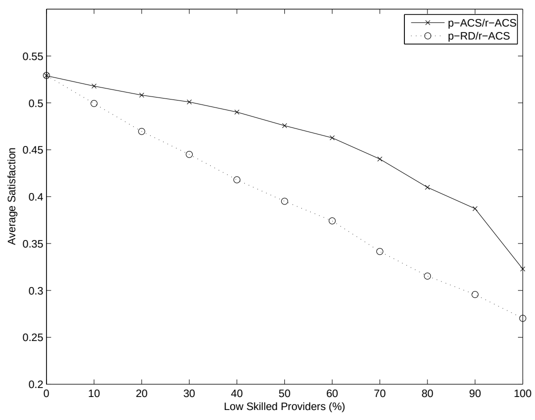


(e) p-CC/r-DB

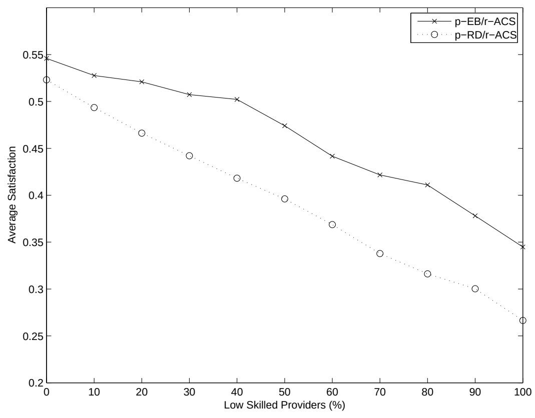
FIGURE A.7: Average satisfaction for all provider selection strategies against the baseline with r-DB requester selection strategy.



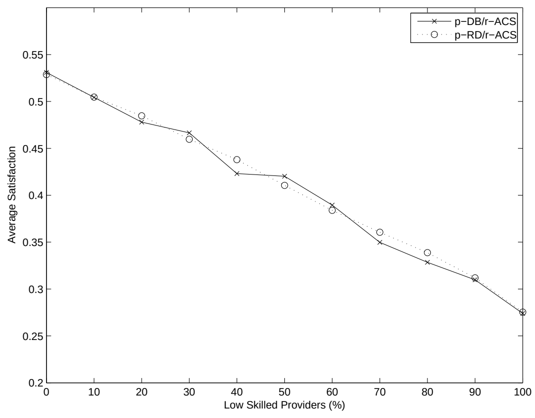
(a) p-SR/r-ACS



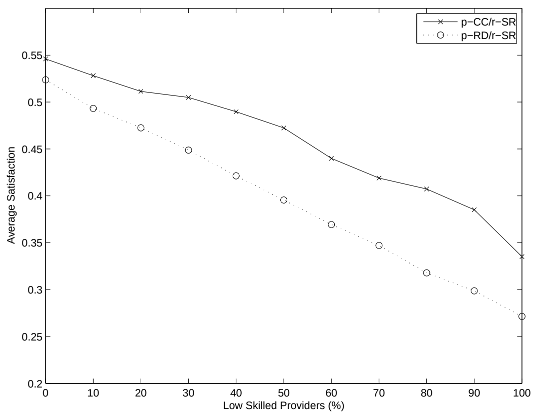
(b) p-ACS/r-ACS



(c) p-EB/r-ACS

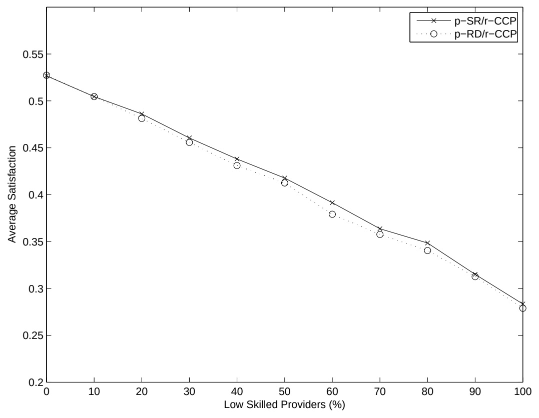


(d) p-DB/r-ACS

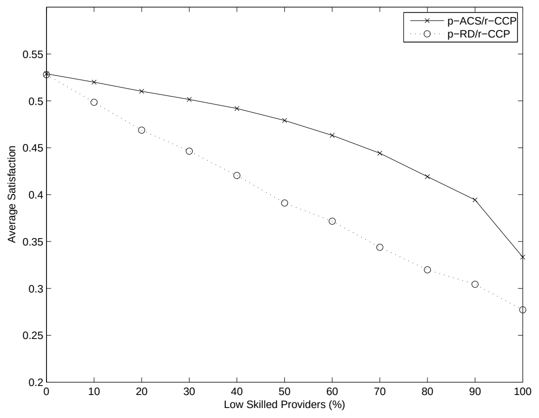


(e) p-CC/r-ACS

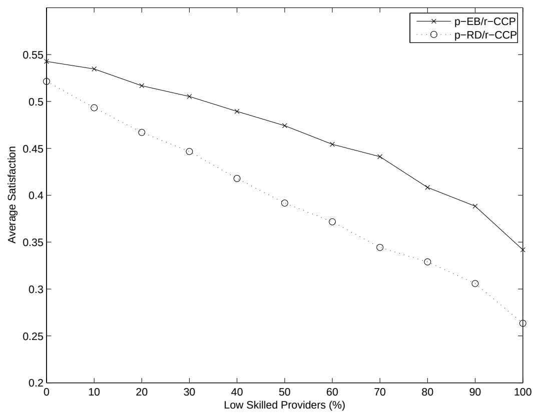
Figure A.8: Average satisfaction for all provider selection strategies against the baseline with r-ACS requester selection strategy.



