A FLEXIBLE APPROACH FOR CONTEXT-AWARE SERVICE SELECTION IN AGENT-MEDIATED E-COMMERCE

by

Murat Şensoy

B. S., Chemical Engineering, Boğaziçi University, 2001

M. S., Computer Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2008

A FLEXIBLE APPROACH FOR CONTEXT-AWARE SERVICE SELECTION IN AGENT-MEDIATED E-COMMERCE

APPROVED BY:

Assist. Prof. Pınar Yolum (Thesis Supervisor)	. . . . .
Prof. Levent Akın	. . . . .
Assist. Prof. Ayşe Bener	. . . . .
Prof. Oğuz Dikenelli	. . . . .
Assist. Prof. Yücel Saygın	. . . . .

DATE OF APPROVAL: 12.06.2008

ACKNOWLEDGEMENTS

I am deeply indebted to all people who helped and inspired me during the preparation of this dissertation: Pınar Yolum, Levent Akın, Yücel Saygın, Ayşe Bener, Suzan Üsküdarlı, Cem Say, Ethem Alpaydın, Tunga Güngör, Munindar Singh, Reyhan Aydoğan, Akın Günay, Çetin Meriçli and many present and past members of Boğaziçi University Artificial Intelligence Laboratory (AILab). I am also grateful to my colleagues from department of Computer Engineering at Kültür University, especially to Murat Taylı for his priceless encouragement and support during my PhD studies. I specially thank Jie Zhang and Robin Cohen from University of Waterloo for their sincere cooperation during my research on deception.

I want to express my heartfelt thanks to my supervisor Pınar Yolum. Without her support, advice, guidance and inspiration, the work presented in this dissertation would not have been possible. I am also profoundly thankful to my family and my fiance Burcu Yılmaz for their support and encouragement; not only during my studies but also all through my life.

This research has been partially supported by Boğaziçi University Scientific Research Projects (under grants BAP05A104, BAP06A103 and BAP07A102) and The Scientific and Technological Research Council of Turkey (TUBITAK) by a CAREER Award under grant 105E073. I specially thank these institutions for their support.

Some results in this dissertation have previously been published: the research related to ontology-based service representation and selection in Chapter 3 appeared in [1–5]; the research related to the deceptive information filtering described in Chapter 4 has been published in [6]; the research related to cooperative evolution of service ontologies in Chapter 5 has been published in [7–9]; finally, the research on choosing an efficient service selection mechanism among alternatives in Chapter 6 appeared in [10].

ABSTRACT

A FLEXIBLE APPROACH FOR CONTEXT-AWARE SERVICE SELECTION IN AGENT-MEDIATED E-COMMERCE

Selecting the right parties to interact with is a fundamental problem in open and dynamic environments. The problem is exemplified when the number of interacting parties is high and the parties reasons for selecting others vary. In this dissertation, we examine the problem of service selection in an e-commerce setting where consumer agents cooperate to identify service providers that would satisfy their service needs the most. There are three major challenges related to service selection: (i) representing consumers' past dealings with providers, (ii) handling deceptive information about providers and (iii) managing evolution of consumers' service needs and semantics.

Previous approaches represent consumers' past dealings with providers only as a rating. There are two important weaknesses related to this representation. One, ratings are given in a particular context. Even though the context is crucial for interpreting the ratings correctly, ratings do not contain any contextual information explicitly. Two, ratings do not explicitly state what kind of a service a provider supplies. A rating merely represents subjective opinion of a rater about the services of a particular provider. Because, the satisfaction criteria and the expectations of the rater are unknown, it is almost impossible to make sense of a rating.

This dissertation deals with these two weaknesses in two steps. First, we extend a classical rating-based approach by adding a representation of context. This addition improves the accuracy of selected service providers only when two consumers with the same service request are assumed to be satisfied with the same service. Next, we replace over-all ratings with detailed *experiences* of consumers. The experiences are represented with an ontology that can semantically capture the requested service and the received service in detail. When

a service consumer decides to share its experiences with a second service consumer, the receiving consumer evaluates the experience using its own context and satisfaction criteria. By sharing experiences rather than ratings, the service consumers can model service providers more accurately and thus can select service providers that are better suited for their needs.

In many settings, consumers may prefer to be dishonest about their past dealings with providers. In deceptive environments where there are liars among the consumers, successful service selection becomes a harder challenge. To deal with this, we propose a method to evaluate consumers' trustworthiness in a distributed setting on the basis of the consumers' shared experiences. We integrate this method to the experience-based service selection to filter out deceptive information during service selection. Our experiments show that using the integrated approach, service consumers can select the service providers for their needs more accurately even if a significant portion of the consumers are liars.

Communication among consumers requires a common vocabulary to facilitate successful information exchange. One way to achieve this is to assume the existence of a common ontology among communicating consumers. However, in this case, consumers with evolving service needs may not use the system, especially if the existing concepts in the common ontology become insufficient after some time. To handle this, we propose an approach in which consumer agents can add new service concepts into their service ontologies and teach others services from their ontologies by exchanging service descriptions. This leads to a society of agents with different but overlapping ontologies where mutually accepted services emerge based on agents' exchange of service descriptions.

With these contributions, this dissertation proposes an integrated approach for service selection, which is flexible enough to enable consumers to evolve their service semantics over time; context-aware and consumer-oriented to enable consumers to use their own satisfaction criteria and context during service selection; and robust to deception to lead satisfactory service selections even in highly deceptive environments.

ÖZET

ETMEN ARACILI E-TİCARET'DE BAĞLAM FARKI GÖZETEN SERVİS SEÇİMİ İÇİN ESNEK BİR YAKLAŞIM

Açık ve dinamik ortamlarda etkileşim için taraf seçimi önemli bir problemdir. Bu problem etkileşen tarafların sayısının fazla olmasıyla ve tarafların birbirlerini seçme sebeplerinin değişmesiyle daha da zorlaşmaktadır. Bu tezde, servis müşterisi etmenlerin servis ihtiyaçlarını en iyi şekilde karşılayacak servis sağlayıcıları seçmek için yardımlaştıkları bir e-ticaret ortamında, servis seçimi problemi incelenmiştir. Servis seçimi ile ilgili üç temel mesele bulunmaktadır: (i) müşterilerin servis sağlayıcılarla geçmiş ilişkilerinin betimlenmesi, (ii) servis sağlayıcılar ile ilgili yanıltıcı bilgilerin ayıklanması, (iii) müşterilerin gelişen servis ihtiyaçlarının ve bunların anlamsal ilişkilerinin yönetilmesi.

Daha önceki yaklaşımlar müşterilerin servis sağlayıcılarla olan geçmiş ilişkilerini betimlerken sadece reytingleri kullanmaktadırlar. Reytingler ile ilgili iki temel problem bulunmaktadır. İlk olarak, reytingler belirli bir bağlamda verilirler. Ait oldukları bağlam reytingleri değerlendirmek için çok önemli olmasına karşın reytingler herhangi bir bağlamsal bilgiyi açıkça içermezler. İkinci olarak ise, reytingler bir servis sağlayıcının verdiği servisin özelliklerini açıkça belirtmezler. Bir reyting bir müşterinin sadece belirli bir servis sağlayıcı ile ilgili sübjektif fikrini ifade eder. Reytingleri veren müşterilerin memnuniyet kriterleri ve beklentileri bilinemeyeceği için reytingleri yorumlamak neredeyse imkansızdır.

Bu tez reytingler ile ilgili belirtilen bu iki problemi iki basamakta çözmektedir. İlk olarak, klasik reytingler bağlam bilgisi eklenerek geliştirilmektedir. Bu ekleme ile birlikte servis sağlayıcı seçme doğruluğu sadece aynı servis talebinde bulunan müşterilerin aynı servislerle memnun oldukları varsayıldığında artırmaktadır. İkinci olarak, müşterilerin tecrübeleri, talep ettikleri ve aldıkları servisleri detaylı ve anlamsal olarak ifade edebilen bir ontoloji ile betimlenmektedir. Bir müşteri başka bir müşteri ile geçmiş tecrübelerini

paylaşmaya karar verdiğinde, paylaşılan tecrübeleri alan müşteri, bu tecrübeleri kendi bağlam ve memnuniyet kriterlerine göre değerlendirebilmektedir. Reytinglerin yerine tecrübelerini paylaşarak, servis müşterileri servis sağlayıcıları daha doğru modelleyebilmekte ve böylece kendi ihtiyaçlarına en uygun servis sağlayıcıları daha iyi seçebilmektedirler.

Bir çok ortamda, müşteri etmenler servis sağlayıcılarla olan geçmiş ilişkileri hakkında dürüst olmamayı tercih edebilir. Müşterilerin arasında yalancıların var olduğu yanıltıcı ortamlarda, başarılı servis seçimi daha da zor bir problem olmaktadır. Bu problemle başa çıkabilmek için, müşterilerin güvenilirliklerini paylaştıkları tecrübelere dayanarak dağınık bir ortamda ölçen bir metod önerilmiştir. Bu metod önerilen tecrübe tabanlı servis seçme yöntemine, servisleri seçerken yanıltıcı bilgileri ayıklamak için entegre edildi. Deneylerimiz gösterdi ki, bu entegre yöntemi kullanarak servis müşterileri ihtiyaçları için en uygun servis sağlayıcıları müşterilerin önemli bir çoğunluğu yalancı olsa bile doğrulukla seçebilmektedirler.

Müşteri etmenler arasında başarılı bir bilgi alışverişini sağlayabilmek ortak bir kelime hazinesini gerektirmektedir. Bunu başarmanın bir yolu iletişim kuran müşteriler arasında ortak bir ontolojinin varlığını farzetmektir. Ancak, bu ontolojide var olan kavramlar bir süre sonra yetersiz kaldığında, servis ihtiyaçları gelişen müşteriler oluşturulan sistemi kullanamayabilirler. Bu durumla başa çıkabilmek için, servis müşterilerin ontolojilerine yeni servis kavramlarını ekleyebilmelerini sağlayacak ve servis tanımlarını değiş tokuş ederek bu servisleri birbirlerine öğretmelerine olanak verecek bir metod önerilmiştir. Bu metod etmenlerin birbirlerinden farklı ancak çakışan ontolojilere sahip olmalarına yol açmaktadır; öyleki müşterilerce karşılıklı olarak kabul edilen servisler, servis tanımlarının değiş tokuşuyla ortaya çıkmaktadır.

Tüm bu katkılarla, bu tez servis seçimi için entegre bir yaklaşım önermektedir. Bu yaklaşım müşterilerin servis kavramlarını zamanla geliştirmelerine olanak sağlayacak kadar esnek, servis seçimi sırasında müşterilerin kendi memnuniyet kriterlerini ve bağlamlarını kullanmalarına olanak sağlayacak kadar bağlam farkı gözeten ve müşteri odaklı, ayrıca yanıltıcı ortamlarda bile başarılı servis seçimi yapabilecek kadar aldatmaya dayanıklıdır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

e	An experience
F_{taste}	Taste function
g	Discriminant
l	Satisfaction level
p	Probability
P_{CD}	Variation in context
PI	Probability of indeterminism for service providers
R	Reputation
R_{liar}	Ratio of liars
R_{pri}	Private credit
R_{pub}	Public credit
R_{subj}	Ratio of subjectivity
s_i	Score of an experience i
$Tr(A)$	Trustworthiness of A
w	Weight
α	Alpha parameter of a beta probability distribution function
β	Beta parameter of a beta probability distribution function
γ	Confidence measure
Γ	Gamma function
ϵ	Maximum acceptable level of error
ρ	Binary rating
ϕ	Maximum acceptable deviation from the majority
AI	Artificial Intelligence
APS	Adaptive Probabilistic Search
BRS	Beta Reputation System
CBR	Case-Based Reasoning
$DAML$	DARPA Agent Markup Language

$DAML+OIL$	DARPA Agent Markup Language+OIL
DHT	Distributed Hash Tables
DL	Description Logic
GM	Gaussian Model
KIF	Knowledge Interchange Format
MAS	Multiagent Systems
OIL	Ontology Interchange Language
OML	Ontology Markup Language
OWL	Web Ontology Language
$P2P$	Peer-to-Peer
PDF	Probability Distribution Function
QoS	Quality of Service
QRP	Query Routing Protocol
RDF	Resource Description Framework
$RDF(S)$	Resource Description Framework and Schema
$RDFS$	Resource Description Framework Schema
RI	Routing Indices
RL	Reinforcement Learning
$SHOE$	Simple HTML Ontology Extension
SPS_{CAR}	Service Selection using Context-Aware Ratings
SPS_{CBR}	Service Selection using experiences and Case-Based Reasoning
SPS_{DT}	Service Selection using experiences and C4.5 Decision Trees
SPS_{GM}	Service Selection using experiences and Gaussian Model
SPS_{NB}	Service Selection using experiences and Naive Bayes
SPS_{RR}	Service Selection using Random Ratings
SPS_{SR}	Service Selection using Selective Ratings
$SWRL$	Semantic Web Rule Language
TTL	Time-to-Live
XOL	XML-based Ontology Exchange Language

1. INTRODUCTION

Electronic commerce (e-commerce) is concerned with a broad range of issues including payment mechanisms, advertising, directory services, security, trust and reputation, and so on. Agent technologies can be applied to any of these areas where an autonomous behavior is desired. Hence, e-commerce becomes a major application area of agent technologies. As a result, a new paradigm, called *agent-mediated e-commerce*, emerges as an important extension of e-commerce in which agent technologies are used effectively [11]. In agent-mediated e-commerce, there are two main sides interacting with each other: service consumers and service providers. Service consumers are autonomous agents that are interested in receiving services for their human users. Service providers are intelligent entities that offer some services with varying quality and properties. In this setting, service consumers interact with service providers to satisfy their users' service needs. Service consumers and providers operate on the Web, which is not managed by a central authority that can monitor all agents' activities and ensure that everyone acts in the best interest of others [12, 13]. Openness of the Web implies that for a given service description, a plethora of service providers with substantially different service offerings will exist. A service consumer that is interested in receiving a particular service should then need to select a subset of the service providers that will satisfy its service needs in the best way.

Selection of the most satisfactory service providers for a specific service demand is known as the *Service Selection* problem in the literature [14, 15]. In service selection approaches, service consumers share valuable information about the service providers (e.g., ratings, referrals, and so on) and they usually depend on the shared information during service selection because their knowledge about the providers is usually limited.

Service selection has been widely studied in the literature from different directions. The most widely used approaches are variants of *reputation systems* in which the consumers rate the service providers and share these ratings with other consumers. A consumer that is looking for a service provider can check the ratings of a provider from the reputation system and decide to select the provider accordingly. Typically, the ratings are kept in a central

server, which aggregates the ratings in various ways to help others decide whether a service provider will act as expected [16]. E-bay[1] is a well-known web site that uses a reputation system. E-Bay gives buyers the opportunity to rate each seller as positive, negative, or neutral (i.e. 1, -1, 0) after completion of a transaction. After collecting ratings from sellers in a centralized manner, e-Bay computes the reputation scores for each buyer. Reputation score of each buyer is simply the sum of positive ratings minus the sum of negative ratings. Reputation systems have also applications in the service domains other than e-commerce. Expert sites such as AskMe[2] use these systems to help their users decide on experts to ask their questions. In this context, service providers are the human experts that are willing to answer questions and the services are their answers to these questions. Depending on the quality of her answers, the users can rate an expert on the system. Then, these ratings are centrally aggregated and a reputation score is displayed for each expert. Most of the commercial reputation systems also allow their users to leave their comments or reviews on the system. However, these comments are in natural language and meant to be read by human users instead of machines. Therefore, they cannot be interpreted by machines (e.g., software agents) for the computation of reputation scores. Traditional reputation systems are suitable for the closed settings where a central authority (such as a company) can monitor the overall system. E-bay and expert sites are examples of these closed systems. However, the traditional reputation systems are not directly applicable in open systems [14], where there is no central authority.

Since reputation systems are not directly applicable, it is necessary to find approaches that are distributed. Most distributed approaches to service selection consider *trust* among entities [14, 17, 18]. Trust captures a truster's expectation from a trustee for a particular service. Whereas different formalizations of trust exists, most formalizations are not expressive enough. That is, typically a truster's trust in the trustee is represented by a mere rating. However, the episode that leads to the rating is as important for understanding the rationale for the rating as the rating itself [19]. For example, a service consumer may give a low rating to a service provider that delivers a book two days late. If the delivery date is not significant for a second service consumer, the first service consumer's low rating will not be significant,

[1] http://www.ebay.com
[2] http://www.askme.com

either. Hence, it is important that ratings are evaluated within their scope considering the satisfaction criteria of the raters. Scope of a rating is the context in which the rater has experienced the service, and satisfaction criteria of the rater are not known to others. Because ratings are not expressive enough to represent context and satisfaction criteria of the consumers are not known for the rated services, rating-based service selection approaches (such as trust and reputation systems) are not context-aware and suffer from subjectivity [20].

For exactly the same service interest (context) and the same service supplied for this service interest, different consumers may give marginally different ratings depending on their satisfaction criteria. That is, different consumers on the Web may have highly different satisfaction criteria for the same service. For example, a consumer buying a book may expect it to be delivered within one week and not tolerate any delay. However, for another consumer, one or two days delay in the delivery may be acceptable. Therefore, evaluation of these two consumers of a specific provider may change if the provider's service is delivered with some delay. As a result, a service selection approach should be flexible enough to work in the settings where consumers have highly different satisfaction criteria for the same services. If a service selection approach cannot handle the differences in the consumers' satisfaction criteria, the shared information among the consumers may suffer from the subjectivity (i.e., ratings given by different consumers for the same services may conflict).

Rating-based approaches implicitly assume that satisfaction criteria of consumers and behavior of providers do not change significantly in different contexts. However, behavior of the service providers and expectations of the service consumers may considerably change in different contexts. For example, while a provider usually delivers products on time, it may not deliver a special type of product (e.g., fridges) on time as expected. Similarly, while a consumer usually values price more than quality, the same consumer considers quality as the only metric when buying a birthday present. Hence, a service selection approach should be *context-aware* while evaluating the shared information about the service providers.

Service selection approaches depend on the information from other consumers while making service decisions. However, there is no guarantee that consumers honestly share the information about the service providers. Some consumers may disseminate deceptive

information to mislead others. Especially in open systems like the one we are addressing in this dissertation, handling deceptive information may be more difficult, because there is no central authority monitoring the behavior of participants in these systems. If the deceptive information is not detected and removed during service selection, service selection approaches may definitely fail [20]. As a result, a service selection approach should have a mechanism to handle deceptive information in order to make successful service selections even in deceptive environments.

The Web is a highly dynamic environment where consumers' service needs evolve over time. A service selection approach should be *flexible* enough to work for new service needs that were not known to the community before. This may require evolution of the language between the consumers to accommodate new vocabulary representing the new service needs; otherwise the consumers may not properly express their evolving service needs and they cannot share related information about the service providers. Consider a service selection approach that enables consumers to request information for a specific service. The consumers should have mutual understanding of the services they request and the general context. If the needs of the consumers change over time, the consumers will be obliged to request new services that are possibly unknown to other consumers. In such situations, the consumers should have a method to express a new service request, describe it properly and possibly teach it to other consumers so that others can be helpful for identifying potential service providers.

In order to enhance current service selection approaches to handle context-awareness and subjectivity, consumers should be given representations that are more expressive than ratings. Semantic Web technologies such as ontologies provide a foundation that allows semantic data to be shared [21]. For example, using an ontology, it is possible to semantically describe the interactions between a consumer and a provider in detail. Then, this representation of past dealings with the provider can easily be interpreted by other consumers using ontological reasoning [22].

Accordingly, this dissertation develops an approach for distributed service selection that allows consumers to semantically represent and share their past *experiences* with the

service providers using ontologies [1]. An experience captures an episode of a customer with a provider and can be thought of as a record of what service the customer has requested and received in return. In this way, experience-based approaches allow the objective facts of the experiences (rather than subjective opinions, i.e., ratings) to be communicated to the other party. A consumer that receives other consumers' particular experiences can interpret what they have experienced with the providers and evaluate the providers using its own satisfaction criteria and context. Although this approach removes subjectivity inherent to ratings, it is still vulnerable to deception, because some consumers may lie about their past experiences. Therefore, we propose a method for filtering deceptive experiences and integrate it into our service selection approach. Finally, we propose a mechanism to enable consumers to evolve their service semantics to accommodate the fact that their service needs may evolve over time.

1.1. Research Challenges

This section summarizes the main research challenges related to the distributed context-aware service selection that we address in this dissertation.

Consumers look for the service providers that satisfy their service needs the most. Therefore, consumers may communicate to share valuable information about the past dealings of service providers. During this communication, it is essential for the parties to understand each other properly. Therefore, consumers should use a shared representation while they are interacting with one another. Ontologies provide a common grounding for communication between different parties [23, 24]. Ontology languages such RDF[3] and OWL have become Web standards [25]. Hence, different consumer agents developed by different organizations can easily communicate with each other using ontologies written in these languages. Ontologies can easily represent concepts and relations between the concepts, as well as axioms and rules regarding these concepts and relations. Although ontologies are powerful tools for representing semantic information, how they can be used for service selection is an open question.

[3] http://www.w3.org/TR/rdf-syntax-grammar

Most of the current approaches in multiagent systems assume that the agents share an ontology through which they can communicate properly [3, 14]. In the service selection domain, this assumption cannot account for the fact that consumers' needs may evolve over time and new concepts may be necessary for describing consumers' evolving service needs. However, a service selection approach should be able to accommodate this, since in many e-commerce settings, individuals learn new concepts and services from different sources, add them to their ontology, and further form service requests that are based on these new concepts. It is a challenge to design a representation that can accommodate evolution.

Consumers may need to retrieve information about service providers (e.g., consumers' past experiences) related to their current service interests. For example, a consumer that wants to buy a notebook may interact with other consumers to collect experiences related to buying a notebook. Current P2P information retrieval approaches propose several solutions for locating interested information (e.g., documents and files) in large communities. However, these approaches are designed to use syntactic matches, instead of semantic matches. For example, these approaches may not locate an experience about buying *IBM Thinkpad T60* because they are not aware that it is actually an instance of a notebook. Moreover, in these approaches, the search results are ranked using a global similarity metric (e.g., edit distance, frequency of search words and so on). However, in the problem we are addressing, the notion of similarity may have different meanings for different consumers. For example, quality of the service may dominate over its price for some consumers, whereas the price may be more important for other consumers. For the former case, similarity between service interests may depend mainly on the similarity between their quality attributes, while for latter case, similarity may depend mainly on the similarity between the constraints on the price. As a result, current approaches for information retrieval may not be convenient for retrieving semantic information in context-aware service selection, because context-awareness and semantics are not included in their design.

In many settings, a consumer may prefer to be dishonest about its past dealings with the providers. For example, the consumers may provide untruthful experiences to promote some providers or cooperate with other providers to drive a provider out of the system. If a trust mechanism is incorporated into the system, untrustworthy consumers can be detected

and their effect on the system can easily be inhibited. So, it is important to detect who is trustworthy in agent-mediated e-commerce. Consumers should rank other consumers according to their trustworthiness and handle the information flow considering this ranking. For example, if a consumer considers another consumer untrustworthy, it should neglect the information provided by this consumer. Each consumer may have different metrics to define who is trustworthy and who is not. Techniques to enable consumers to effectively compute trustworthiness of others should be developed.

After collecting past experiences about the providers from other consumers, a consumer should interpret these experiences and decide on a service provider that is expected to provide the most satisfactory service. There are many methods for decision making in the literature (e.g., decision trees, case-based reasoning and so on). However, these methods are not designed for handling semantic information. Hence, these algorithms may need to be enhanced before they can be used for the proposed context-aware service selection framework.

1.2. Objectives

The aim of the dissertation is, therefore, to provide techniques to solve the challenges related to the context-aware service selection problem. This aim is further developed into the following objectives:

1. To develop ontology-based representations for the description of consumers' past experiences with the service providers, so as to replace current rating-based representations with semantically rich representations.
2. To design different methods for decision making, based on the proposed representations of the past experiences.
3. To integrate trust mechanisms to the proposed service selection framework, so as to handle possible deceptive information problem.
4. To enable consumers to evolve their service semantics so as to represent their evolving service needs better without obstructing the communication between them.

By fulfilling these objectives, this dissertation proposes a flexible framework for context-aware service selection. This framework enables successful service selection; (i) in dynamic environments where service interest of consumers evolve over time; (ii) in deceptive environments where a significant portion of consumers are liars; and (iii) in diverse environments where the consumers frequently change the context of their service demands and their satisfaction criteria vary significantly.

1.3. Organization of the Dissertation

The remainder of the dissertation is organized as follows:

In Chapter 2, we present a general overview of the literature related to *representation*, *information retrieval* and *trust and reputation systems*. In Chapter 3, we extend a classical rating-based approach by adding a representation of context. For this purpose, an ontology is proposed. This addition improves the accuracy of selected service providers only when two consumers with the same service request are assumed to be satisfied with the same service. Next, we replace ratings with detailed *experiences* of consumers. The experiences are represented using a proposed ontology that can capture the requested service and the received service in detail. When a service consumer decides to share its experiences with a second service consumer, the receiving consumer evaluates the experience using its own context and satisfaction criteria. By sharing experiences rather than ratings, the service consumers can model service providers more accurately and thus can select service providers that are better suited for their needs. In Chapter 4, we propose a deceptive information filtering approach tailored for experience-based service selection and integrate this approach into the proposed framework. Our experiments show that using the proposed approach, service consumers can select the service providers for their needs more accurately even if a significant portion of them are liars.

In Chapter 5, we propose an approach to enable consumers to add new service concepts into their ontologies when necessary, without obstructing the communication between them. Using this approach, agents not only add new service concepts into their service ontologies, but also teach others service concepts from their ontologies by exchanging service

descriptions. This leads to a society of agents with different but overlapping ontologies where mutually accepted services emerge based on agents' exchange of service descriptions. Our simulations of societies show that allowing cooperative evolution of local service ontologies facilitates better representation of agents' needs. Further, through cooperation, not only more useful services emerge over time, but also ontologies of agents having similar service needs become aligned gradually.

In this dissertation, we propose novel service selection approaches. Although, there are various service selection approaches available in the literature, each approach (including the ones proposed in this dissertation) has different strengths and weaknesses for different settings. In Chapter 6, we propose a novel approach for consumers to learn how to choose the most useful service selection mechanism among different alternatives in dynamic environments. In this approach, consumers continuously observe outcomes of different service selection mechanisms. Using their observations and a reinforcement learning algorithm, consumers learn to choose the most useful service selection mechanism with respect to their trade-offs. Lastly, in Chapter 7, we review the work presented and sketch the perspectives for further research.

2. OVERVIEW OF THE LITERATURE

With the coming of information age, concepts like e-mail, e-learning and e-commerce are introduced into our lives and automation of our daily activities became an indispensable obligation. Now, autonomous systems are replacing automated systems. As a result, the Multiagent Systems (MAS) is emerged as an important field of research. In MAS, software agents autonomously interact with their environment to fulfill desires of their owners. Agent mediated e-Commerce is an important research area in this scope. This section roughly summarizes the approaches regarding to the service selection in agent mediated e-commerce.

In a fully distributed environment, communicating parties should use a common language in order to communicate properly. This mandates a common representation of concepts and domain knowledge. In Section 2.1, we summarize approaches for knowledge representation. In agent-mediated e-commerce, agents may discover the service interests of one another and collect the information of their interest in a peer-to-peer setting. Hence, in Section 2.2, we overview the approaches for peer-to-peer communication and information retrieval. As the number of autonomous agents in the environment increases, determination of right parties to communicate with becomes harder. Trust becomes an essential parameter during interactions and constitution of trust between agents is a non-trivial issue. In Section 2.3, we overview approaches for the constitution of trust and reputation.

2.1. Representation

In order to represent domain knowledge, ontologies are usually used. Ontologies provide a shared and common description of concepts and relationships of a domain in a computer readable form. The main objective of an ontology is to enable communication and interoperability between parties by providing a shared conceptualization. An ontology can be represented in various ways. However, an ontology must contain a vocabulary of concepts and relations as well as specification of their meanings. Ontologies usually exist within a hierarchy. Meta-ontologies are used to define concepts of domain and generic ontologies. A generic ontology, also called top ontology, specifies general concepts which can be used in

different domains. On the other hand, a domain ontology is used to define concepts and relations in a specific domain of application. Different types of ontologies can be used in conjunction with each other for better representation and management of knowledge.

An ontology may take a variety of forms depending on the language it is represented in. There are a lot of potential representation languages for ontology definition. These languages range from highly informal languages such as natural languages to formal languages. For the sake of machine readability, formal languages are more promising for ontology representation. Some well-known formal ontology representation languages are KIF, CycL, Ontolingua, Frame logic, SHOE, RDF(S), XOL, OML, OIL, DAML+OIL and OWL.