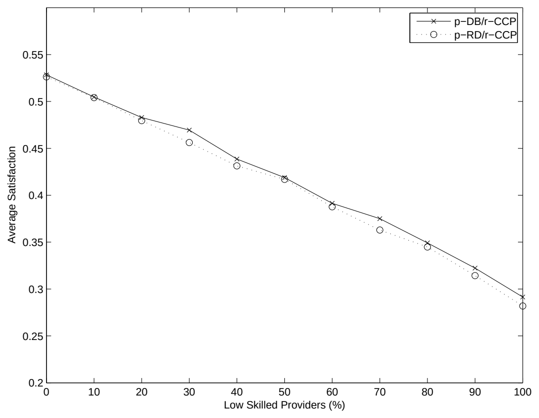
(a) p-SR/r-CCP



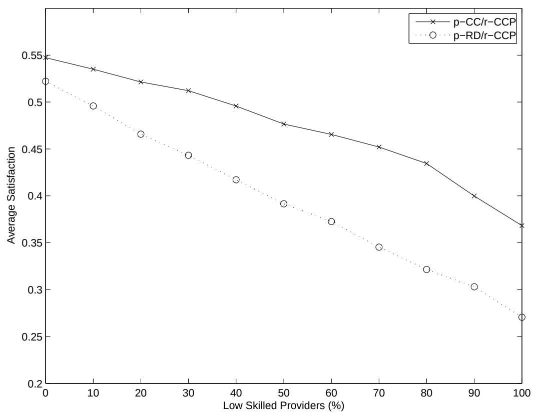
(b) p-ACS/r-CCP



(c) p-EB/r-CCP

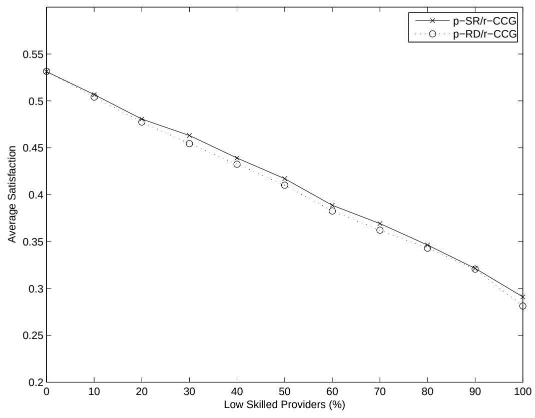


(d) p-DB/r-CCP

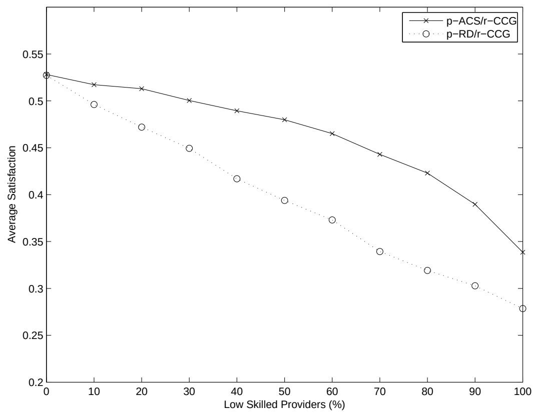


(e) p-CC/r-CCP

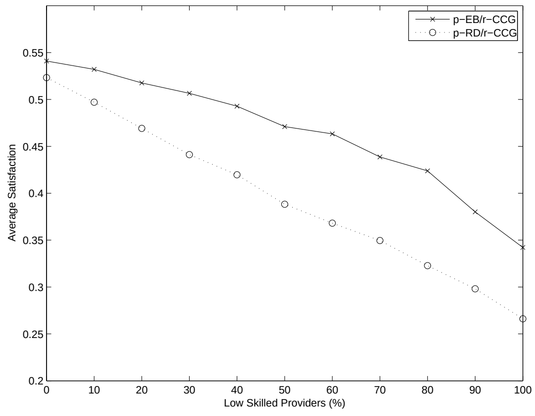
Figure A.9: Average satisfaction for all provider selection strategies against the baseline with r-CCP requester selection strategy.



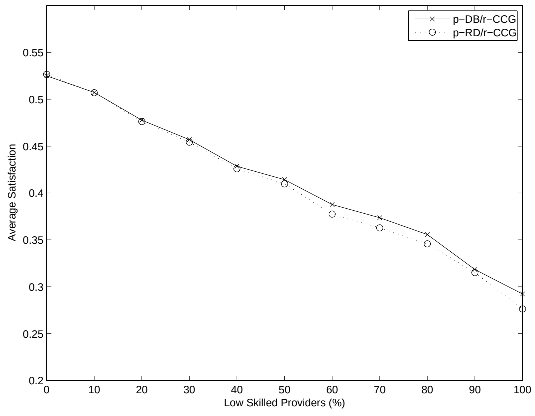
(a) p-SR/r-CCG



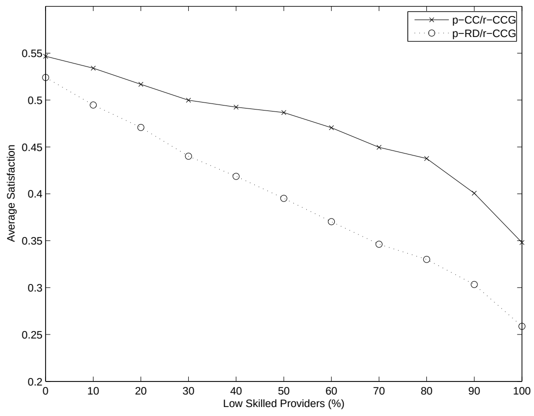
(b) p-ACS/r-CCG



(c) p-EB/r-CCG



(d) p-DB/r-CCG



(e) p-CC/r-CCG

Figure A.10: Average satisfaction for all provider selection strategies against the baseline with r-CCG requester selection strategy.

Appendix B

Significance Results

This appendix provides the statistical significance data for experimental results in Chapter 9. To that end, here we present tables with p-values for the t statistics used to test the difference between means from pairs of strategies.

B.1 Experiment 1

Figure	Strategies	Observations										
		6	10	20	30	40	50	60	70	80	90	100
9.4(a)	p-SR/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9.4(b)	p-ACS/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.33
9.4(c)	p-DB/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9.4(d)	p-CC/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
9.5(a)	p-SR/r-DB	0.00	0.00	0.28	0.00	0.01	0.00	0.00	0.21	0.00	0.00	0.07
9.5(b)	p-ACS/r-DB	0.01	0.07	0.00	0.12	0.41	0.74	0.92	0.68	0.12	0.21	0.67
9.5(c)	p-DB/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9.5(d)	p-CC/r-DB	0.25	0.14	0.07	0.00	0.00	0.00	0.00	0.00	0.01	0.07	0.02
A.1(a)	p-SR/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.1(b)	p-ACS/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.34
A.1(c)	p-DB/r-ACS	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
A.1(d)	p-CC/r-ACS	0.03	0.00	0.00	0.00	0.01	0.00	0.00	0.20	0.18	0.30	0.03
A.2(a)	p-SR/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.2(b)	p-ACS/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.34	0.12	0.50	0.10
A.2(c)	p-DB/r-CCP	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.2(d)	p-CC/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.14	0.00	0.00	0.00
A.3(a)	p-SR/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.3(b)	p-ACS/r-CCG	0.00	0.00	0.00	0.00	0.05	0.00	0.47	0.00	0.00	0.05	0.79
A.3(c)	p-DB/r-CCG	0.20	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.3(d)	p-CC/r-CCG	0.30	0.00	0.00	0.00	0.03	0.00	0.01	0.00	0.15	0.07	0.17

Table B.1: Significance levels for Experiment 1. Strategies are compared against p-EB.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.6(a)	p-DB/r-SR p-ACS/r-SR	0.00	0.46	0.05	0.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-DB/r-SR p-CC/r-SR	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
	p-DB/r-SR p-SR/r-SR	0.09	0.40	0.11	0.75	0.01	0.00	0.00	0.12	0.00	0.00	0.01
	p-SR/r-SR p-ACS/r-SR	0.20	0.15	0.00	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-SR p-CC/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00
	p-ACS/r-SR p-CC/r-SR	0.00	0.00	0.28	0.00	0.09	0.74	0.57	0.46	0.00	0.34	0.27
9.6(b)	p-DB/r-ACS p-ACS/r-ACS	0.00	0.00	0.02	0.01	0.64	0.05	0.00	0.00	0.00	0.00	0.10
	p-DB/r-ACS p-CC/r-ACS	0.00	0.07	0.03	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.25
	p-DB/r-ACS p-SR/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.07	0.18	0.82	0.00
	p-SR/r-ACS p-ACS/r-ACS	0.02	0.77	0.02	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-ACS p-CC/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-ACS p-CC/r-ACS	0.00	0.00	0.00	0.00	0.00	0.23	0.61	0.95	0.61	0.79	0.64
9.6(c)	p-DB/r-DB p-ACS/r-DB	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-DB/r-DB p-CC/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-DB/r-DB p-SR/r-DB	0.12	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-DB p-ACS/r-DB	0.52	0.28	0.03	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-DB p-CC/r-DB	0.00	0.00	0.00	0.07	0.11	0.82	0.25	0.74	0.00	0.37	0.10
	p-ACS/r-DB p-CC/r-DB	0.00	0.00	0.10	0.49	0.17	0.01	0.10	0.07	0.00	0.00	0.14