Berners-Lee *et al.* define the Semantic Web as an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [26]. It will bring structure to the meaningful content of Web pages and will enable software agents to carry out sophisticated tasks for their users. In order to fulfill this new semantic oriented structure of the web, ontologies and languages for the expression of those languages is an obligation. Today, there are several ontology languages becoming web standards. Some of them are based on XML syntax, such as Ontology Exchange Language (XOL), SHOE (which was previously based on HTML), Ontology Markup Language (OML), Resource Description Framework (RDF), Ontology Interchange Language (OIL), DARPA Agent Markup Language+OIL (DAML+OIL) and Web Ontology Language (OWL).

Languages such as RDF, OIL, DAML+OIL and OWL depend on Description Logic [27]. These languages are shortly explained in this section. Description logic provides powerful tools for describing concepts and relations. Description logic inherits a lot from the early research on Semantic Web and defines a formal semantics for the description of concepts and relations. Classification taxonomies can be automatically derived from these descriptions. Whereas the concept hierarchy is modeled explicitly in frame-based systems, a distinguishing feature of description logics is that concepts and concept hierarchy do not have to be defined explicitly. A concept can be defined in terms of descriptions specifying the properties that objects must satisfy in order to belong to the concept.

2.1.1. Resource Description Framework (RDF) and RDF Schema (RDFS)

RDF(S) is developed by W3C for describing Web resources. It provides a way of specifying the semantics of data using XML syntax in a standardized and interoperable manner. Moreover, RDF(S) provides mechanisms to explicitly represent services, processes, and business models and allows recognition of non-explicit information. The RDF data model well fits to the semantic webs formalism. In RDF representation, resources are described by RDF expressions and are always named by URIs. Resources may have properties, which define specific aspects of or relations between the resources. RDFS is used to define the relationships between properties and resources. RDFS provide a basic schema for RDF and could directly be used to describe ontologies. In RDF(S), predefined properties can be used to model instance-of or subclass-of relationships as well as domain and range restrictions of attributes. An RDF document consists of triples. All RDF triples are asserted facts. An RDF triple consists of a subject, a predicate, and an object. The assertion of an RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. A simple RDF triple is shown in Figure 2.1.

```
<foaf:knows>
  <foaf:Person>
    <foaf:nick>Cal</foaf:nick>
    <foaf:name>Cal Henderson</foaf:name>
    <foaf:mbox_sha1sum>2971b1c2fd1...5671d1c4e32</foaf:mbox_sha1sum>
    <rdfs:seeAlso rdf:resource="http://www.iamcal.com/foaf.xml"/>
  </foaf:Person>
</foaf:knows>
```

Figure 2.1. A simple RDF triple

The primitives used by RDFS for defining knowledge model are similar to that of frame-based approaches. However, RDFS provides reified second-order logic as in CycL and KIF and RDF expressions are terms in meta expressions. Neither Frame Logic nor most Description Logic based languages provide such expressiveness. In RDF(S), properties are defined globally and are not encapsulated as attributes in class definitions. Hence, a frame ontology can only be expressed by reifying the property names with class name suffixes. Al-

though RDF(S) provides a rather strong reification mechanisms and has serious contributions to the ontology representation research, it provides limited expressiveness power and lacks for a standard of describing logical axioms . So, RDF(S) is extended by several ontology languages.

2.1.2. Ontology Interchange Language (OIL)

OIL is developed in the OntoKnowledge project to enable semantic interoperability between Web resources. The problem with most of the ontology representation languages is their high expressive power that is provided without any means of control. So, no reasoning support has ever been provided for these languages. OntoKnowledge project started OIL as a very simple and limited core language. By restricting initial complexity and making controlled extension on OIL, researchers in OntoKnowledge project added reasoning support to the language. Its syntax and semantics are based on the preceding ontology languages; OKBC, XOL, and RDF(S). Modeling primitives in OIL are similar to that of frame-based approaches. On the other hand, formal semantics and reasoning support in OIL is inherited from description logic approaches. So OIL has decidability and an efficient inference mechanism. OIL is built on the top of RDF(S). It has four levels; Core OIL, Standard OIL, Instance OIL and Heavy OIL. Core OIL provides a direct mapping of OIL primitives to RDF(S) primitives. Standard OIL is the complete OIL model and provides more primitives than the defined in RDF(S). Instance OIL provides instances of concepts and roles and Heavy OIL is reserved for the future extensions to OIL and it may include additional representational and reasoning capabilities.

2.1.3. DARPA Agent Markup Language+OIL (DAML+OIL)

The DARPA agent markup language (DAML) aims to enable the next generation of the web, which is a transition from a web simply displaying content to that actually infers the meaning of the content. The DAML program has generated the DAML+OIL markup language on the top of RDF(S). As its name implies, DAML+OIL shares the same objective as OIL and there is a close relationship between DAML+OIL and OIL. The DAML inherits many aspects from OIL, and the capabilities of the two languages are relatively similar.

2.1.4. Web Ontology Language (OWL)

OWL was initiated as a revision to the DAML+OIL by W3C. So, it is regarded as an extension of RDF. OWL is part of the growing stack of W3C recommendations regarding the Semantic Web. OWL aims at to be a standard ontology language for the web with an interchangeable ontology format. It unifies three important aspects from previous work on ontologies. It takes powerful knowledge modeling primitives from frame systems. It inherits efficient reasoning support from Description Logics. Lastly, it uses a syntax which is compatible with the current Web standards. OWL uses RDF's flexible approach to representing data. A part of a simple OWL ontology is shown in Figure 2.2.

```
<owl:Class rdf:about="/ka.daml#Institute">
  <rdfs:subClassOf>
    <owl:Class rdf:about="file:/ka.daml#Organization"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="file:/ka.daml#hasParts"/>
      <owl:allValuesFrom>
        <owl:Class rdf:about="file:F/ka.daml#ResearchGroup"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 2.2. A part of a simple OWL ontology

OWL adds more vocabulary to RDF Schema for describing properties and classes such as relations between classes (e.g. disjointness), equality, cardinality (e.g. 'exactly one'), characteristics of properties (e.g. symmetry), and enumerated classes. Increasing expressiveness considerably reduces the reasoning capabilities and decidability of the language. In order to avoid this trade off as much as possible, OWL provides three increasingly expressive sublanguages; OWL Lite, OWL DL and OWL Full. Each of these sublanguages is an extension of its simpler predecessor.

OWL Lite has a lower formal complexity than OWL DL and OWL Full. OWL Lite is a sublanguage of OWL DL that supports only a subset of the OWL language constructs. It is particularly targeted at the users who want to start with a relatively simple basic set of features. Some semantic restrictions are applied to OWL lite in order to guarantee decidability. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.

OWL DL is a sublanguage of OWL Full. DL in its name is an acronym for Description Logic. OWL DL provides the maximum expressiveness while retaining computational completeness and decidability. Computational completeness means that all conclusions are guaranteed to be computable. Decidability means that all computations will finish in finite time. There are currently several reasoners for OWL DL, such as KAON2, JENA, FaCT, FaCT++, RACER and Pellet.

OWL Full has the maximum expressiveness with no computational guarantees. It uses all semantic and syntactic properties of RDF and additional constructs of OWL. For example, in OWL Full a class can be treated simultaneously both as a collection of individuals and as an individual. These powerful expressiveness of OWL FULL makes it unlikely to develop a reasoning software supporting every feature of OWL Full. Hence, a complete implementation of a reasoner supporting OWL Full does not currently exist.

2.1.5. OWL-S: An OWL based Ontology for Web Services

With the semantic web, it will be possible to access Web resources by content rather than just by keywords. New generation of Web markup languages such as OWL [25,28] and its predecessor DAML+OIL [29,30] is an important advancement in this regard. So, creation of ontologies for any domain and the instantiation of these ontologies in the description of Web sites (or Web resources) are now possible. Most important web resources are the ones that provide services. These resources are called Web Services. While introducing the Semantic Web and its goals in this section, we stated that Semantic Web will enable software agents to carry out sophisticated tasks for their users. In order to make use of Web Services, software agents should be able to discover, compose, invoke, and monitor

Web resources providing particular services. Moreover, software agents need computer-interpretable descriptions of these services. A service description should contain what the service does, how it works and how it will be accessed. Semantic markup languages for Web Services aim to make these descriptions within an interchangeable format. OWL-S[4] (formerly known as DAML-S) is an ontology of services, which is developed as a part of the DARPA Agent Markup Language program. This ontology has three sub-ontologies; the service profile, the process model and the grounding. The service profile is used for advertising and discovering services. The process model gives a detailed description of a service's operation and the grounding provides details on how to interoperate with a service. Figure 2.3 depicts these sub-ontologies of the service ontology.

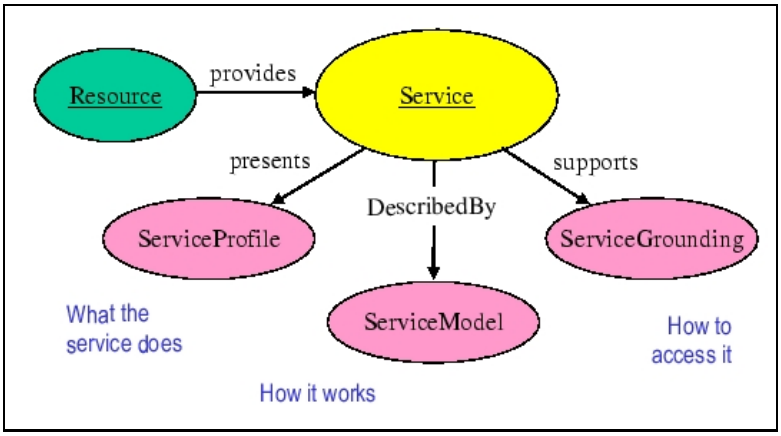

Figure 2.3. Top level of the service ontology

The service profile describes what the service does. It gives sufficient information to a service consumer, which can be either a service-seeking agent or a matchmaking agent acting on the behalf of a service-seeking agent, to determine whether the service meets its needs. The service profile does not only be used for the description of service capabilities but also it can be used for the description of the service demand of a service consumer. By service demand, we mean the needs of a service consumer regarding a specific service domain. So, the service profile provides a dual-purpose representation both for service providers and service consumers.

The service model describes how the service works. This description may be used

[4]http://www.w3.org/Submission/OWL-S

by a service consumer to perform detailed analysis of whether the service really meets its needs. Service consumer can also use the service model to monitor the execution of the service. Moreover, the service model can be useful to compose service descriptions from multiple services to perform a specific task and to coordinate the activities of the different participants during the service enactment.

A service grounding describes the details of how an agent can access a service. This description usually covers a communication protocol, required message formats, and other service-specific details such as port numbers used in contacting the service.

To sum up, the service profile provides the information needed for an agent to discover a service. On the other hand, the service model and service grounding provide enough information for an agent to make use of a service. Therefore, OWL-S, as a whole, enables software agents to automatically discover, invoke, compose, and monitor Web resources offering services, under specified constraints.

2.1.6. Semantic Web Rule Language

Semantic Web Rule Language[5] (SWRL) is proposed to combine sub-languages of the OWL Web Ontology Language (OWL DL and Lite) with the Rule Markup Language[6] (RuleML). SWRL extends the set of OWL axioms to include Horn-like rules ($a \wedge b \Rightarrow c$) [31]. Hence, SWRL enables Horn-like rules to be combined with an OWL ontology. Each SWRL rule is in the form of an antecedent (body) and consequent (head). A SWRL rule can be interpreted as: whenever the conditions specified in the body hold, then the conditions specified in the head must also hold.

Both the body and the head of a rule may consist of zero or more atoms. Multiple atoms are treated as a conjunction. Therefore, a rule with multiple atoms can easily transformed into multiple rules with single atom, using the Lloyd-Topor transformations [32]. An empty rule body is treated as trivially true (i.e. satisfied by every interpretation), so the rule head

[5] http://www.w3.org/Submission/SWRL
[6] http://www.ruleml.org

must also be satisfied by every interpretation. However, an empty rule head is treated as trivially false (i.e., not satisfied by any interpretation), so the rule body must also not be satisfied by any interpretation. Atoms in these rules can be in the form of $C(x)$, $P(x, y)$, $sameAs(x, y)$ or $differentFrom(x, y)$, where C is an OWL description, P is an OWL property, and x, y are either variables, OWL individuals or OWL data values. Figure 2.4 demonstrates a simple SWRL rule for the definition of *Uncle* concept; the rule states that Y is an uncle of X if it is the brother of X's father.

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#swrlexample"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Figure 2.4. SWRL example for the rule:
$hasParent(?x, ?z) \wedge hasBrother(?y, ?z) \Rightarrow hasUncle(?x, ?y)$

2.2. Peer-to-Peer Communication

During service selection, service consumers may need to locate service providers of their interest or other service consumers from which they can request information (ratings or referrals) about these service providers. However, locating right parties is not a trivial task in an open system, because there is no central authority for indexing every service consumers or provider together with their specialties. Nonetheless, there is a solid literature on the Peer-to-Peer(P2P) information discovery and retrial. Therefore, some of the approaches from the literature can be adopted in order to locate parties to interact with.

P2P search protocols can be classified into two main categories depending on how the search is conducted; blind and informed search. In a blind search, peers have no information related to location of the documents of their interest. Therefore, other peers are randomly

probed to locate these documents. So, blind search protocols are usually very simple and they require peers to maintain minimum amount of knowledge. On the other hand, in informed search protocols, peers maintain additional information about locations of the documents. This information is used for the future searches. Unlike blind search protocols, informed search protocols decide which peers to contact with depending on this additional information.

P2P search protocols can also be classified depending on how the network structure is modified to facilitate the searching. Some protocols arrange network structure and build a new network topology in accordance with locations of the documents in the network or locations of computationally powerfully peers. Therefore, the connections between peers are set in a special manner to support an efficient and productive searching scheme. On the other hand, some protocols do not change network structure and do not rearrange connections between the peers. So, network structure evolves chaotically and it is maintained as is. In this section, some P2P searching protocols are briefly explained according to the categories cited above.

2.2.1. Blind Search Approaches

Gnutella [33] implements a decentralized file search approach. It does not reorganize network structure and peers do not maintain information about file locations. So, Gnutella is an example of Blind Search protocols without Network Reorganization. During a search, various peers are randomly asked if they have the desired file. This is accomplished using a technique called controlled flooding. A peer initiates a query message by sending it to all of its neighbors, which are peers connected to it. After a peer receives a query message it checks whether it has the desired file or not and informs initiator of the query if it has the desired file. Each query message has a Time-to-Live (TTL) attribute. Each time a peer receive a query message, it checks this attribute and decrements its value. If TTL becomes zero, the query message is not forwarded to other peers; otherwise it is forwarded to the neighbors of the peer receiving the message. So, the generated network traffic for search operation is limited. By implementing such an unstructured search approach, Gnutella is not affected by the joins and leaves of the peers. However, scalability of the resultant network is influenced because of the high degree of flooding.

FastTrack is based on the Gnutella protocol. Main drawback of Gnutella is its scalability problems. FastTrack overcomes this drawback by introducing heterogeneity between peers. There are two types of peers in the P2P network supernodes and regular peers. The supernodes are peers, which have computer power, bandwidth and availability. These peers are connected to each other and constitute a top level topology. Other peers, namely normal peers, are connected to these peers. So, FastTrack forms a two-level topology on the P2P network. Introduction of supernodes improves scalability, because these peers handle traffic and routing on behalf of weak peers connected to them. This approach is also used in popular P2P applications such as KaZaa, Morpheus, iMesh and MLdonkey. In KaZaa, peers dynamically elect supernodes. Then, these supernodes form an unstructured network. Each regular peer is connected to a supernode. Whenever a regular peer initiates a query, it forwards the message only to its supernode. Then the supernode uses query flooding to locate content on the behalves of the regular peer. Although scalability is improved somehow with the introduction of supernodes, flooding is still resulted in a high volume of network traffic. Additionally, planned attacks to the supernodes may results in serious damages in the network.

Another blind search approach is Random Walks [34]. Random Walks is similar to the Gnutella protocol, but peers do not send query messages to all of their neighbors. After taking a query message, a peer forwards this message to a randomly chosen neighbor. So, overall network load due to a search is reduced considerably. Random Walks uses the properties of network topology. If the network topology has a power-law degree distribution as in most of the social networks, high degree nodes and the peers available for a long time will experience high query loads.

The Logical Clustering [35] approach does not change the actual network topology but it creates dynamic logical clusters by circulating messages called wanderers. Main motivation of these logical clusters is to improve quality of service (QoS) regarding the search response time. The P2P networks have a dynamic nature and their size can become very huge. The clustering divides the network into small groups called clusters such that QoS constraints such as time requirements will hold. These clusters are created on the basis of the time in which peers want to be supplied with information. A wanderer is responsible

for the creation and maintenance of a cluster. This wanderer must be able to visit all peers in that cluster within a given time constraint. Communication between different clusters is accomplished by the wanderers dedicated to these logical clusters. These wanderers choose one peer from each cluster and these peers are connected and construct a tree like structure. This tree is used for spreading messages between clusters during a search. So, the number of messages circulating in the network is reduced considerably.

2.2.2. Informed Search Approaches

Query Routing Protocol (QRP) [36] uses additional information about the locations of the documents in the network. In QRP, each peer maintains a routing table. Entries in this routing table include hashed keywords, which are used to describe locally offered files. Peers regularly exchange these entries with their neighbors by propagating them up to several hops away. In order to code a set of keywords in a transmittable and concise form, bloom filters are used. If a peer receives routing table entries from its neighbors, the peer merges these entries with its routing table. Peers use routing tables to decide which of the neighbors it is worth routing a search query, because their routing tables contain information regarding the documents in their neighborhood and further. In order to decrease the number of hops required to locate a desired document, routing tables should be propagated by peers as far as possible. However, propagating routing tables decreases the validity or recentness of the information due to network dynamics; routing tables of leaving peers will continue to effect routing tables of others.

Routing Indices (RI) approach [37] is similar to QRP, but it does not use bloom filters to summarize content located in their neighborhood. The purpose of RI is to allow peers to select the best neighbors to forward a search query message. It uses a data structure called routing indices which return a list of neighbors ranked according to their goodness for a specific search query. The notion of goodness may change depending on the implementation but, in [37], the goodness of a peer is taken as the number of documents related to the query in its neighborhood. In order to accomplish this task, documents in the network are categorized. Each peer maintains an IR structure for each of its neighbors. This IR structure indicates how many documents of which category could be found through that neighbor.

When a new neighbor is added to the neighborhood of a peer, this peer aggregates its IR structures related to its other neighbors and sends this aggregate to its new neighbor. So the new neighbor gets a comprehensive view of its new neighborhood. In order to keep IR structures valid and recent, each peer informs its neighbors about the changes such as addition or removal of documents or joins/leaves of other neighbors.

Adaptive Probabilistic Search (APS) [38] uses feedback from previous searches to probabilistically guide future searches. Each peer maintains an index table, which summarizes which objects (documents or files) were requested by each of its neighbors. Weights in this index table are used to compute the probability of choosing neighbors for forwarding a search query. So, the probability of choosing a neighbor to find a particular document depends on previous search results. This approach depends on the fact that many of the requested files are usually close to the requesters [39]. So, by forwarding new queries to previous requesters, the probability of finding the requested documents increases. Hence, the P2P network demonstrates a self-learning property. Unlike RI and QRP, this approach does not require additional information regarding the placements of documents in the network. This property makes it more robust to changes in network topology. However, this approach may lead to some sort of starvation; popular files could be located very fast, while other files could not easily be located.

In order to overcome the scalability limitations of previous P2P search systems, Distributed Hash Tables (DHT) approach is developed. Some important P2P content search systems such as CAN [40] and Chord [41] use this approach. In DHT, documents are associated with a key which is produced by a common hashing algorithm. Hashing can be done on the content of the document or its name or both depending on the application. Output of this hashing algorithm constitutes an ID space. For example, if the algorithm is producing 64 bit hexadecimal keys, set of all 64 bit hexadecimal keys constitutes the ID space. Every peer in the P2P network is responsible for storing a certain range of keys. So, each peer is responsible for a partition of ID space. Network structure is arranged by routing tables locally stored on individual peers. A routing table is composed of a list of other peers with addresses and range of the keys they are responsible for. So the network topology is tightly controlled by placing information about documents at the precisely specified locations de-

fined by their keys. DHT based systems provide a highly scalable P2P network because of careful information placement and tightly controlled network topology. However, network structure becomes less reliable, in case the user population is very transient. Additionally, difficulties related to load balancing can reduce the performance of the network. For example, some peers may be overloaded because the range of keys, which they store, belongs to very popular documents. So these peers could not response to all of queries due to their limited capacity.

Zhang *et al.* proposed a multiagent approach for P2P information retrieval systems [42]. In the paper, each agent is modeled using five components: a collection, a collection descriptor, a local search engine, an agent-view structure and a control center. Collection is the collection of documents located in the agent. Collection descriptor is the language model of the collection, which is concise representation or signature of the collection. A local search engine is used to search the documents in the collection of the agent. The agent-view structure of an agent contains information about the collection descriptors of other agents in the neighborhood. An agent-view is analogous to the routing table of a network router. The control center is responsible for accepting queries and performing the distributed search algorithm. The initial topology of the agent society is randomly created using a method inspired from Gnutella. Then, agent-view reorganization algorithm (AVRA) is run to dynamically reorganize the underlying agent-view topology. This algorithm makes agents to share and aggregate their agent-views. This algorithm dynamically adapts the topology by placing semantically similar agents together to form loose content clusters. This clustered topology is used to speed up the distributed search task. The proposed approach associates virtual clusters with collection descriptors and reorganizes those content based clusters dynamically. Hence, it increases the performance of the distributed search algorithms by dividing the space into content-based clusters.

2.3. Trust and Reputation Systems

Trust and reputation systems are on the center of most Internet mediated service selection and provision architectures. In these systems, the basic idea is to enable parties rate each other and use the aggregated ratings to derive a trust or reputation score. These scores

can be used by parties in the service selection in the future. Notions of trust and reputation are frequently used in the agent mediated e-commerce literature. These notions can easily be confused with each other. There can be different definitions of trust. McKnight and Chervany [43] define trust as the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. On the other hand, reputation is defined in Oxford dictionary as what is generally said or believed about a person's or thing's character or standing. This definition overlaps with what is usually understood from the word 'reputation' in the literature. Reputation is be derived quantitatively from the underlying social network. While trust is usually personal and subjective phenomenon, reputation is global. The difference between trust and reputation can be illustrated by the following statements [20]: (i) I trust you because of your good reputation, (ii) I trust you despite your bad reputation. The first statement reflects that the trust is originated from the good reputation of the other party. On the other hand, the second statement reflects that the trust is originated from the private knowledge such as previous experiences or intimate relationship between two parties and it overrules any reputation. These two statements reveal that personal experiences are more important in the constitution of trust than the referrals or reputation, which can be regarded as the aggregation of others' level of trust to a person.

Reputation systems can be can be classified as Centralized and Distributed Reputation Systems. In centralized reputation systems, information about a member is collected as ratings from other members by a central authority. Then, this central authority computes a reputation score for the member and makes this score publicly available. This score is later used by other members to decide whether or not to transact with the member. These systems depend on the observation that transactions with reputable members will result in more favorable outcomes on the average, when all transactions are considered.

Unlike centralized systems, in a distributed system, there is no central location for submitting ratings or obtaining reputation scores. In these systems, ratings are distributed over the network and the members should discover, collect and aggregate these ratings to compute reputation scores. After a transaction, each participant simply records its opinion with the other party (rating) and provides this information to others if it is requested. Before decid-

ing to transact with a member, other members compute the reputation score of the member based on the received ratings. Distributed reputation systems are robust to failures, because there is no single point of failure, and they are suitable for open systems. However, these systems have two challenges. Firstly, a distributed and scalable communication protocol is necessary to enable participants to discover and collect rating from other members of the community. Secondly, a reputation computation method is required, which will be used by each member to derive reputation scores of other members based on the received ratings and other information. P2P search methods, which are stated in Section 2.2, can be used to overcome the first challenge. Hence, this section mainly focuses on the current approaches for the second challenge. Computation of reputation scores from ratings is the most important research question also for the centralized reputation systems.