Table B.2: Significance levels for Experiment 1. Comparing reciprocation-based provider selection strategies.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.6(d)	p-DB/r-CCP p-ACS/r-CCP	0.00	0.00	0.00	0.10	0.18	0.01	0.00	0.00	0.00	0.00	0.00
	p-DB/r-CCP p-CC/r-CCP	0.99	0.63	0.82	0.21	0.00	0.20	0.00	0.00	0.00	0.00	0.00
	p-DB/r-CCP p-SR/r-CCP	0.00	0.00	0.00	0.00	0.68	0.07	0.21	0.51	0.77	0.38	0.01
	p-SR/r-CCP p-ACS/r-CCP	0.61	0.79	0.75	0.18	0.05	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-CCP p-CC/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-CCP p-CC/r-CCP	0.00	0.00	0.00	0.00	0.15	0.28	0.92	0.40	0.28	0.40	0.87
9.6(e)	p-DB/r-CCG p-ACS/r-CCG	0.00	0.00	0.34	0.81	0.00	0.07	0.00	0.00	0.00	0.00	0.00
	p-DB/r-CCG p-CC/r-CCG	0.28	0.01	0.01	0.02	0.00	0.36	0.00	0.00	0.00	0.00	0.00
	p-DB/r-CCG p-SR/r-CCG	0.00	0.00	0.00	0.20	0.54	0.23	0.81	0.02	0.50	0.09	0.30
	p-SR/r-CCG p-ACS/r-CCG	0.14	0.56	0.05	0.11	0.00	0.00	0.00	0.02	0.01	0.00	0.00
	p-SR/r-CCG p-CC/r-CCG	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.02	0.00	0.00	0.11
	p-ACS/r-CCG p-CC/r-CCG	0.00	0.00	0.00	0.03	0.15	0.37	0.23	0.51	0.61	0.72	0.15

Table B.3: Significance levels for Experiment 1. Comparing reciprocation-based provider selection strategies.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.7(a)	r-DB, r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
	r-DB, r-CCP	0.00	0.00	0.09	0.15	0.00	0.36	0.05	0.62	0.00	0.31	0.05
	r-DB, r-CCG	0.00	0.00	0.05	0.00	0.03	0.07	0.10	0.47	0.00	0.46	0.03
	r-DB, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
	r-CCP, r-SR	0.00	0.62	0.15	0.07	0.85	0.00	0.14	0.10	0.02	0.00	0.47
	r-CCP, r-ACS	0.52	0.00	0.00	0.00	0.00	0.00	0.01	0.14	0.05	0.00	0.14
	r-SR, r-ACS	0.00	0.00	0.02	0.00	0.00	0.40	0.25	0.86	0.70	0.74	0.38

Table B.4: Significance levels for Experiment 1. Comparing requester selection strategies for p-SR.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.7(b)	r-DB, r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.47	0.03
	r-DB, r-CCP	0.00	0.00	0.60	0.15	0.18	0.02	0.00	0.00	0.69	0.91	0.93
	r-DB, r-CCG	0.00	0.00	0.01	0.01	0.00	0.05	0.28	0.97	0.12	0.77	0.20
	r-DB, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.03	0.01
	r-CCP, r-SR	0.00	0.20	0.00	0.20	0.00	0.01	0.30	0.72	0.01	0.44	0.01
	r-CCP, r-ACS	0.05	0.00	0.00	0.00	0.00	0.00	0.14	0.15	0.01	0.05	0.00
	r-SR, r-ACS	0.00	0.01	0.31	0.00	0.27	0.14	0.62	0.05	0.91	0.15	0.68

Table B.5: Significance levels for Experiment 1. Comparing requester selection strategies for p-ACS.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.7(c)	r-DB, r-SR	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.10	0.20
	r-DB, r-CCP	0.00	0.00	0.00	0.00	0.01	0.01	0.05	0.01	0.10	0.98	0.74
	r-DB, r-CCG	0.00	0.00	0.00	0.00	0.12	0.37	0.40	0.68	0.18	0.12	0.10
	r-DB, r-ACS	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.03	0.28	0.10
	r-CCP, r-SR	0.00	0.10	0.20	0.83	0.33	0.86	0.95	0.91	0.41	0.05	0.28
	r-CCP, r-ACS	0.85	0.00	0.01	0.10	0.10	0.57	0.40	0.94	0.80	0.21	0.14
	r-SR, r-ACS	0.00	0.00	0.00	0.10	0.40	0.68	0.40	0.94	0.50	0.61	0.62

Table B.6: Significance levels for Experiment 1. Comparing requester selection strategies for p-EB.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.7(d)	r-DB, r-SR	0.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	r-DB, r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	r-DB, r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	r-DB, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	r-CCP, r-SR	0.00	0.00	0.07	0.25	0.07	0.12	0.98	0.62	0.93	0.77	0.50
	r-CCP, r-ACS	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
	r-SR, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table B.7: Significance levels for Experiment 1. Comparing requester selection strategies for p-DB.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.7(e)	r-DB, r-SR	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
	r-DB, r-CCP	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.05	0.02	0.11
	r-DB, r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.01	0.20
	r-DB, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	r-CCP, r-SR	0.00	0.28	0.56	0.07	0.63	0.00	0.86	0.93	0.18	0.56	0.47
	r-CCP, r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.05	0.00	0.00	0.11
	r-SR, r-ACS	0.00	0.00	0.00	0.12	0.00	0.01	0.15	0.02	0.09	0.07	0.34

Table B.8: Significance levels for Experiment 1. Comparing requester selection strategies for p-CC.

B.2 Experiment 2

Figure	Strategies	Observations			
		1	2	3	4
9.8(a)	p-SR/r-SR	0.00	0.00	0.00	0.00
9.8(b)	p-ACS/r-SR	0.00	0.00	0.01	0.00
9.8(c)	p-DB/r-SR	0.00	0.00	0.00	0.00
9.8(d)	p-CC/r-SR	0.00	0.00	0.00	0.25
9.9(a)	p-SR/r-DB	0.00	0.00	0.00	0.00
9.9(b)	p-ACS/r-DB	0.44	0.63	0.17	0.31
9.9(c)	p-DB/r-DB	0.00	0.00	0.00	0.00
9.9(d)	p-CC/r-DB	0.00	0.00	0.00	0.20
A.4(a)	p-SR/r-ACS	0.00	0.00	0.00	0.00
A.4(b)	p-ACS/r-ACS	0.00	0.00	0.00	0.00
A.4(c)	p-DB/r-ACS	0.00	0.00	0.00	0.91
A.4(d)	p-CC/r-ACS	0.00	0.00	0.00	0.00
A.5(a)	p-SR/r-CCP	0.00	0.00	0.00	0.00
A.5(b)	p-ACS/r-CCP	0.00	0.10	0.00	0.00
A.5(c)	p-DB/r-CCP	0.00	0.00	0.12	0.07
A.5(d)	p-CC/r-CCP	0.00	0.00	0.31	0.00
A.6(a)	p-SR/r-CCG	0.00	0.00	0.00	0.02
A.6(b)	p-ACS/r-CCG	0.00	0.00	0.01	0.00
A.6(c)	p-DB/r-CCG	0.00	0.00	0.00	0.11
A.6(d)	p-CC/r-CCG	0.00	0.00	0.01	0.00

Table B.9: Significance levels for Experiment 2. Strategies are compared against p-EB.

Figure	Strategies	Observations			
		1	2	3	4
9.10(a)	r-DB, r-SR	0.01	0.00	0.00	0.02
	r-DB, r-CCP	0.00	0.92	0.40	0.00
	r-DB, r-CCG	0.00	0.11	0.46	0.00
	r-DB, r-ACS	0.00	0.00	0.00	0.00
	r-CCP, r-SR	0.00	0.00	0.00	0.00
	r-CCP, r-ACS	0.00	0.00	0.00	0.00
	r-SR, r-ACS	0.30	0.01	0.12	0.00

Table B.10: Significance levels for Experiment 2. Comparing requester selection strategies for p-SR.

Figure	Strategies	Observations			
		1	2	3	4
9.10(b)	r-DB, r-SR	0.00	0.00	0.00	0.85
	r-DB, r-CCP	0.00	0.12	0.00	0.00
	r-DB, r-CCG	0.00	0.00	0.00	0.34
	r-DB, r-ACS	0.28	0.15	0.18	0.15
	r-CCP, r-SR	0.00	0.00	0.00	0.00
	r-CCP, r-ACS	0.00	0.01	0.00	0.10
	r-SR, r-ACS	0.00	0.41	0.05	0.18

Table B.11: Significance levels for Experiment 2. Comparing requester selection strategies for p-ACS.

Figure	Strategies	Observations			
		1	2	3	4
9.10(c)	r-DB, r-SR	0.00	0.00	0.00	0.00
	r-DB, r-CCP	0.00	0.00	0.03	0.00
	r-DB, r-CCG	0.58	0.00	0.60	0.02
	r-DB, r-ACS	0.00	0.00	0.00	0.00
	r-CCP, r-SR	0.00	0.00	0.69	0.05
	r-CCP, r-ACS	0.00	0.00	0.14	0.18
	r-SR, r-ACS	0.43	0.00	0.20	0.37

Table B.12: Significance levels for Experiment 2. Comparing requester selection strategies for p-EB.

Figure	Strategies	Observations			
		1	2	3	4
9.10(d)	r-DB, r-SR	0.00	0.00	0.00	0.00
	r-DB, r-CCP	0.07	0.00	0.00	0.00
	r-DB, r-CCG	0.93	0.00	0.00	0.00
	r-DB, r-ACS	0.00	0.00	0.00	0.00
	r-CCP, r-SR	0.02	0.00	0.00	0.00
	r-CCP, r-ACS	0.00	0.00	0.15	0.00
	r-SR, r-ACS	0.00	0.00	0.00	0.00

Table B.13: Significance levels for Experiment 2. Comparing requester selection strategies for p-DB.

Figure	Strategies	Observations			
		1	2	3	4
9.10(e)	r-DB, r-SR	0.00	0.23	0.00	0.00
	r-DB, r-CCP	0.18	0.03	0.00	0.00
	r-DB, r-CCG	0.80	0.87	0.00	0.00
	r-DB, r-ACS	0.00	0.07	0.00	0.00
	r-CCP, r-SR	0.00	0.00	0.67	0.03
	r-CCP, r-ACS	0.00	0.00	0.20	1.00
	r-SR, r-ACS	0.23	0.70	0.31	0.01

Table B.14: Significance levels for Experiment 2. Comparing requester selection strategies for p-CC.

B.3 Experiment 3

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.11(a)	p-SR/r-SR	0.94	0.43	0.76	0.97	0.75	0.62	0.77	0.76	0.95	0.62	0.69
9.11(b)	p-ACS/r-SR	0.66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9.11(d)	p-DB/r-SR	0.93	0.23	0.33	0.54	0.87	0.28	0.75	0.40	0.15	0.07	0.38
9.11(e)	p-CC/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9.11(c)	p-EB/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.7(a)	p-SR/r-DB	0.10	0.44	0.80	0.72	0.02	0.52	0.69	0.89	0.87	0.85	0.54
A.7(b)	p-ACS/r-DB	0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.7(d)	p-DB/r-DB	0.43	0.74	0.93	0.86	0.92	0.69	0.98	0.30	0.93	0.18	0.34
A.7(e)	p-CC/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.7(c)	p-EB/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.8(a)	p-SR/r-ACS	0.81	0.56	0.68	0.31	0.09	0.25	0.74	0.76	0.99	0.77	0.34
A.8(b)	p-ACS/r-ACS	0.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.8(d)	p-DB/r-ACS	0.31	0.93	0.14	0.28	0.02	0.18	0.43	0.20	0.14	0.80	0.86
A.8(e)	p-CC/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.8(c)	p-EB/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.9(a)	p-SR/r-CCP	0.69	0.99	0.15	0.18	0.15	0.23	0.00	0.30	0.14	0.67	0.51
A.9(b)	p-ACS/r-CCP	0.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.9(d)	p-DB/r-CCP	0.25	0.87	0.56	0.02	0.28	0.81	0.63	0.17	0.64	0.40	0.18
A.9(e)	p-CC/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.9(c)	p-EB/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.10(a)	p-SR/r-CCG	0.88	0.46	0.40	0.05	0.18	0.18	0.28	0.23	0.56	0.89	0.10
A.10(b)	p-ACS/r-CCG	0.66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.10(d)	p-DB/r-CCG	0.50	0.94	0.79	0.72	0.68	0.55	0.20	0.18	0.27	0.68	0.07
A.10(e)	p-CC/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A.10(c)	p-EB/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table B.15: Significance levels for Experiment 3. Strategies are compared against p-RD in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.12(a)	p-ACS/r-SR p-CC/r-SR	0.00	0.00	0.11	0.01	0.18	0.36	0.00	0.01	0.47	0.40	0.00
	p-ACS/r-SR p-EB/r-SR	0.00	0.00	0.00	0.11	0.01	0.75	0.95	0.20	0.12	0.30	0.03
	p-ACS/r-SR p-SR/r-SR	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-SR p-DB/r-SR	0.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
	p-CC/r-SR p-EB/r-SR	0.66	0.10	0.23	0.94	0.40	0.66	0.05	0.40	0.46	0.88	0.68
	p-CC/r-SR p-SR/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-SR p-DB/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-SR p-SR/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-SR p-DB/r-SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-SR p-DB/r-SR	0.83	0.70	0.75	0.31	0.75	0.67	0.25	0.50	0.60	0.01	0.09

Table B.16: Significance levels for Experiment 3. Comparing provider selection strategies for r-SR in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.12(b)	p-ACS/r-ACS p-CC/r-ACS	0.00	0.07	0.00	0.92	0.93	0.03	0.25	0.50	0.18	0.62	0.00
	p-ACS/r-ACS p-EB/r-ACS	0.00	0.00	0.00	0.18	0.01	0.77	0.00	0.00	0.89	0.25	0.00
	p-ACS/r-ACS p-SR/r-ACS	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-ACS p-DB/r-ACS	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-ACS p-EB/r-ACS	0.97	0.38	0.97	0.34	0.05	0.21	0.12	0.14	0.36	0.17	0.46
	p-CC/r-ACS p-SR/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-ACS p-DB/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-ACS p-SR/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-ACS p-DB/r-ACS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-ACS p-DB/r-ACS	0.07	0.61	0.68	0.54	0.05	0.34	0.61	0.07	0.09	0.27	0.07

TABLE B.17: Significance levels for Experiment 3. Comparing provider selection strategies for r-ACS in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.12(c)	p-ACS/r-DB p-CC/r-DB	0.00	0.00	0.05	0.99	0.75	0.20	0.54	0.81	0.36	0.68	0.00
	p-ACS/r-DB p-EB/r-DB	0.00	0.00	0.05	0.50	0.93	0.50	0.10	0.07	0.23	0.23	0.37
	p-ACS/r-DB p-SR/r-DB	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-DB p-DB/r-DB	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-DB p-EB/r-DB	0.75	0.37	0.76	0.62	0.86	0.14	0.10	0.07	0.73	0.49	0.17
	p-CC/r-DB p-SR/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-DB p-DB/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-DB p-SR/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-DB p-DB/r-DB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-DB p-DB/r-DB	0.00	0.76	0.66	0.98	0.41	0.76	0.44	0.68	0.87	0.05	0.50

TABLE B.18: Significance levels for Experiment 3. Comparing provider selection strategies for p-DB in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.12(d)	p-ACS/r-CCP p-CC/r-CCP	0.00	0.00	0.00	0.00	0.38	0.62	0.64	0.30	0.07	0.57	0.00
	p-ACS/r-CCP p-EB/r-CCP	0.00	0.00	0.15	0.40	0.68	0.43	0.20	0.72	0.34	0.56	0.33
	p-ACS/r-CCP p-SR/r-CCP	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-CCP p-DB/r-CCP	0.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-CCP p-EB/r-CCP	0.20	0.93	0.36	0.21	0.38	0.75	0.14	0.27	0.02	0.31	0.00
	p-CC/r-CCP p-SR/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-CCP p-DB/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-CCP p-SR/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-CCP p-DB/r-CCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-CCP p-DB/r-CCP	0.38	0.94	0.56	0.10	0.93	0.88	1.00	0.18	0.93	0.43	0.31