One of the simplest solutions for this question is simply summing the ratings. Surprisingly, the most famous auction site in the World, e-Bay, uses this simple approach to compute reputation scores [44]. After an auction is completed, both the buyer and the seller can give the other party a rating of +1 (positive), 0 (neutral), or -1 (negative) in addition to their textual comments. Then, e-Bay simply sums these ratings to compute one's reputation score. Actually, e-Bay displays several aggregates of these ratings to its members; (1) the difference between the number of positive and negative feedback rating which is actually the sum of ratings, (2) the percentage of positive feedback ratings, (3) the date when the seller registered with e-Bay, (4) a complete record of the comments received by each seller. So, e-Bay enhances its reputation mechanism. Another approach in this category is to average ratings instead of just summing them up. Numerous commercial web sites such as Epinions[7] and Amazon[8] uses this approaches to compute reputation scores. More advanced approaches in this category use a weighted average of all the ratings to compute reputation scores. In these approaches, a rating is weighted depending on the several factors such as reputation of the rater, age of the rating, distance between rating and current score [20].

FIRE [18] is a trust and reputation model consisting of four components: interaction trust, role-based trust, witness reputation and certified reputation. Interaction trust is related

[7] http://www.epinions.com
[8] http://www.amazon.com

to experiences resulted from direct interactions of an agent with other agents. This is accomplished by rating the attributes of the transaction, including price, quality and delivery date with a range of [-1,+1] (-1 for absolute negative, 0 for neutral and +1 for absolute positive). Experiences are summed with weights such that more recent ones have higher weights. In this way, an agent can build a trust model of another agent. Also, the agent can score the reliability of this trust model using the number and deviation of the ratings used in the computation. Role-based trust models result from the role-based relationships between two agents (e.g. owned by the same organization, acquaintance). For this purpose, rules can be defined to assign role-based trust values. Witness reputation of a target agent is built by making use of past experiences of other agents (witnesses). Certified reputation is related to ratings used by the rated agent as certifications of its past performance. It allows an agent to prove its achievable performance as viewed by previous interaction partners. These four types of trust components and their reliability are combined in order to obtain a composite trust value and its reliability. FIRE system assumes that the agents report their trust information truthfully.

Abdul-Rahman and Hailes [45] proposed a discrete trust model. They used four degrees of agent trustworthiness, which are stated as vt (very trustworthy), t (trustworthy), u (untrustworthy) and vu (very untrustworthy). For each participant and context, the agent maintains a tuple with the number of past experiences in each category, vt, t, u and vu. For instance, if tuple for a participant in a specific context is (3, 4, 0, 0), it means that in 3 out of 7 interactions, the participant was very trustworthy and in 4 out of 7 interactions, it was only trustworthy. For the future interactions, this participant is classified as trustworthy by the agent because the maximum value in the tuple belongs to trustworthy. Whenever the agent has had personal experience with the participant, this tuple is updated and trustworthiness is recalculated. Derived trustworthiness values can also be used to weight referrals. For example, suppose a participant A informs X that another participant B is vt, but X's evaluation for B's trustworthiness with respect to its own experiences is only t. X weights the referrals from A accordingly before taking them into account. So, referrals from participants who are recognized to overrate will be downgraded, and referrals from participants who are recognized to underrate will be upgraded.

The REGRET reputation system is proposed by Sabater and Sierra [46, 47]. Sabater

and Sierra defined three dimensions of reputation; individual dimension, social dimension and ontological dimension. Individual dimension is derived from direct interaction with a participant. On the other hand, social dimension of a participant's reputation is derived from the information coming from other members of the society and the social relations. Social dimension may be very important in the lack of direct interactions. However, social relations may have different forms such as competitive relation or cooperative relation. In order to increase the reliability of social information, REGRET uses fuzzy rules. Lastly, ontological dimension of reputation is derived from the contextual information related to interactions. REGRET computes reputation of a participant using these three dimensions of reputation.

Yolum and Singh proposed to use PageRank [48] for reputation calculations [49] in referral networks. PageRank is a metric used by Google to rank Web pages. Yolum and Singh study P2P service networks consisting of autonomous agents who seek and provide information services among their neighbours. Here, the agents track each other's trustworthiness locally and can give and receive referrals as well. In the model proposed, agents interact with each other through referral graphs. A referral graph is a directed graph where the nodes denote agents and an edge denotes that the source of the edge has referred to the target of the edge. In this graph, an increase in the number of links to an agent leads to an increase in its PageRank value. Then, reputation of the agent is derived from its PageRank value.

One of the main problems in the current trust and reputation systems is unfair ratings. Unfortunately, many of the approaches in the literatures assume that agents share their ratings honestly. However, this is not a realistic assumption in open systems where there is no central authority to monitor who is telling the truth and who is not. In the literature, there are several approaches that develop methods for handling unfair ratings while computing trustworthiness. Some of these approaches are shortly described in the following sections.

2.3.1. Beta Reputation System

In Beta Reputation System (BRS), reputation scores of service providers are computed by statistically updating beta probability density functions (PDF) [50]. The reputation score of a provider is represented by the beta PDF parameters (α, β). These parameters represent

the amount of positive and negative ratings, respectively.

The beta distributions are a family of statistical distribution functions that are characterized by two parameters α and β. The beta distribution denoted as $Beta(p|\alpha, \beta)$ is defined as in Equation 2.1, where Γ is the gamma function and $\alpha, \beta > 0$. Expected value of $Beta(p|\alpha, \beta)$ is defined as in Equation 2.2 and used to shortly characterize the distribution.

$$Beta(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) \times \Gamma(\beta)} p^{(\alpha-1)}(1-p)^{\beta} \tag{2.1}$$

$$E(Beta(p|\alpha, \beta)) = \frac{\alpha}{\alpha + \beta} \tag{2.2}$$

For a single transaction, the reputation system enables a consumer to rate a provider both positively and negatively, by arbitrary amounts. This rating takes the form of a vector $\rho = [r, s]$, where $r \geq 0$ and $s \geq 0$ and they represent the positive and negative components of the rating respectively. That is, for a satisfactory transaction ρ^+ is $[1, 0]$ and for a dissatisfactory transaction ρ^- is $[0, 1]$. A particular rating of the consumer X of provider Z at time t is denoted as $\rho^X_{Z,t}$. X's aggregated rating of Z can be calculated by summing all of its previous ratings as shown in Equation 2.3.

$$\rho^t(X, Z) = \sum_{t_R < t} \rho^X_{Z,t_R} \tag{2.3}$$

If a consumer does not have direct interactions with Z, it can collect aggregated ratings of other consumers to Z and using Equation 2.4, the consumer can calculate Z's aggregate rating.

$$\rho^t(Z) = \sum_{X \in S} \rho^t(X, Z) \tag{2.4}$$

Once aggregated ratings for a particular service provider are known, it is possible to calculate the reputation probability distribution for that provider, which is expressed as $Beta(\rho^t(Z)) = Beta(\rho|r + 1, s + 1)$, where $\rho^t(Z) = [r, s]$. Expected value of this distribution is used as the

reputation score of the provider Z, denoted as $R^t(Z)$.

$$R^t(Z) = E[Beta(\rho^t(Z))] = \frac{r+1}{r+s+2} \tag{2.5}$$

In beta reputation system, an endogenous method is used to filter unfair ratings. This filtering algorithm is iteratively executed whenever Z's reputation score must be recalculated. It assumes the existence of cumulative rating vectors $\rho^t(X, Z)$ for each rater X in the community. The basic principle of the filtering algorithm is to verify that overall score $R^t(Z)$ falls between the q quantile (lower) and $1-q$ quantile (upper) of $\rho^t(X, Z)$ for each rater X. Whenever this is not the case for a given rater, its cumulative rating is considered unfair, and excluded.

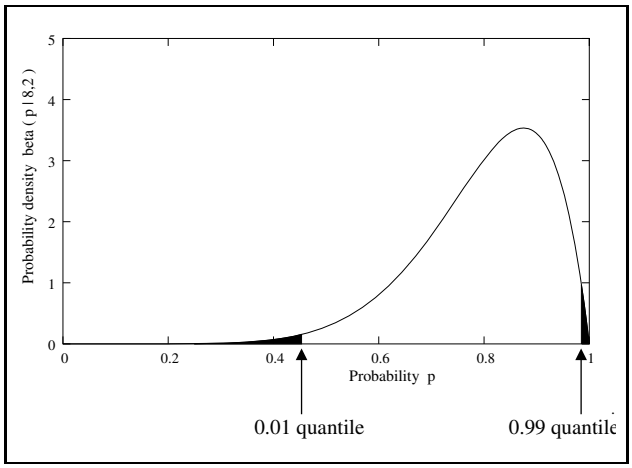

Figure 2.5. 1% and 99% quantiles of $beta(p|8, 2)$

Let consumer X give an aggregate rating $\rho(X, Z) = [7, 1]$ for Z. Then, reputation distribution for Z according to X's rating is defined by $Beta(p|8, 2)$, which is shown in Figure 2.5. Assume that $q = 0.01$, so 1% percent of data fall below and 99% fall above the lower quantile. If the overall rating score $R^t(Z)$ stays within the margins defined by the quantiles in the figure, then X 's rating is regarded as fair. Otherwise, it is regarded as unfair and excluded. For example, if $R^t(Z) < 0.45$ or $R^t(Z) > 0.98$, X's rating is regarded as unfair. This approach depends on the assumption that majority of the ratings are fair and it does not consider the personal observations of the consumers while determining unfair ratings.

2.3.2. Robust Reputation System for Mobile Ad-Hoc Networks

Buchegger and Boudec propose a robust reputation system for mobile ad-hoc networks (RRSMAN) [51]. In this section, we describe this approach shortly in the context of service selection. In RRSMAN, a service consumer in the network maintains first-hand information and a reputation rating about each service provider.

The first-hand information record of the consumer X for a service provider Z has the form of $F_{X,Z} = (\alpha, \beta)$. In this notation, α and β are the parameters of the Beta distribution assumed by X in its Bayesian view of Z's behavior. These parameters are initiated as $(1, 1)$ and updated after X's each transaction with Z as shown in Equation 2.6 and 2.7. In these equations, u is a discount factor for past experiences and s is 1.0 if the transaction is satisfactory, otherwise it is 0.0.

$$\alpha = u.\alpha + s \tag{2.6}$$

$$\beta = u.\beta + (1 - s) \tag{2.7}$$

The reputation rating $R_{X,Z}$ is also defined by two parameters, α' and β'. These parameters are also initiated as $(1, 1)$. However, they are updated in two cases: (1) when first-hand information is updated (2) when a first-hand information ($F_{K,Z}$) published by some other consumer K is received. In the first case, the update is the same as the update of the first-hand information using the observation $s \in \{0, 1\}$. In the second case, $R_{X,Z}$ is updated using $F_{K,Z}$ using Equation 2.8, where w is a small positive constant.

$$R_{X,Z} = R_{X,Z} + w.F_{K,Z} \tag{2.8}$$

Buchegger and Boudec propose a exogenous method to filter unfair first-hand information published by consumers. For this purpose, they use *Trust Ratings*. Trust rating of X to K is denoted as $T_{X,K}$ and defined as (γ, δ). Initially, $T_{X,K} = (\gamma, \delta) = (1, 1)$ and it

is updated whenever a first-hand information is received from K. Let X have a reputation rating $R_{X,Z} = (\alpha, \beta)$ about the provider X and let X receive $F_{K,Z} = (\alpha_F, \beta_F)$, which is the first-hand information of K about Z. Then, X updates (γ, δ) using Equation 2.9 and 2.10. In the equations, $s = 1.0$ if $F_{K,Z}$ passes deviation test in Equation 2.11, otherwise $s = 0.0$. In the deviation test, d is a positive constant (deviation threshold). Trusfulness of K is determined using $T_{X,K}$. Therefore, $R_{X,Z}$ is updated with $F_{K,Z}$ only if $E[T_{X,K}]$ is greater than a threshold such as 0.5.

$$\gamma = v.\gamma + s \tag{2.9}$$

$$\delta = v.\delta + (1 - s) \tag{2.10}$$

$$|E[Beta(p|\alpha_F, \beta_F)] - E[Beta(p|\alpha, \beta)]| < d \tag{2.11}$$

2.3.3. TRAVOS

This is also a probabilistic approach to model a consumer's trust on a service provider using its individual experiences [52]. If a truster, namely consumer X, has complete information about a trustee, namely provider Z, then, according to X, the probability that Z produces satisfactory services can easily be expressed by a beta distribution. Parameters of this distribution is computed using the set of all interaction outcomes X has observed about Z. The level of trust $\tau_{X,Z}$ is defined at time t as the expected value of the beta distribution given the set of outcomes $O_{X,Z}^{1:t}$. Let $m_{X,Z}^t$ and $n_{X,Z}^t$ be the number of successful and unsuccessful interactions for X with Z, respectively. Then, level of trust $\tau_{X,Z}$ is defined as in Equation 2.12.

$$\tau_{X,Z} = E[Beta(p|\alpha, \beta)] \tag{2.12}$$

$$\alpha = m_{X,Z}^t + 1 \tag{2.13}$$

$$\beta = n^t_{X,Z} + 1 \tag{2.14}$$

If the confidence of the computed trust value is low, consumer X computes the reputation of the provider Z using the opinions of the other consumers. Received opinion of the consumer K about Z at time t is defined as $R^t_{K,Z} = (m^t_{K,Z}, n^t_{K,Z})$. The consumer X, aggregates these opinions using Equation 2.15. This aggregation can be used to calculate shape parameters for a beta distribution as shown in Equation 2.16 and 2.17, to give a trust value determined by the opinions provided from others.

$$M_{X,Z} = \sum_{i=0} m^t_{K_i,Z},\ and\ N_{X,Z} = \sum_{i=0} n^t_{K_i,Z} \tag{2.15}$$

$$\alpha' = M^t_{X,Z} + 1 \tag{2.16}$$

$$\beta' = N^t_{X,Z} + 1 \tag{2.17}$$

TRAVOS estimates the probability that a rater's stated opinion of a service provider is accurate using an exogenous method. Let $R^r_{K,Z} = (m^r_{K,Z}, n^r_{K,Z})$ be the stated opinion of K about Z. The E^r is the expected value of a beta distribution D^r, such that $\alpha^r = m^r_{K,Z} + 1$ and $\beta^r = n^r_{K,Z} + 1$. Accuracy of K'opinion about Z, denoted as $\rho_{K,Z}$, is computed as follows.

First, interval [0,1] is divided into N bins and the one that contains E^r is determined. This bin is called bin^o. The outcomes of all previous interactions for which K provided an opinion to X are considered. Let $H_{X,K}$ be the set of all pairs of the form $(O_{X,A}; R_{K,A})$, where A is any service provider that K provided opinion about, and $O_{X,A}$ is the outcome of the X's interaction with A. Third, the subset $H^r_{X,K} \in H_{X,K}$, which comprises all pairs for which the opinion falls in bin^o are found. From this set, we count the total number of pairs in $H^r_{X,K}$ for which the interaction outcome was successful (denoted $C_{success}$) and, similarly, for those which were not successful (denoted C_{fail}). Then we define a beta distribution D^o with parameters $\alpha^o = C_{success} + 1$ and $\beta^o = C_{fail} + 1$. Accuracy of the K's rating is computed as portion of the total area under D^o that lies in the interval defined by bin^o. If the accuracy of an opinion is low (e.g., less than 0.5), it is regarded as unfair. Figure 2.6 shows an illustration

of how the accuracy $\rho_{X,K}$ is estimated.

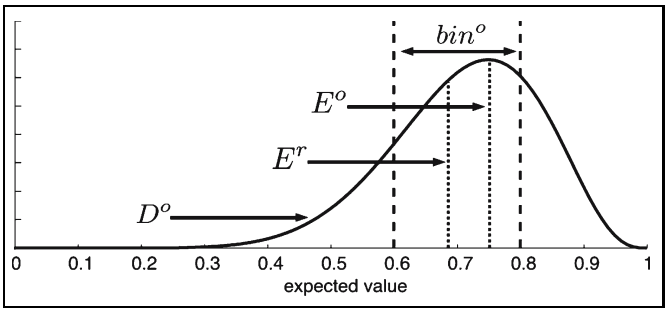

Figure 2.6. Illustration of $\rho_{X,K}$ Estimation Process (5 bins are used)

2.3.4. Yu and Singh's Model of Trust

Yu and Singh proposed a model, which uses Dempster-Shafer theory of evidence to calculate reputation scores [53, 54]. Dempster-Shafer theory of evidence allows the explicit representation of ignorance and combination of evidence. In Dempster-Shafer theory of evidence, the combination of evidence is expressed by Dempster's combination rules, which allow to combine multiple sources of evidence. In the model of Yu and Singh, each agent stores a history of its direct interactions with other participants. In this history, an interaction with a participant is represented by a set of values that reflect the quality of the interaction, namely QoS. There are three QoS categories in this model; trustworthy, uncertain and untrustworthy. Each agent defines two thresholds (upper and lower thresholds) to differentiate between these QoS categories. In other words, using these thresholds, an agent can categorize an interaction with a specific participant into one of these three QoS categories. Using this information together with Dempster-Shafer theory of evidence, an agent can calculate the probability that a participant gives a service within a specific QoS category. The participant being evaluated is considered as trustworthy if the difference between the probabilities that the service belongs to the trustworthy category and it belongs to untrustworthy category is greater than a threshold for the trustworthiness. For these calculations, only the most recent information in the history is considered. In this model, an agent can ask for information about other participants if direct interactions with these participants are not available. When a participant is requested for information about another participant, it can provide the information about the participant only if this participant is in its interaction history. If not,

it will provide referrals that can be queried to obtain the requested information. In order to aggregate the information collected from different participants about a specific participant, Dempster's rule of combination is used. This approach uses information from other participants in the lack of direct interactions. So it does not combine direct interactions with the information gathered from other participants.

Yu and Singh also propose a rating-based service selection algorithm that handles the unfair ratings using a version of weighted majority algorithm [55]. In their algorithm for service selection, weights are assigned to the raters. These weights are initiated as 1.0 and they can be considered as the trustworthiness of the corresponding raters. The algorithm makes predictions about service providers based on the weighted sum of the ratings provided by those raters. This idea is summarized in Equation 2.18, where λ^j denotes the predicted rating for the provider under consideration in trial j, w_i^j is the weight of the rater i and lastly $r_i^j \in [0,1]$ is the rating of the rater i to the provider.

$$\lambda^j = \frac{\sum_{i=0}^{n} w_i^j r_i^j}{\sum w_i^j} \tag{2.18}$$

Yu and Singh propose to tune the weights of the raters after an unsuccessful prediction so that the weights assigned to the unsuccessful raters are decreased. They assume that the ratings of dishonest raters may conflict with the observations of the consumer receiving these ratings. By decreasing the weights of these raters over time, unfair ratings are filtered. The consumer X updates the weight of the rater K using Equation 2.19 and Equation 2.20, where $\rho^j \in [0,1]$ denotes the observation of X after it makes a transaction with the provider for which the rating r_K^j is given by K.

$$w_K^{j+1} = w_K^j \times \theta_K \tag{2.19}$$

$$\theta_K = 1 - \frac{|r_K^j - \rho^j|}{2} \tag{2.20}$$

It seems that $\theta \leq 1.0$ all the time. Therefore, weights of raters cannot increase with their successful ratings, but they decrease with any of their unsuccessful ratings.

3. ONTOLOGY-BASED SERVICE REPRESENTATION AND SELECTION

This chapter develops two novel approaches that allow consumer agents (simply denoted as *consumers*) to make context-aware service selections. In these approaches, consumers use ontologies to express the context of their interactions with service providers. The first of these approaches makes the context of ratings explicit so that the consumers evaluate the ratings within their scope. The second approach enables consumers to record their experiences with service providers in detail. An experience contains the consumer's service demand and the provided service in response to the service demand. More specifically, an experience expresses the story between the consumer and the provider regarding a specific service demand. Equipped with such a description, any consumer receiving an experience can evaluate the service provider according to its own criteria using the objective data in the experience.

The rest of this chapter is organized as follows. Section 3.1 explains the proposed rating-based approach for the context-aware service selection. Section 3.2 experimentally evaluates the rating-based approach. Section 3.3 explains the proposed experience-based approach for the context-aware service selection in detail and proposes two different decision-making schemes. Section 3.4 presents the experimental evaluation of the experience-based approach. Section 3.5 summarizes our contribution, discusses our results, and compares our work to related work in the literature.

3.1. A Rating-Based Approach for Context-Aware Service Selection

We consider an architecture where service consumers are looking for service providers to handle their service demands. A *service demand* is expressed in terms of well-defined constraints on attributes of a service such as service completion time, service price, and so on. If a given service does not meet those constraints, we expect the owner of the demand to be *unsatisfied*. However, another service consumer having weaker demand constraints could potentially be *satisfied*. If consumers only expose their levels of satisfaction (e.g., with

a plain rating), the former service consumer will reveal a low level of satisfaction to the latter consumer. Even though the latter service consumer might have been satisfied with the service provider, it will not choose to interact with the service provider. Instead, if the latter consumer recognizes the scope of the rating, it can infer that the former consumer could be misleading. This shows that the ratings should be made more expressive. In order to increase the expressiveness, service consumers use a common ontology called context ontology for the specified service domain [56]. This ontology covers the domain level knowledge and fundamental concepts such as service demand, service provider, and rating.

3.1.1. Context Ontology

Figure 3.1 demonstrates the context ontology for an online shopping domain. This ontology is used to represent context-aware ratings denoted as $ContextAwareRating$. A context-aware rating mainly represents what a service consumer has requested from a service provider (service demand) and its evaluation of what is received at the end (service rating). Hence, each $ContextAwareRating$ should represent a service demand, a provider supplying the service for this demand, its rating and the date for the supplied service.

Service demands are represented by $Demand$ class in the ontology. Properties of $Demand$ class are *hasShoppingItem*, *toLocation*, *hasDeliveryType*, *hasDeliveryDuration*, *hasShipmentCost* and *hasPrice*. These properties refer to shopping items, delivery location, delivery type, delivery duration, shipment cost and price, respectively. Some boolean properties are also included in this set of properties: *isRefundable* and *hasConsumerSupport*. These properties indicate whether the transaction is refundable or not and whether consumer support is provided or not. Service consumers represent their service interests using these domain level properties. For example, *hasPrice* property is used to represent the money a consumer is willing to pay for a service. The range of *hasShoppingItem* property is *ShoppingItem* class. This class has properties, *hasQuantity*, *hasUnitPrice* and *hasQuality*. The range of *hasQuality* property is *Quality* class. This class describes quality properties of shopping items. Service providers in the ontology are represented by $ServiceProvider$ class. Each context-aware rating is related to a plain rating value by data type property $hasRating$. Overall, a $ContextAwareRating$ represents a service demand and the quality

of provided service in terms of ratings. A rating can be +1 (positive) or -1 (negative). If the consumer is satisfied with the provided service for its service demand, it gives the provider a positive rating otherwise it gives a negative rating. The date of the supplied services may be important while evaluating the ratings. For example, a service provider may degrade its service quality by time and a service consumer may want to weight context-aware ratings according to their ages. For this reason, $hasDate$ property is added to the ontology. An example of context-aware rating and its representation is given in Example 1 and Figure 3.2, respectively.

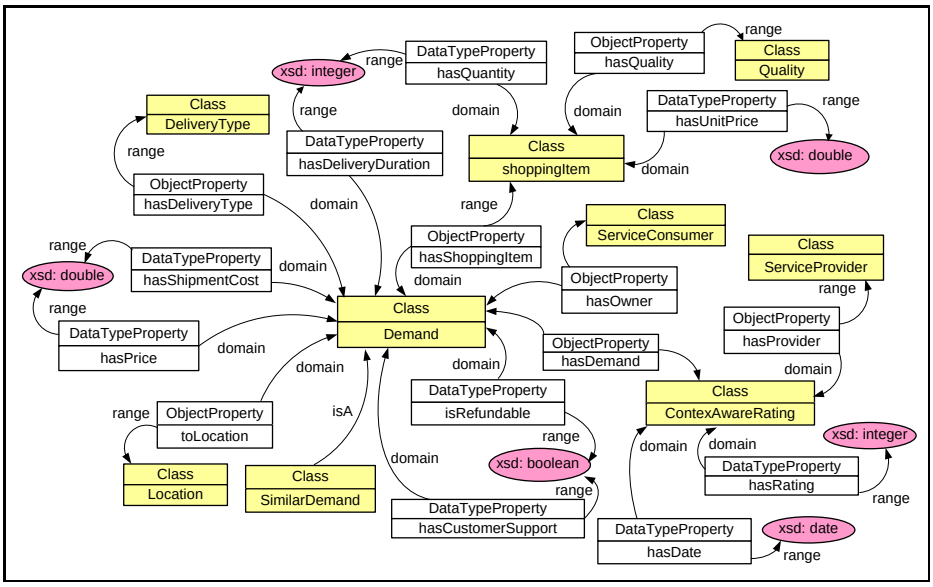

Figure 3.1. Context ontology for online shopping domain

Example 1: In his experience (represented in Figure 3.2), the buyer states that he ordered an IBM ThinkPad T60 notebook from a seller named *TechnoShop* on 15 October 2007. He requested the merchandise to be delivered to New York within 14 days. The service of *TechnoShop* for this demand was *bad*, so it is given a negative rating.

A service consumer may want to communicate to other service consumers with similar demands. But similarity is a subjective concept and may change for each consumer. To allow a consumer to express its description of a similar demand, *SimilarDemand* class is included in the context ontology. *SimilarDemand* class is a subclass of *Demand*. A service consumer