Table B.19: Significance levels for Experiment 3. Comparing provider selection strategies for r-CCP in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.12(e)	p-ACS/r-CCG p-CC/r-CCG	0.00	0.00	0.43	0.91	0.54	0.15	0.40	0.34	0.05	0.20	0.15
	p-ACS/r-CCG p-EB/r-CCG	0.00	0.00	0.33	0.33	0.56	0.21	0.79	0.61	0.92	0.33	0.66
	p-ACS/r-CCG p-SR/r-CCG	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-ACS/r-CCG p-DB/r-CCG	0.20	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-CCG p-EB/r-CCG	0.07	0.66	0.88	0.38	0.95	0.05	0.38	0.27	0.17	0.05	0.50
	p-CC/r-CCG p-SR/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-CC/r-CCG p-DB/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-CCG p-SR/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-EB/r-CCG p-DB/r-CCG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	p-SR/r-CCG p-DB/r-CCG	0.00	0.93	0.57	0.37	0.11	0.70	0.89	0.58	0.28	0.74	0.87

Table B.20: Significance levels for Experiment 3. Comparing provider selection strategies for r-CCG in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.13(a)	r-CCP, r-SR	0.40	0.58	0.20	0.85	0.23	0.15	0.37	0.92	0.40	0.62	0.43
	r-CCP, r-DB	0.00	0.12	0.07	0.43	0.86	0.12	0.10	0.89	0.10	0.52	0.74
	r-CCP, r-ACS	0.87	0.46	0.05	0.56	0.58	0.40	0.23	0.80	0.12	0.50	0.37
	r-CCP, r-CCG	0.00	0.49	0.12	0.49	0.81	0.89	0.60	0.33	0.68	0.28	0.25
	r-ACS, r-SR	0.50	0.88	0.62	0.50	0.46	0.50	0.80	0.73	0.54	0.89	0.05
	r-ACS, r-DB	0.00	0.02	0.85	0.23	0.46	0.44	0.57	0.72	0.86	0.18	0.56
	r-ACS, r-CCG	0.01	0.91	0.81	0.93	0.44	0.49	0.61	0.50	0.25	0.64	0.77
	r-DB, r-SR	0.05	0.05	0.56	0.60	0.15	0.91	0.43	0.98	0.44	0.28	0.20

Table B.21: Significance levels for Experiment 3. Comparing requester selection strategies for p-SR in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.13(b)	r-CCP, r-SR	0.64	0.49	0.05	0.02	0.00	0.00	0.10	0.12	0.33	0.03	0.00
	r-CCP, r-DB	0.81	0.02	0.73	0.97	0.11	0.37	0.09	0.67	0.98	0.86	0.49
	r-CCP, r-ACS	0.94	0.34	0.46	0.82	0.57	0.31	0.91	0.41	0.18	0.34	0.11
	r-CCP, r-CCG	0.69	0.21	0.30	0.66	0.34	0.81	0.64	0.80	0.58	0.56	0.40
	r-ACS, r-SR	0.57	0.85	0.23	0.05	0.03	0.01	0.10	0.40	0.68	0.20	0.15
	r-ACS, r-DB	0.75	0.20	0.25	0.87	0.41	0.97	0.10	0.75	0.14	0.37	0.40
	r-ACS, r-CCG	0.63	0.75	0.05	0.82	0.82	0.17	0.56	0.52	0.03	0.75	0.00
	r-DB, r-SR	0.82	0.17	0.01	0.05	0.20	0.03	0.92	0.30	0.27	0.03	0.02

Table B.22: Significance levels for Experiment 3. Comparing requester selection strategies for p-ACS in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.13(c)	r-CCP, r-SR	0.57	1.00	0.91	0.91	0.43	0.54	0.80	0.15	0.54	0.86	0.31
	r-CCP, r-DB	0.61	0.50	0.46	0.87	0.74	0.47	0.15	0.15	0.99	0.58	0.56
	r-CCP, r-ACS	0.34	0.10	0.46	0.74	0.09	0.99	0.17	0.05	0.83	0.33	0.74
	r-CCP, r-CCG	0.60	0.56	0.87	0.86	0.68	0.73	0.30	0.81	0.23	0.46	0.97
	r-ACS, r-SR	0.69	0.10	0.51	0.68	0.31	0.54	0.10	0.58	0.31	0.37	0.15
	r-ACS, r-DB	0.66	0.02	0.95	0.62	0.05	0.46	0.00	0.62	0.80	0.70	0.31
	r-ACS, r-CCG	0.11	0.33	0.56	0.92	0.18	0.73	0.00	0.10	0.23	0.83	0.76
	r-DB, r-SR	0.94	0.50	0.52	0.98	0.27	0.18	0.25	0.98	0.50	0.67	0.64

Table B.23: Significance levels for Experiment 3. Comparing requester selection strategies for p-EB in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.13(d)	r-CCP, r-SR	0.76	0.58	0.97	0.01	0.62	0.49	0.18	0.09	0.89	0.18	0.97
	r-CCP, r-DB	0.07	0.51	0.37	0.10	0.54	0.50	0.82	0.44	0.43	0.01	1.00
	r-CCP, r-ACS	0.30	0.93	0.43	0.69	0.05	0.87	0.83	0.01	0.03	0.25	0.05
	r-CCP, r-CCG	0.18	0.68	0.43	0.12	0.23	0.63	0.69	0.88	0.56	0.74	0.93
	r-ACS, r-SR	0.25	0.50	0.40	0.07	0.15	0.34	0.23	0.43	0.02	0.00	0.07
	r-ACS, r-DB	0.00	0.56	0.79	0.23	0.20	0.37	0.98	0.11	0.25	0.15	0.09
	r-ACS, r-CCG	0.02	0.62	0.94	0.25	0.50	0.50	0.83	0.01	0.00	0.40	0.07
	r-DB, r-SR	0.20	0.23	0.34	0.62	0.91	0.92	0.28	0.40	0.47	0.00	0.97

Table B.24: Significance levels for Experiment 3. Comparing requester selection strategies for p-DB in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.13(e)	r-CCP, r-SR	0.68	0.10	0.02	0.15	0.34	0.50	0.00	0.00	0.00	0.18	0.00
	r-CCP, r-DB	0.60	0.77	0.68	0.07	0.12	0.21	0.09	0.34	0.01	0.34	0.03
	r-CCP, r-ACS	0.70	0.00	0.94	0.05	0.34	0.07	0.15	0.07	0.09	0.46	0.09
	r-CCP, r-CCG	0.86	0.79	0.38	0.05	0.58	0.10	0.47	0.79	0.72	0.93	0.01
	r-ACS, r-SR	0.99	0.31	0.05	0.50	0.99	0.25	0.07	0.07	0.14	0.52	0.09
	r-ACS, r-DB	0.86	0.00	0.75	0.99	0.51	0.69	0.67	0.37	0.38	0.83	0.80
	r-ACS, r-CCG	0.83	0.00	0.44	0.80	0.67	0.00	0.07	0.11	0.03	0.37	0.68
	r-DB, r-SR	0.86	0.14	0.10	0.52	0.54	0.51	0.23	0.00	0.61	0.67	0.12

Table B.25: Significance levels for Experiment 3. Comparing requester selection strategies for p-CC in terms of average satisfaction.