```
<owlx:Individual owlx:name="ContextAwareRatingInstance">
<owlx:type owlx:name="ContextAwareRating" />
<owlx:ObjectPropertyValue owlx:property="hasDemand">
<owlx:Individual owlx:name="demandInstance" />
</owlx:ObjectPropertyValue>
<owlx:DataPropertyValue owlx:property="hasRating">
<owlx:DataValue owlx:datatype="&xsd;int">0</owlx:DataValue>
</owlx:DataPropertyValue>
<owlx:DataPropertyValue owlx:property="hasDate">
<owlx:DataValue owlx:datatype="&xsd;Date">2007-10-15</owlx:DataValue>
</owlx:DataPropertyValue>
<owlx:ObjectPropertyValue owlx:property="hasProvider">
<owlx:Individual owlx:name="TechnoShop" />
</owlx:ObjectPropertyValue>
</owlx:Individual>
```

```
<owlx:Individual owlx:name="demandInstance">
<owlx:type owlx:name="Demand" />
<owlx:ObjectPropertyValue owlx:property="hasOwner">
<owlx:Individual owlx:name="MuratSensoy" />
</owlx:ObjectPropertyValue>
<owlx:ObjectPropertyValue owlx:property="hasShoppingItem">
<owlx:Individual owlx:name="#IBM_ThinkPad_T60" />
</owlx:ObjectPropertyValue>
<owlx:ObjectPropertyValue owlx:property="toLocation">
<owlx:Individual owlx:name="NewYork" />
</owlx:ObjectPropertyValue>
<owlx:DataPropertyValue owlx:property="hasDeliveryDuration">
<owlx:DataValue owlx:datatype="&xsd;Integer">14</owlx:DataValue>
</owlx:DataPropertyValue>
</owlx:Individual>
```

Figure 3.2. A context-aware rating that is about buying a notebook from a seller named TechnoShop

can express what a similar demand is with respect to its similarity criteria using Semantic Web Rule Language (SWRL) [31]. A simple rule for similarity is shown in Figure 3.3. In this rule, the consumer states that a demand is similar demand only if it concerns a book and requires a delivery duration less than or equal to 14 days.

```
<ruleml:imp>
 <ruleml:_head>
    <swrlx:classAtom>
       <owlx:Class owlx:name="#SimilarDemand"/><ruleml:var> DEMAND  </ruleml:var>
    </swrlx:classAtom>
 </ruleml:_head>
 <ruleml:_body>
    <swrlx:DataPropertyValue swrlx:property="#hasDeliveryDuration">
    <ruleml:var>DEMAND</ruleml:var><ruleml:var>DURATION </ruleml:var>
    </swrlx:DataPropertyValue>
      <swrlx:individualPropertyAtom  swrlx:property="&ex;#hasShoppingItem">
      <ruleml:var>DEMAND</ruleml:var><owlx:Individual owlx:name="&ex;#book"/>
    </swrlx:individualPropertyAtom>
    <swrlx:predicateAtom swrlx:predicate="..#ifTrue">
      <owlx:DataValue owlx:datatype="..#string">$1 <= 14 </owlx:DataValue>
      <ruleml:var>DURATION</ruleml:var>
    </swrlx:predicateAtom>
 </ruleml:_body>
</ruleml:imp>
```

Figure 3.3. Example SWRL rule for similar demands

SWRL is introduced as a way to integrate rules with OWL-DL ontologies (see Section 2.1 for details). Unlike other rule languages such as RuleML [57], SWRL is purposely constrained to make automated reasoning more tractable. Hence, using SWRL rules, consumers can represent logical axioms and reasoning on those axioms can be made in a tractable manner. That is, if a consumer has a particular service demand and a list of others' service demands, then it can apply the SWRL rule representing its similar demand definition to select those demands which were similar to that of its own. If the consumer makes its SWRL rule for similar demands public, other consumers can also use this expression of

similarity to reason about whether their past service demands were similar to the demand of the consumer or not. A Description Logic (DL) reasoner with OWL support can be used for the reasoning on similarity [27].

3.1.2. Selecting Service Providers

Let us consider a service consumer that uses context-aware ratings to select appropriate service providers for its current service demand. The consumer collects context-aware ratings, instead of plain ratings, from other consumers so that service demands in these context-aware ratings are similar to the current service demand of the consumer.

After collecting sufficient number of context-aware ratings in a repository, the service consumer weights the context-aware ratings according to the similarity between the service demands within these context-aware ratings and its current service demand. Similarity metrics and computation of similarity depends on the consumer. Then, the set of context-aware ratings are divided into subsets so that each subset belongs to one service provider. Finally, ratings in each subset are averaged using the computed weights and the weighted average is assigned as the aggregated rating of the corresponding service provider. Equation 3.1 shows computation of the aggregated rating, AR_i, given by a consumer for the provider i:

$$AR_i = \sum_j \frac{r_i^j \times w_j \times R_j}{\sum_t r_i^t \times w_t} \tag{3.1}$$

where, j refers to a context-aware rating, w_j is the weight of j, R_j refers the rating within j and r_i^j has value 1 only if i is the service provider within j, otherwise its value is 0. After computing aggregated ratings, the consumer chooses the provider with highest aggregated rating. That is, this formula aggregates ratings using the similarity of their contexts to the current service demand of the consumer. Hence, a provider's aggregated rating is computed as high only if the produced services by the provider for the similar service demands have highly rated by the other consumers.

3.2. Evaluation of the Rating-Based Approach

In order to demonstrate the performance of the proposed methods, we implemented a simulator and conducted exhaustive simulations on it. The simulator is implemented in Java. KAON2[9] is used as OWL-DL reasoner. Simulations are run on a computer with a 1.66 GHz Intel Core Duo CPU and 1.0 GB RAM under Windows OS. Simulations are repeated 10 times in order to increase the reliability. Only the mean values of the results are reported in this section. Although we estimate and report the mean values, these mean values may not reflect the true mean values. The reason is that the estimated mean values may vary from sample to sample. Hence, we may compute a confidence interval that generates a lower and upper limit for the mean values. This interval estimate gives an indication of how much uncertainty there is in our estimate of the true mean values. The narrower the interval, the more precise is our estimate. In order to compute confidence intervals of the mean values, *t-test* is used when the number of samples is small (e.g., 10 samples). Therefore, our simulation results are also analyzed with *t-test* for 95% confidence interval [58]. Our tests show that with 95% probability, the true mean values deviate at most 4% from the calculated mean values.

We conduct simulations to measure the performance of our model in selecting an appropriate service provider. In the simulator, different service provider selection strategies are implemented and compared with each other in terms of achieved satisfaction through numerous experiments. These strategies are as follows:

1. Service provider selection using context-aware ratings (SPS_{CAR}): This strategy is proposed in Section 3.1.
2. Service provider selection using selective ratings (SPS_{SR}): In this strategy, the service consumer uses the plain ratings from other consumer agents. However, the ratings are taken from those agents that have had similar demands with respect to similarity criteria of the agent. That is, SPS_{CAR} and SPS_{SR} actually use the information from the same service consumers for a given decision process. SPS_{SR} uses plain ratings while SPS_{CAR} uses context-aware ratings.

[9] http://kaon2.semanticweb.org

3. Service provider selection using random ratings (SPS_{RR}): This strategy collects plain ratings from randomly chosen service consumers. Most of the previous rating based approaches do not differentiate between the consumers using their previous service interests. Hence, these approaches just collect ratings from consumers that are willing to rate the providers. This strategy represents these approaches. For comparison reasons, number of consumers to be requested for ratings is equal to that of the other strategies, SPS_{SR} and SPS_{CAR}.

There are two important facts to note about the simulations. First, the simulations enforce agents to make decisions based on others' experiences rather than their own previous experiences. This is done on purpose to test how well agents can find information from other sources. Second, as frequently seen in real world, service consumers periodically change their service demands. This is done to mimic variations on context.

3.2.1. Simulation Environment Factors

In our simulations, service characteristics of a service provider are generated as the following. A service space is defined so that all possible services are represented within this service space. Dimensions of the service space and their ranges are tabulated in Table 3.1. Each service provider has a multidimensional region called service region in this service space. This region is randomly generated. The service space and the service regions have 15 dimensions. A service region covers all of the services produced by the service provider. If a consumer that is located in *Istanbul* orders *two* books titled *Anagrams* from the service provider, the service that the provider delivers will be constructed as follows: The properties that are specified (shopping item, quantity and location) will be fixed if they stay within the service region of the provider. For the remaining attributes, the service provider will choose random values making sure that the values stay in the range of its service region. So, for this example, the degree of freedom for generating services will be reduced to 12.

Given the service constraints, the simulation environment generates the demand of the service consumer. To do so, the demand space is constructed by removing dimensions of service space that do not belong to $Demand$ class. Then a random region in this demand

Table 3.1. Dimensions of service space and their ranges

Dimension Name	Type	Range
hasShoppingItem	Integer	1 - 1000
toLocation	Integer	1 - 100
hasDeliveryType	Integer	1 - 6
hasDeliveryDuration	Integer	1 - 60
hasShipmentCost	Double	0 - 250
hasPrice	Double	10 - 11000
hasUnitPrice	Double	1 - 100
hasQuantity	Integer	1 - 100
hasQuality	Integer	1 - 10
isRefundable	Boolean	0 - 1
hasConsumerSupport	Boolean	0 - 1
didRecieveMerchandise	Boolean	0 - 1
hasStockInconsistency	Boolean	0 - 1
isAsDescribed	Boolean	0 - 1
isDamaged	Boolean	0 - 1

space is chosen. The center of this region represents the demanded service. In response to this demand, the chosen provider provides a service. If the provided service for this demand stays within the margins of demand region, then the service consumer gets satisfied, otherwise it gets dissatisfied. The simulation environment guarantees that each demand can be satisfied by at least one service provider.

Next, the simulator creates the similar demand criteria for the service consumer. This is again done by creating a new region (*similar demand region*). Essentially, this is the demand region after some dimensions have been removed. The number of dimensions to be removed and these dimensions are chosen randomly. Service demands staying within the margins of the similar demand region are classified as similar demand by the consumer.

There are two important factors in the simulations:

1. Variations in the context of service demands. As noted before, each service consumer changes its demand characteristics after receiving a service. This is done with a predefined probability (P_{CD}). After changing its demand characteristics, the service consumer collects information for its new service demand. Each service consumer has a probability of requesting a service for any epoch. This probability is uniformly chosen between 0 and 1. In other words, only around 50% of consumers consume a service at a given epoch.
2. Variations in service satisfaction (Subjectivity). Even though a service consumer X regards the service demand of consumer Y as a similar demand, this does not mean that Y and X share the same satisfaction criteria. Therefore, a service dissatisfying Y may satisfy X and vice versa. This fact is also imitated in the simulations. A parameter called ratio of subjectivity (R_{subj}) defines what ratio of the service consumers having similar service demands with respect to similarity criteria of X will have satisfaction criteria conflicting with the satisfaction criteria of X. So, ratings of these consumers will probably mislead the consumer X during service selection.

The simulation environment is setup with 10 service providers and 200 service consumers. Simulations are run for 100 epochs. When the simulations start, agents do not have any prior experiences with service providers. As the simulations advance, agents gain experiences and collect context-aware ratings.

When there is no subjectivity ($R_{subj} = 0$) in the environment, each service demand is satisfied by two service providers in the system. Therefore, probability of random service selection is 0.2 for this case. However, when there is some subjectivity in the environment ($R_{subj} > 0$), exactly one service provider can satisfy a service demand. In this case, probability of random service selection decreases to 0.1. To sum up, for any service demand, there exists at least one service provider that can satisfy this demand completely. Hence, a successful service selection approach should always find a satisfactory service provider for any service demand, while random selection of the providers will not achieve more than 20% success on the average in any setting.

3.2.2. Simulation Results

With this setup, we are interested in understanding how the variations in context of service demands (P_{CD}) and the differences in consumers' satisfaction criteria (R_{subj}) will affect finding service providers successfully. To study this, we will have service selection approaches that use plain ratings from consumers (SPS_{RR} and SPS_{SR}) and the proposed service selection approach that uses context-aware ratings instead of plain ratings.

We initially run simulations for the case that consumers do not change their service demands over time ($P_{CD} = 0$) and satisfaction criteria of the consumers having similar service demands are the same ($R_{subj} = 0$). In this setting, we observe that, satisfaction ratios of SPS_{CAR} and SPS_{SR} approach 1.0. This means that these approaches are equally good in this setting and they lead to satisfactory service selections almost all the time. On the other hand, ratio of decisions resulted in satisfaction is constant around 0.2 for SPS_{RR}, which is the performance of random service selection. Note that, in this setting, consumers never change their service demands and if two consumers have similar contexts, any provider satisfying one of them will also satisfy another. Hence, ratings taken from consumers having similar contexts will always lead to satisfaction. Unfortunately, these assumptions are not reasonable for many real-life scenarios where consumers change their service demands over time ($P_{CD} > 0$) and satisfaction criteria of the consumers are different even though their service interest are the same ($R_{subj} > 0$). Therefore, we examine performances of the service selection approaches for more realistic settings.

Figure 3.4 shows our simulation results when the parameters are set as P_{CD}=0.2 and R_{subj}=0. This means that a consumer will change its service demand with a probability of 0.2 after receiving a service. Figure 3.4 indicates that the performance of SPS_{SR} decreases sharply with time, whereas the performance of SPS_{CAR} is constant around 100% satisfaction. The decrease in the performance of SPS_{SR} is due to the fact that ratings of a consumer reflect the aggregation of its past transactions for all of its previous demands. In other words, as consumers change their demands, their ratings become more misleading than before. However, SPS_{CAR} can differentiate ratings for different contexts and evaluates each rating within its scope. Hence, as seen in the Figure 3.4, SPS_{CAR} leads to decisions

with 100% satisfaction but satisfaction ratio of SPS_{SR} decreases and approaches to that of SPS_{RR} as consumers change their demands. Satisfaction ratio is low and constant around 0.2 for SPS_{RR}. This is equal to the performance of the random service selection.

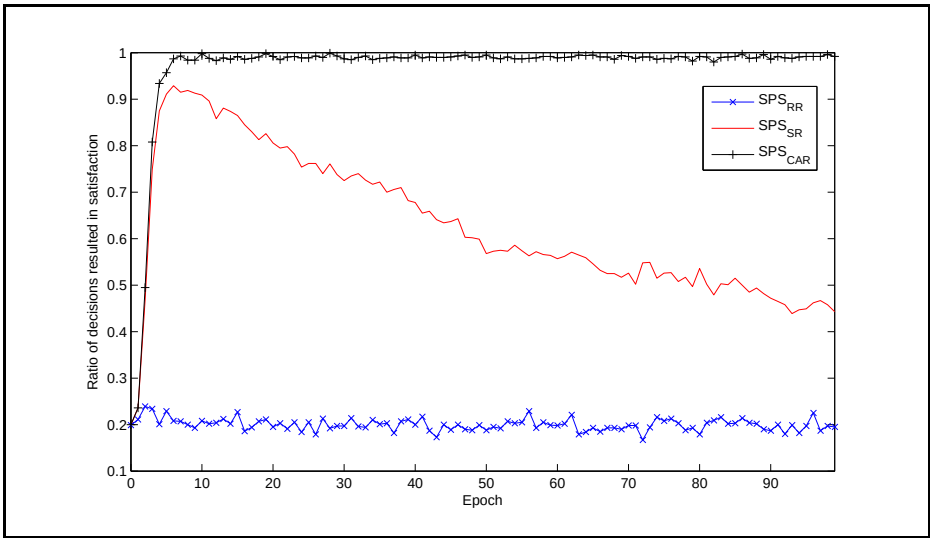

Figure 3.4. Consumers change their demands with probability 0.2 (P_{CD}=0.2), but have the same satisfaction criteria if their service demands are similar (R_{subj}=0)

Figure 3.5 shows the simulation results for P_{CD}=0, and R_{subj}=0.5. This is the case when the consumers do not change the context of their service demands and approximately half of the consumers owning similar contexts will have conflicting satisfaction criteria. For this setting, SPS_{CAR} and SPS_{SR} have almost the same performance. Moreover, the performance of SPS_{SR} does not decrease over time, because consumers using SPS_{SR} do not change their service demands over time (P_{CD}=0). Approximately 50-60% of the selected services leads to the satisfaction of service consumers both for SPS_{SR} and SPS_{CAR}, because approximately half of the ratings for the same context will reflect the taste of the consumer making service decision (R_{subj}=0.5). So, the consumer will make a wrong decision around half of the time. For these settings, satisfaction ratio of SPS_{RR} is constant around 0.1, which is equal to success probability of random service selection. Hence, we can conclude that a rating-based approach does not result in a service selection performance better than that of the random service selection in this environment if the raters are selected randomly.

Unlike SPS_{SR}, SPS_{CAR} is robust to the variations in context of service demands

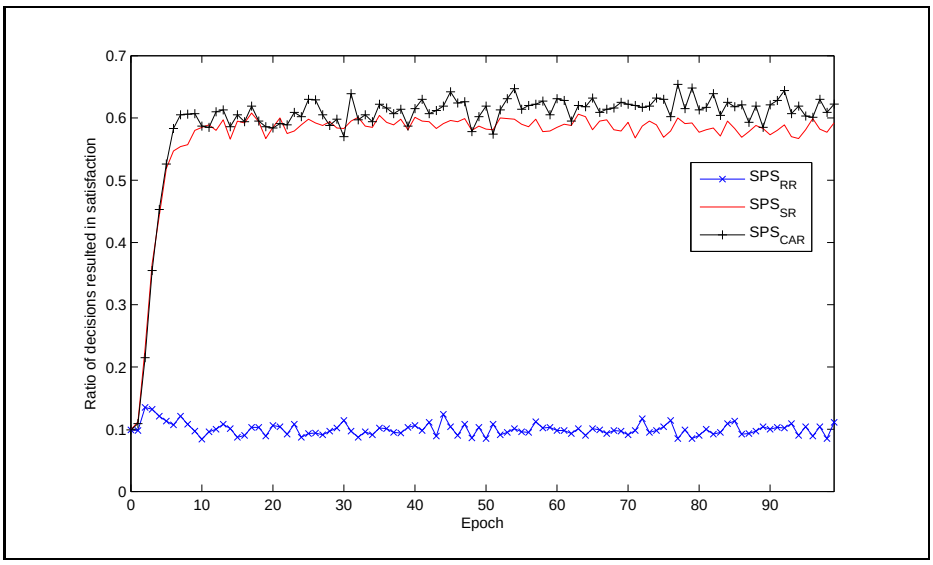

Figure 3.5. Half of the consumers having similar demands have conflicting satisfaction criteria (R_{subj}=0.5), but the consumers do not change their service demands over time (P_{CD}=0)

(P_{CD}), because it can differentiate between the ratings that are given in different contexts. That is, it aggregates only the ratings that are given in the context of current service demand, while SPS_{SR} considers each rating equally even though they are given in different contexts. Therefore, performance of SPS_{CAR} is insensitive to P_{CD} parameter. On the other hand, both SPS_{SR} and SPS_{CAR} are seriously affected by the subjectivity (R_{subj}), because both of these approaches depend on the subjective opinions by using ratings. Figure 3.6 shows simulation results for the parameters P_{CD}=0.2 and R_{subj}=0.5. Performance of SPS_{CAR} is similar in both Figure 3.5 and Figure 3.6, because in both cases environment is highly subjective; half of the consumers having similar service interests give positive ratings to different providers ($R_{subj} = 0.5$). However, the performance of SPS_{SR} is lower than that of SPS_{CAR} and continuously decreases over time, because consumers' ratings become more misleading as they change the context of their service demands ($P_{CD} = 0.2$).

Our experiments show that, using SPS_{CAR}, consumers can successfully represent the context of their ratings. Hence, each rating is evaluated in its scope to lead satisfactory service selections when there is no subjectivity in the environment ($R_{subj} = 0$). Our simulations also show that, in subjective environments ($R_{subj} > 0$), representing context-information to-

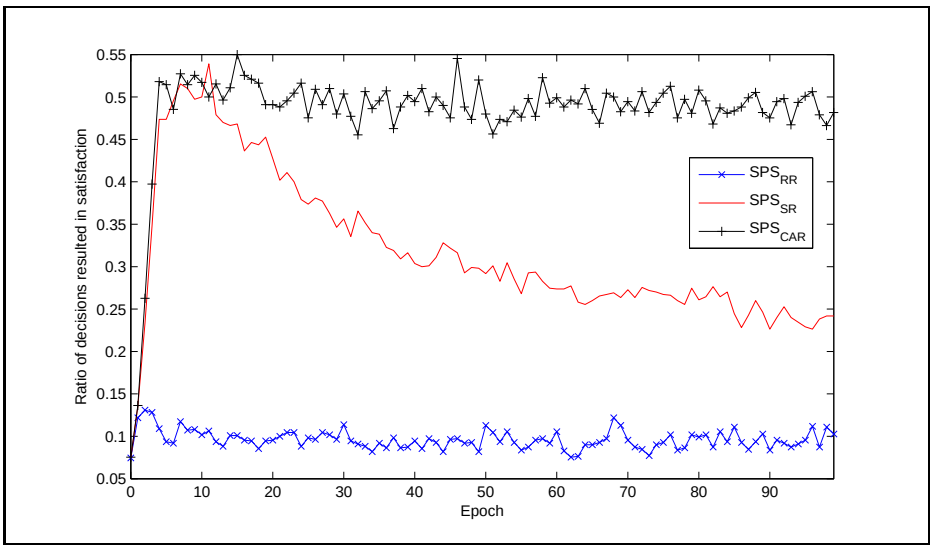

Figure 3.6. Consumers change their service demands (P_{CD}=0.2) and their satisfaction criteria are conflicting (R_{subj}=0.5)

gether with the ratings is not enough for successful service selection, because the ratings reflect the subjective opinions of the consumers. Therefore, in the next section, we propose another representation that is not only context-aware but also robust to subjectivity in the environment.

3.3. An Experience-Based Approach for Context-Aware Service Selection

By using context-aware ratings, consumers can explicitly express the scope of ratings and can use ratings for context-aware service selection. However, ratings will reflect the subjective opinion of the raters. In order to minimize the subjectiveness of rating-based approaches, we propose to use an objective experience-based approach in this section. In this approach, experience of a consumer with a provider is represented using ontologies. This representation contains the requested service and the supplied service in detail. Example 2 demostrates a simple experience. By using experiences, other consumers can evaluate the supplied services with respect to their own evaluation criteria.

Example 2: In his experience, the buyer states that he ordered an IBM ThinkPad T60 notebook from a seller named *TechnoShop* on 15 October 2007. He requested the merchandise

to be delivered to New York within 14 days. The seller received $700 for the product and delivered the merchandise within 7 days without requesting any extra money for shipping. However, the delivered product was not refundable and TechnoShop did not provide any customer support.

In order to represent experiences, the ontology in Figure 3.1 is extended appropriately. The new ontology is called the *experience ontology*. The experience ontology covers the fundamental concepts (such as demand, service, commitment and experience), which exist in the base level ontology and domain specific concepts and properties, which exist in the domain level ontology. Using these concepts and properties, a service consumer can express the details of its dealings with different service providers.

3.3.1. Base Level Ontology

The base level ontology (Figure 3.7) consists of the domain-independent infrastructure of the experience ontology. The main class in the base level ontology is the *Experience* class. Instances of this class represent the experiences of service consumers in the system. As in the real life, an experience in the ontology contains information about what a service consumer has requested from a service provider and what the service consumer has received at the end. To conceptualize the service demand and the received service of the consumer, *Demand* and *Service* classes are included in the base level ontology. Both the demand and the supplied service concepts are descriptions of a service for a specific domain and hence share a number of properties. These shared properties are captured in the *Description* class in the base level ontology. The domain level ontology contains extensions to this class. Domain-dependent properties of *Description* class can be used to describe service demands, supplied services, responsibilities and fulfillments of sides during transactions. These properties are shown in domain level ontology (Figure 3.8).

Each *Description* class has a date and an owner that is also represented as a class. For a demand, the owner is a service consumer and for a service, the owner is a service provider. The date value keeps the date of demanded service or the provided service.

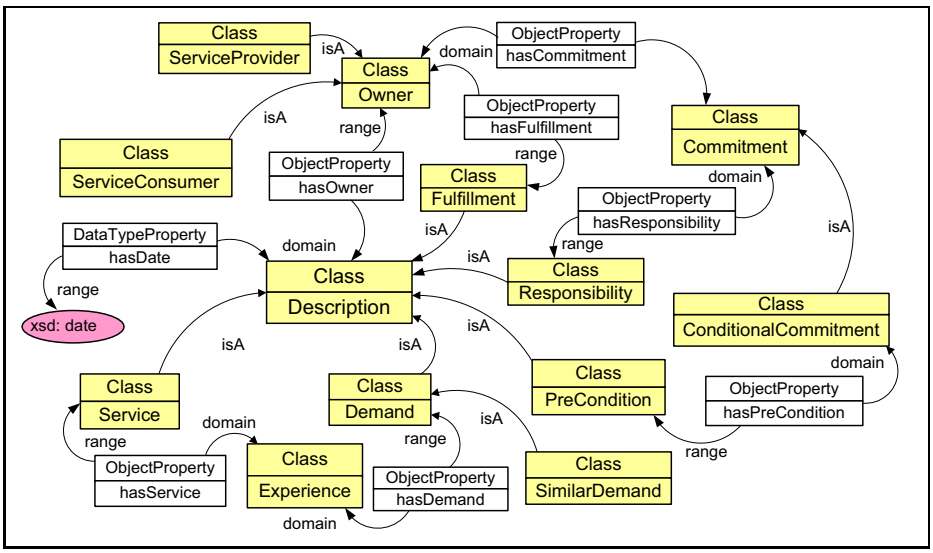

Figure 3.7. Base level ontology

An owner may have commitments toward others to carry out responsibilities [59]. A commitment always has an instance of responsibility. This means that the owner of the commitment is responsible for the realization of conditions described in the responsibility instance. Example 3 demonstrates a simple responsibility instance. *Commitment* and *Responsibility* classes are used to express commitments and responsibilities, respectively in the experience ontology. Fulfillments are accomplishments of responsibilities and are denoted with the *Fulfillment* class. Owners of responsibilities or fulfillments can be service consumers or providers depending on the context.

Example 3: Consider a service provider that is responsible for delivering particular goods to New York City with a shopping cost of $5. In the ontology, this can be represented as an instance of a *Commitment* class, where the instance of the *Responsibility* of the commitment has *toLocation* property referring to New York City and has *hasShipmentCost* property referring to $5.

Transactions between the consumers and providers are usually based on business contracts. The contracts can be represented by conditional commitments. Unlike commitments, conditional commitments have preconditions. For example, a conditional commitment $CC(X, Y, P, Q)$ denotes that if the precondition P is carried out by Y, X will be

committed to carry out responsibility Q. In this definition, Y is the owner of the precondition and X is the owner of responsibility. *ConditionalCommitment* and *Precondition* classes are used in the ontology to specify conditional commitments and preconditions. Conditional commitments can be used to represent contracts and offers made by service consumers and providers. An example case is demonstrated in Example 4.

Example 4: A service consumer can offer to pay an additional $100 for one week early delivery. If the provider makes the shipment one week early, the consumer is committed to pay $1100 for a product whose actual value is only $1000. Service providers can also make offers using conditional commitments.

3.3.2. Domain Level Ontology

Since the base level ontology deals only with domain independent concepts, a second ontology is necessary to capture domain dependent concepts and properties. The domain level ontology is developed for this purpose. The core class of domain level ontology is *Description* class. Domain specific properties of *Description* class are used to describe service demands, supplied services, responsibilities and fulfillments of parties during transactions. A domain level ontology for online shopping is shown in Figure 3.8.

The properties of the *Description* class in this ontology are the same as the properties of the *Demand* class in the context ontology, which is explained in Section 3.1.1. However, the properties of *Description* class have slightly different meanings for different sub-classes of *Description* class. For example, *hasPrice* property refers to the money a consumer is willing to pay for a service if it is used to describe a service demand. This property refers to the money the consumer is requested to pay for the supplied service if it is used to describe provided service. If *hasPrice* is used to describe a responsibility, *hasPrice* refers to the money a service consumer promises to pay for a service or it refers to the money a service provider will accept for a service, depending on the owner of the responsibility. For the first case, service consumer is the owner of responsibility and for the second case owner is service provider. If *hasPrice* is used to describe a fulfillment of a service consumer, *hasPrice* refers

to the money paid by the service consumer for the specified service.

The properties of *Description* have different semantics also for *Precondition* class. For example, if *hasPrice* property is used to describe precondition P in conditional commitment $CC(X, Y, P, Q)$, this property refers the amount of money Y should pay out for X to carry out responsibility Q. If the precondition is realized, owner of the conditional commitment will be responsible for the realization of the responsibilities described by the *Responsibility* instance Q.

In addition to properties of *Description* class, concepts in the ontology may also have domain-specific properties that other concepts do not have. For example, for consumer goods domain, properties such as *didRecieveMerchandise*, *hasStockInconsistency*, *isAsDescribed* and *isDemaged* are included as properties of $Service$ class in domain level ontology. The property *didRecieveMerchandise* is used to state whether the merchandise is received by the consumer or not. The semantics of *hasStockInconsistency* property is derived from the comments of online shopping consumers. During shopping, a product is seen in stock on the Web. However, after ordering, delivery of the product takes very long time and the consumer is informed that the product is not in the stock. This situation is expressed by a service consumer using *hasStockInconsistency* property. Other properties can also be added to this ontology.

Service consumers maintain, exchange, and interpret experiences related to the providers. These experiences are expressed using the OWL ontology proposed in this section. Therefore, they can be interpreted easily by the agents using an OWL reasoner such as *KAON2* or *Pellet* [60]. In Figure 3.9, we demonstrate how the experience in Example 2 can be represented using the proposed *experience ontology*.

3.3.3. Exchanging Experiences

A consumer society emerges as a result of consumers' need to retrieve experiences of other consumers. Initially, each service consumer knows only a subset of all consumers in the society and lists these consumers in an *acquaintance list*. An acquaintance list is a dynamic

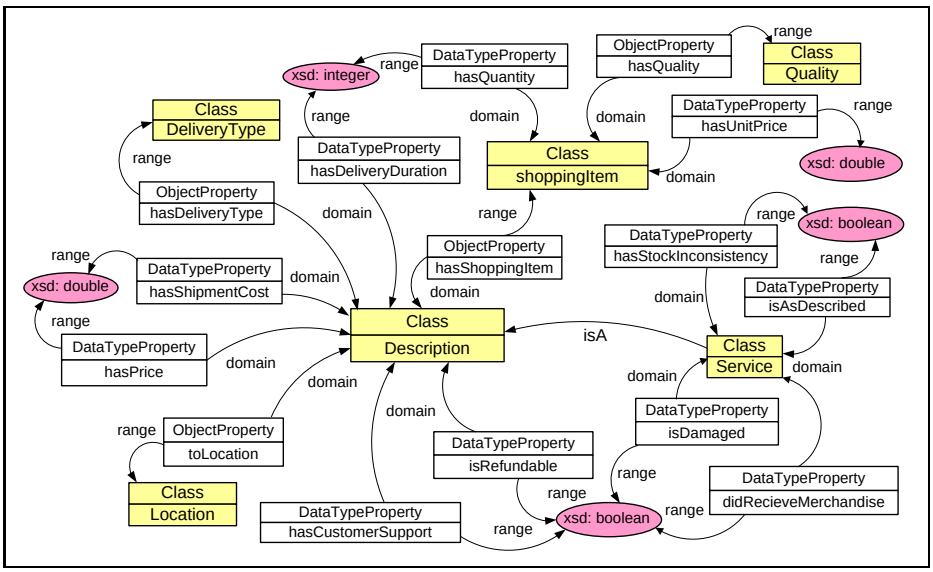

Figure 3.8. Domain level ontology for online shopping

list of service consumers having service demands classified as similar demand by the owner of the list. When a new service consumer joins the society, its acquaintance list is populated with a small number of randomly chosen service consumers. Note that the acquaintance lists are not symmetric: because X is on Y's acquaintance list does not mean that Y will be on X's list.

Each consumer compiles other agents' experiences in an *experience repository*. Each time a service consumer makes a decision, it uses the experiences in this repository. The service consumer refreshes and updates its experience repository periodically by removing old experiences and adding newly found experiences.

When the system starts to function, the service consumers do not have any experiences. When a service consumer X needs experiences for reasoning on which service provider to choose, it should discover other service consumers having a similar demand and should populate its acquaintance list with those service consumers and their service demands. In order to accomplish this, the consumer follows the procedure summarized the algorithm in Figure 3.10.

In this algorithm, when a service consumer decides to receive a service, it checks its