B.4 Experiment 4

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.14(a)	p-ACS/r-SR	0.00	0.00	0.56	0.27	0.07	0.00	0.05	0.00	0.00	0.63	0.02
9.14(b)	p-ACS/r-DB	0.00	0.00	0.00	0.92	0.66	0.28	0.70	0.28	0.72	0.18	0.37
9.14(c)	p-ACS/r-ACS	0.00	0.00	0.57	0.01	0.00	0.00	0.00	0.00	0.00	0.03	0.09
9.14(d)	p-ACS/r-CCP	0.00	0.00	0.82	0.38	0.02	0.00	0.00	0.00	0.00	0.05	0.44
9.14(e)	p-ACS/r-CCG	0.00	0.01	0.02	0.20	0.07	0.00	0.00	0.00	0.00	0.05	0.99

Table B.26: Significance levels for Experiment 4. Strategy p-ACS is compared against p-EB in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.15(a)	p-CC/r-SR	0.69	0.28	0.11	0.15	0.86	0.09	0.07	0.38	0.28	0.70	0.07
9.15(b)	p-CC/r-DB	0.54	0.50	0.62	0.14	0.99	0.07	0.44	0.75	0.05	0.07	0.50
9.15(c)	p-CC/r-ACS	0.12	0.46	0.61	0.23	0.91	0.76	0.55	0.52	0.73	0.05	0.77
9.15(d)	p-CC/r-CCP	0.40	0.43	0.56	0.58	0.41	0.47	0.72	0.14	0.20	0.12	0.30
9.15(e)	p-CC/r-CCG	0.37	0.11	0.11	0.88	0.20	0.15	0.33	0.40	0.21	0.76	0.31

Table B.27: Significance levels for Experiment 4. Strategy p-CC is compared against p-EB in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.16(a)	r-CCP, r-SR	0.11	0.93	0.18	0.03	0.07	0.01	0.05	0.00	0.00	0.07	0.01
	r-CCP, r-DB	0.92	0.27	0.10	0.79	0.07	0.00	0.64	0.00	0.01	0.00	0.31
	r-CCP, r-ACS	0.05	0.50	0.82	0.62	0.01	0.15	0.46	0.09	0.09	0.75	0.77
	r-CCP, r-CCG	0.37	0.10	0.64	0.66	0.47	0.34	0.91	0.58	0.93	0.55	0.31
	r-CCG, r-SR	0.01	0.09	0.07	0.02	0.30	0.00	0.05	0.00	0.00	0.31	0.00
	r-CCG, r-DB	0.44	0.01	0.05	0.87	0.23	0.00	0.75	0.02	0.02	0.05	0.03
	r-CCG, r-ACS	0.00	0.01	0.49	0.40	0.09	0.00	0.60	0.25	0.11	0.75	0.18
	r-ACS, r-SR	0.97	0.43	0.23	0.10	0.40	0.37	0.00	0.05	0.52	0.12	0.03
	r-ACS, r-DB	0.05	0.54	0.15	0.49	0.66	0.18	0.88	0.23	0.49	0.01	0.43
	r-DB, r-SR	0.10	0.23	0.62	0.03	0.75	0.56	0.03	0.62	0.87	0.28	0.25

Table B.28: Significance levels for Experiment 4. Comparing requester selection strategies for p-CC in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.16(b)	r-CCP, r-SR	0.34	0.00	0.17	0.40	0.73	0.00	0.00	0.00	0.01	0.25	0.00
	r-CCP, r-DB	0.83	0.30	0.33	0.36	0.23	0.00	0.00	0.00	0.00	0.54	0.64
	r-CCP, r-ACS	0.01	0.46	0.68	0.87	0.98	0.03	0.25	0.00	0.15	0.18	0.62
	r-CCP, r-CCG	0.11	0.64	0.88	0.43	0.00	0.44	0.62	0.31	0.89	0.56	0.12
	r-CCG, r-SR	0.41	0.02	0.18	0.12	0.00	0.00	0.01	0.00	0.00	0.07	0.00
	r-CCG, r-DB	0.09	0.50	0.37	0.12	0.00	0.00	0.00	0.02	0.00	0.23	0.05
	r-CCG, r-ACS	0.49	0.77	0.55	0.33	0.00	0.14	0.46	0.02	0.12	0.05	0.05
	r-ACS, r-SR	0.09	0.05	0.07	0.44	0.70	0.10	0.10	0.34	0.27	0.79	0.02
	r-ACS, r-DB	0.00	0.66	0.20	0.40	0.21	0.01	0.00	0.63	0.00	0.50	0.99
	r-DB, r-SR	0.27	0.23	0.92	0.89	0.40	0.36	0.54	0.75	0.02	0.64	0.03

Table B.29: Significance levels for Experiment 4. Comparing requester selection strategies for p-ACS in terms of average satisfaction.

Figure	Strategies	Observations										
		0	10	20	30	40	50	60	70	80	90	100
9.16(c)	r-CCP, r-SR	0.18	0.64	0.93	0.52	0.92	0.03	0.43	0.20	0.27	0.62	0.43
	r-CCP, r-DB	0.56	0.76	0.01	0.93	0.12	0.18	0.93	0.64	0.61	0.92	0.91
	r-CCP, r-ACS	0.10	0.50	0.46	0.23	0.94	0.05	0.09	0.05	0.25	0.28	0.68
	r-CCP, r-CCG	0.46	0.62	0.09	1.00	0.14	0.27	0.38	0.93	0.98	0.70	0.51
	r-CCG, r-SR	0.05	0.93	0.05	0.52	0.17	0.34	0.80	0.33	0.18	0.92	0.15
	r-CCG, r-DB	0.81	0.46	0.37	0.93	0.98	0.83	0.43	0.63	0.54	0.63	0.50
	r-CCG, r-ACS	0.02	0.92	0.25	0.23	0.07	0.50	0.46	0.12	0.17	0.15	0.80
	r-ACS, r-SR	0.81	0.81	0.38	0.55	0.86	0.70	0.21	0.50	0.99	0.10	0.21
	r-ACS, r-DB	0.02	0.34	0.05	0.20	0.07	0.67	0.10	0.03	0.58	0.34	0.62
	r-DB, r-SR	0.05	0.44	0.00	0.47	0.15	0.46	0.47	0.11	0.60	0.56	0.55

Table B.30: Significance levels for Experiment 4. Comparing requester selection strategies for p-EB in terms of average satisfaction.

Bibliography

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, and J. Pruyne. Web services agreement specification. Technical report, Global Grid Forum, 2004.

G. Armano, G. Mancosu, A. Orro, and E. Vargiu. A multi-agent system for protein secondary structure prediction. In *Transactions on Computational Systems Biology III*, volume 3737 of *Lecture Notes in Computer Science*, pages 14–32. Springer-Verlag, 2005.

M. Balmer, N. Cetin, K. Nagel, and B. Raney. Towards truly agent-based traffic and mobility simulations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 60–67. IEEE Computer Society, July 2004.

D. Banerjee, S. Saha, P. Dasgupta, and S. Sen. Reciprocal resource sharing in P2P environments. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 853–859. ACM Press, 2005.

J. Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7): 122–125, 1994.

M. Beer, M. d'Inverno end N. Jennings, M. Luck, C. Preist, and M. Schroeder. Negotiation in multi-agent systems. *Knowledge Engineering Review*, 14(3):285–289, 1999.

A. Birk. Learning to trust. In R. Falcone, M. Singh, and Yao-Hua Tan, editors, *Trust in Cyber-societies: Integrating the Human and Artificial Perspectives*, volume 2246 of *Lecture Notes in Computer Science*, pages 133–144. Springer-Verlag, 2001.

M. Borodovsky and J. McIninch. GeneMark: parallel gene recognition for both DNA strands. *Computers & Chemistry*, 17(19):123–133, 1993.

K. Bryson, M. Luck, M. Joy, and D. T. Jones. Applying agents to bioinformatics in geneweaver. In M. Klush and L. Kerschberg, editors, *Cooperative Information Agents IV*, volume 1860 of *Lecture Notes in Artificial Intelligence*, pages 60–71. Springer-Verlag, 2000.

C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, 1997.

R. Burt. The network structure of social capital. In R. Sutton and B. Staw, editors, *Research in Organizational Behavior*, volume 22, pages 345–423. Elsevier Science JAI, 2000.

A. M. Campbell and L. J. Heyer. *Discovering Genomics Proteomics and Bioinformatics*. Benjamin Cummings, 2002.

F. Casati, M. Castellanos, U. Dayal, and M. Shan. Probabilistic context-sensitive and goal-oriented service selection. In *Proceedings of the Second International Conference On Service Oriented Computing*, pages 316–321. ACM Press, 2004.

C. Castelfranchi. Social Power: A point missed in multi-agent, DAI and HCI. In Y. Demazeau and J. Müller, editors, *Decentralized A.I.*, pages 49–62. Elsevier Science, 1990.

C. Castelfranchi. Modelling social action for AI agents. *Artificial Intelligence*, 103(1-2): 157–182, 1998.

C. Castelfranchi and R. Conte. Limits of economic and strategic rationality for agents and MA systems. *Robotics and Autonomous Systems*, 24(3-4):127–139, 1998.

C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized A.I 3*, pages 215–227. Elsevier Science, 1992.

J. Caverlee, L. Liu, and D. Rocco. Discovering and ranking web services with BASIL: a personalized approach with biased focus. In *Proceedings of the Second International Conference on Service-oriented Computing*, pages 153–162. ACM Press, 2004.

D. C. Chamrad, G. Koerling, J. Gobom, H. Thiele, J. Klose, H. E. Meyer, and M. Blueggel. Interpretation of mass spectrometry data for hight-throughtput proteomics. *Analytical and Bioanalytical Chemistry*, 376(7):1014–1022, 2003.

D. C. Chamrad, G. Korting, K. Stuhler, H. E. Meyer, J. Klose, and M. Bluggel. Evaluation of algorithms for protein identification from sequence databases using mass spectrometry data. *Proteomics*, 4(3):619–628, 2004.

G. Chin, J. Myers, and D. Hoyt. Social networks in the virtual science laboratory. *Communications of the ACM*, 45(8):87–92, 2002.

J. Cohen. Bioinformatics - An introduction for computer scientists. *ACM Computing Surveys*, 36(2):122–158, 2004.

R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm acceptance. In *Proceedings of the Fifth International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 99–112. Springer-Verlag, 1998.

R. Conte and R. Falcone. Norms obligations and conventions. *AI Magazine*, 18(4): 145–147, 1997.

R. Conte, R. Hegselmann, and P. Terna. Social simulation - A new disciplinary synthesis. In R. Conte, R. Hegselmann, and P. Terna, editors, *Simulating Social Phenomena*, volume 456 of *Lectures Notes in Economics and Mathematical Systems*, pages 1–20. Springer-Verlag, 1997.

G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems : concepts and design*. Addison-Wesley, 2001.

R. Craig and R. C. Beavis. A method for reducing the time required to match protein sequences with tandem mass spectra. *Rapid communications in mass spectrometry*, 17(20):2310–2316, 2003.

F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in web services. *Communications of the ACM*, 46(10):29–34, 2003.

N. David, J. S. Sichman, and H. Coelho. Agent-based simulation with coalitions in social reasoning. In S. Moss and P. Davidson, editors, *Multi-Agent-Based Simulation*, volume 1979 of *Lecture Notes in Artificial Intelligence*, pages 245–266. Springer-Verlag, 2001.

H. Dawid, M. Reimann, and B. Bullnheimer. To innovate or not to innovate? *IEEE Transactions on Evolutionary Computation*, 5(5):471–481, 2001.

J. Day and R. Deters. Selecting the best web service. In *Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research*, pages 293–307. IBM Press, 2004.

D. De Roure and J. A. Hendler. E-science: The grid and the semantic web. *IEEE Intelligent Systems*, 19(1):65–71, 2004.

K. Decker, S. Khan, C. Schmidt, G. Situ, R. Makkena, and D. Michaud. BioMAS: A multi-agent system for genomic annotation. *International Journal of Cooperative Information Systems*, 11(3–4):265–292, 2002.

K. Decker, X. Zheng, and C. Schmidt. A multi-agent system for automated genetic annotation. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 433–440. ACM Press, 2001.

D. Dhyani, K. Wee, and S. S. Showmick. A survey of web metrics. *ACM Computing Surveys*, 34(4):469–503, 2002.

F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

V. Dignum. Using agent societies to support knowledge sharing. In *Proceedings of the Workshop on Autonomy, Delegation and Control at AAMAS Conference*, 2003.

V. Dignum and F. Dignum. Modelling agent societies: Co-ordination frameworks and institutions. In *Progress in Artificial Intelligence Knowledge Extraction, Multi-agent Systems, Logic Programming, and Constraint Solving*, volume 2258 of *Lecture Notes in Computer Science*, pages 191–204. Springer-Verlag, 2001.

V. Dignum and F. Dignum. Knowledge market: Agent-mediated knowledge sharing. In *Proceedings of the Third International/Central and Eastern European Conference on Multi-Agent Systems*, pages 168–179, 2003.

V. Dignum, J. Meyer, H. Weigand, and F. Dignum. An organizational-oriented model for agent societies. In G. Lindemann, D. Moldt, M. Paolucci, and B. Yu, editors, *Proceedings of the First International Workshop on Regulated Agent-Based Social Systems: Theories and Applications*, pages 31–50, 2002.

D. Dubois and H. Prade. *Fuzzy sets and systems: theory and applications.* Academic Press, 1980.

E. H. Durfee. Distributed problem solving and planning. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, 1999.

N. Edwards and R. Lippert. Generating peptide candidates from amino-acid sequence databases for protein identification via mass spectrometry. In R. Guigo and D. Gusfield, editors, *Proceedings of the Second Workshop on Algorithms in Bioinformatics*, volume 2452 of *Lectures Notes in Computer Science*, pages 68–81. Springer-Verlag, 2002.

W. Edwards and J. R. Newman. *Multiattribute Evaluation*, volume 26 of *Quantitative Applications in the Social Sciences.* Sage, 1982.

A. Elfatatry and P. Layzell. Negotiating in service-oriented environments. *Communications of the ACM*, 47(8):103–108, 2004.

M. Ellisman, M. Brady, and et al. The emerging role of BioGrids. *Communications of the ACM*, 47(11):53–62, November 2004.

D. G. Feitelson and M. Treinin. The blueprint for life? *IEEE Computer*, 35(7):34–40, 2002.

M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the Fifth ACM conference on Electronic commerce*, pages 102–111. ACM Press, 2004.

I. Foster. Service-oriented science. *Science*, 308(5723):814–817, 2005.

I. Foster, N. Jennings, and C. 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*, pages 8–15. IEEE Computer Society, 2004.

I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3): 200–222, 2001.

H. T. Gao, J. H. Hayes, and H. Cai. Integrating biological research through web services. *IEEE Computer*, 38(3):26–31, 2005.

L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *Proteomics Research*, 3(5):958–964, 2004.

M. P. Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 125–129, 1983.

N. Gilbert and K. G. Troitzsch. *Simulation for the Social Scientist*, chapter Simulation and Social Science, pages 15–26. Open University Press, 2005.

A. Glass and B. Grosz. Socially conscious decision-making. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 217–224. ACM Press, 2000.

R. H. Glitho, E. Olougouna, and S. Pierre. Mobile agents and their use for information retrieval: a brief overview and an elaborate case study. *IEEE Network*, 16(1):34–41, 2002.

C. Goble, C. Wroe, R. Stevens, and the myGrid consortium. The mygrid project: services architecture and demonstrator. In *Proceedings of the UK e-Science All Hands Meeting*, pages 595–603, 2003.

D. Hales and S. Arteconi. SLACER: a self-organizing protocol for coordination in peer-to-peer networks. *IEEE Intelligent Systems*, 21(2):29–35, 2006.

D. Hales and B. Edmonds. Applying a socially-inspired technique (tags) to improve cooperation in P2P networks. *IEEE Transactions in Systems Man and Cybernetics - Part A: Systems and Humans*, 35(3):385–395, 2005.

S. Hanash. Disease proteomics. *Nature*, 422(6928):226–232, 2003.

G. C. Homans. *Sentiments and Activities*, chapter Social Behaviour as Exchange, pages 278–293. Routledge and Kegan Paul, 1962.

G. C. Homans. *Social Behaviour: Its Elementary Forms.* Harcourt Brace Jovanovich, 1974.

M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

M. N. Huhns, M. P. Singh, M. Burstein, K. Decker, E. Durfee, T. Finin, L. Gasser, H. Goradia, N. Jennings, K. Lakkaraju, H. Nakashima, V. Parunak, J. S. Rosenschein,

A. Ruvinsky, G. Sukthankar, S. Swarup, K. Sycara, M. Tambe, T. Wagner, and L. Zavala. Research directions for service-oriented multigent systems. *IEEE Internet Computing*, 9(6):65–70, 2005.

P. C. K. Hung, H. Li, and J. Jeng. Ws-negotiation: An overview of research issues. In *Proceedings of the Thirthy Seventh Annual Hawaii International Conference on System Sciences*, pages 33–42. IEEE Computer Society, 2004.

N. R. Jennings. Coordination techniques for distributed artificial intelligence. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 187–210. 1996.

N. R. Jennings, K. P. Sycara, , and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–36, 1998.

E. A. Kapp, F. Sch´utz, L. M. Connolly, and et al. An evaluation, comparison, and accurate benchmarking of several publicly available ms/ms search algorithms: Sensitivity and specificity analysis. *Proteomics*, 5(13):3475–3490, 2005.

K. Karasavvas, A. Burger, and R. A. Baldock. A multi-agent bioinformatics integration system with adjustable autonomy. In *Proceedings of the Seventh Pacific Rim International Conference on Artificial Intelligence*, pages 492–501. Springer-Verlag, 2002.

J. Kim. Computers are from Mars, Organisms are from Venus. *IEEE Computer*, 35(7): 25–32, 2002.

K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park. Qos for web services: Requirements and possible approaches. Working group note, W3C, November 2003.

M. Little. Transactions and web services. *Communications of the ACM*, 46(10):49–54, 2003.

R. Lopez. EBI - Bioinformatics educational resources. Available at http://www.ebi.ac.uk/2can/home.html, November 2003.

F. Lopez y Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12(2-3):227–250, 2005.

P. Lord, Chris Wroe, Robert Stevens, Carole Goble, S. Miles, L. Moreau, K. Decker, T. Payne, and J. Papay. Semantic and personalised service discovery. In *Proceedings of the UK e-Science All Hands Meeting*, pages 787–794, 2003.

M. Luck and M. d'Inverno. Autonomy: A nice idea in theory. In C. Castelfranchi and Y. Lesperance, editors, *Intelligent Agents VII*, volume 1986 of *Lecture Notes in Artificial Intelligence*, pages 351—353. Springer-Verlag, 2001.

M. Luck and M. d'Inverno. *Understanding Agent Systems.* Springer-Verlag, 2001.

M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing).* AgentLink, 2005.

S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

L. Moreau, S. Miles, C. Goble, and et al. On the use of agents in a bioinformatics grid. In *Proceedings of the Network Tools and Applications in Biology Workshop (NETTAB) - Agents in Bioinformatics*, pages 14–27, 2002.

S. Munroe, M. Luck, and M. d'Inverno. Motivation-based selection of negotiation partners. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1520–1521. IEEE Computer Society, 2004.

T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray, and N. Fiddian. Agent-based formation of virtual organisations. *Knowledge Based Systems*, 17(2-4):103–111, 2004.

R. Overbeek, T. Disz, and R. Stevens. The SEED: A peer-to-peer environment for genome annotation. *Communications of the ACM*, 47(11):47–50, 2004.

S. Parsons and N. R. Jennings. Negotiation through argumentation - A preliminary report. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 267–274. AAAI Press, 1996.

S. Parsons and M. Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

D. N. Perkins, D. J. C. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

E. Phizicky, P. Bastiaens, H. Zhu, M. Snyder, and S. Fields. Protein analysis on a proteomic scale. *Nature*, 422(6928):208–215, 2003.

J. Piaget. *Sociological Studies.* Routlege, 1973.

R. L. Riolo, M. D. Cohen, and R. Axelrod. Evolution of cooperation without reciprocity. *Nature*, 414(6862):441–443, 2001.

M. R. Rodrigues and M. Luck. Analysing partner selection through exchange values. In J. Sichman and L. Antunes, editors, *Multi-Agent-Based Simulation VI*, volume 3891 of *Lecture Notes in Artificial Intelligence*, pages 24–40. Springer-Verlag, 2006a.

M. R. Rodrigues and M. Luck. Cooperative interactions: An exchange values model. In *Proceedings of the Coordination, Organization, Institutions and Norms in Agent*

Systems Workshop at the Seventeenth European Conference on Artificial Intelligence, pages 63–70, 2006b.

M. R. Rodrigues and M. Luck. Evaluating dynamic services in bioinformatics. In M. Klusch, M. Rovatsos, and T. Payne, editors, *Cooperative Information Agents X*, volume 4149 of *Lecture Notes in Artificial Intelligence*, pages 183–197. Springer-Verlag, 2006c.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter Intelligent Agents, pages 31–50. Prentice Hall, 1995.

J. Sabater and C. Sierra. REGRET: Reputation in gregarious societies. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 194–195. ACM Press, 2001.

S. Saha, S. Sen, and P. S. Dutta. Helping based on future expectations. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 289–296. ACM Press, 2003.

S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

T. W. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press, 1999.

S. Sen and P. S. Dutta. The evolution and stability of cooperative traits. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1114–1120. ACM Press, 2002.

S. Sen, P. S. Dutta, and S. Saha. Emergence and stability of collaborations among rational agents. In *Cooperative Information Agents VII*, volume 2782 of *Lecture Notes in Computer Science*, pages 192–205. Springer-Verlag, 2003.

J. S. Sichman, R. Conte, C. Castelfranchi, and Y. Demazeau. A social reasoning mechanism based on dependence networks. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 188–192, 1994.

R. G. Smith. The Contract Net Protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

L. Stein. Creating a bioinformatics nation. *Nature*, 417(6885):119–120, 2002.

L. D. Stein, P. Sternberg, M. Mangone, R. Durbin, J. Thierry-Mieg, and J. Spieth. Wormbase: network access to the genome and biology of caenorhabditis elegans. *Nucleic Acids Research*, 29(1):82–86, 2001.

R. Stevens, K. Glover, C. Greenhalgh, C. Jennings, S. Pearce, P. Li, M. Radenkovic, and A. Wipat. Performing in silico experiments on the grid: A users perspective. In *Proceedings of the UK e-Science All Hands Meeting*, pages 43–50, 2003.

H. Sun, B. Fang, and H. Zhang. User-perceived web QoS measurement and evaluation system. In X. Zhou, J. Li, H. T. Shen, M. Kitsuregawa, and Y. Zhang, editors, *Proceedings of the Eighth Asia-Pacific Web Conference*, volume 3841 of *Lecture Notes in Computer Science*, pages 157–165. Springer-Verlag, 2006.

A. Tate. The helpful environment: Distributed agents and services which cooperate. In M. Klusch, M. Rovatsos, and T. Payne, editors, *Cooperative Information Agents X*, volume 4149 of *Lecture Notes in Artificial Intelligence*, pages 23–32. Springer-Verlag, 2006.

W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 997–1004. ACM Press, 2005.

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

The Wellcome Trust. Key facts about the human genome. Available at http://www.wellcome.ac.uk/en/genome, February 2001.

The Wellcome Trust Case Control Consortium. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447(7145):661–678, 2007.

M. Tyers and M. Mann. From genomics to proteomics. *Nature*, 422(6928):193–197, 2003.

M. P. Wellman. The economic approach to artificial intelligence. *ACM Computing Surveys*, 27(3):360–362, 1995.

A. B. Williams, T. A. Krygowski, and T. L. Casavant. I-DOCS: distributed agent-assisted knowledge fusion for disease gene discovery. In *Proceedings of the Eighth International Conference on Parallel and Distributed Systems*, pages 698–705. IEEE Computer Society, 2001.

M. Wooldridge. Intelligent agents. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–77. MIT Press, 1999.

M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

M. J. Wooldridge and N. R. Jennings. Cooperative problem solving. *Journal of Logic and Computation*, 9(4):563–592, 1999.

C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems*, 19(1):48–55, 2004.

K. P. Yoon and C. Hwang. Multiple attribute decision making: An introduction. volume 104 of *Quantitative Applications in the Social Sciences*. Sage, 1995.