```
<owlx:Individual owlx:name="ExperienceInstance">
  <owlx:type owlx:name="Experience" />
  <owlx:ObjectPropertyValue owlx:property="hasDemand">
    <owlx:Individual owlx:name="demandInstance" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hasService">
    <owlx:Individual owlx:name="serviceInstance" />
  </owlx:ObjectPropertyValue>
</owlx:Individual>
```

```
<owlx:Individual owlx:name="demandInstance">
  <owlx:type owlx:name="Demand" />
  <owlx:ObjectPropertyValue owlx:property="hasOwner">
    <owlx:Individual owlx:name="MuratSensoy" />
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasDate">
    <owlx:DataValue owlx:datatype="&xsd;Date">2007-10-15</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hasShoppingItem">
    <owlx:Individual owlx:name="#IBM_ThinkPad_T60" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="toLocation">
    <owlx:Individual owlx:name="NewYork" />
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasDeliveryDuration">
    <owlx:DataValue owlx:datatype="&xsd;Integer">14</owlx:DataValue>
  </owlx:DataPropertyValue>
</owlx:Individual>
```

```
<owlx:Individual owlx:name="serviceInstance">
  <owlx:type owlx:name="Service" />
  <owlx:ObjectPropertyValue owlx:property="hasOwner">
    <owlx:Individual owlx:name="TechnoShop" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hasShoppingItem">
    <owlx:Individual owlx:name="#IBM_ThinkPad_T60" />
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasDeliveryDuration">
    <owlx:DataValue owlx:datatype="&xsd;Integer">7</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:DataPropertyValue owlx:property="recivedMerchandise">
    <owlx:DataValue owlx:datatype="&xsd;boolean">true</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:DataPropertyValue owlx:property="isRefundable">
    <owlx:DataValue owlx:datatype="&xsd;boolean">false</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasCustomerSupport">
    <owlx:DataValue owlx:datatype="&xsd;boolean">false</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasShipingCost">
    <owlx:DataValue owlx:datatype="&xsd;Integer">0</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:DataPropertyValue owlx:property="hasPrice">
    <owlx:DataValue owlx:datatype="&xsd;Integer">700</owlx:DataValue>
  </owlx:DataPropertyValue>
</owlx:Individual>
```

Figure 3.9. An experience that is about buying a notebook from a seller named TechnoShop

experience repository (Lines 1-5). In order to make a reliable decision, the service consumer should compute the minimum number of experiences for decision making within a 99% confidence interval $[58, 61]$ (Line 3). If the number of experiences in the repository is not enough to perform this computation, the service consumer collects new experiences (Lines 6-25). However, in order to collect new experiences, the consumer should have sufficient number of acquaintances, so that it can ask for their experiences. For this reason, the consumer checks the number of its acquaintances (Lines 7-9). If it does not have sufficient number of acquaintances, it should increase the number of its acquaintances (Lines 10-19).

To discover new acquaintances and collect interested experiences, consumers use a protocol. The protocol is based on three message types: *Peer Discovery Message (PDM)*, *Request for Acquaintances Message (RAM)* and *Request for Experience Message (REM)*. In order to learn new acquaintances, a service consumer sends PDM or RAM messages. Both PDM and RAM messages contain a SWRL rule that expresses the similar demand criteria of the originator of the message (Line 6, Line 10 and Line 15). When a consumer Y receives a PDM message, it checks if its service demands are similar to that of the originator X. If so, it notifies X and X adds Y as a new acquaintance entry in its acquaintance list. This entry contains identity of Y and its demands classified as similar demand by the similarity criteria of X. The consumer Y also forwards the request to a set of service consumers in its acquaintance list. Y selects consumers having demands similar to demand of the originator to forward the request. If there is no such consumer, Y randomly selects consumers from

its acquaintance list. How long the request is going to be forwarded is controlled using a time-to-live field. All other consumers that receive the request act the same way Y does. When Y receives a RAM message from the originator X, it checks its acquaintance list for entries containing consumers having demands similar to the demand of X. Then, Y sends these entries to X. So, X can add these entries to its acquaintance list.

```
 1: isShopping = decideShopping()
 2: if (isShopping) then
 3:    Nexp = getRequiredNumberOfExperiences()
 4:    ExpSize = ExperienceRepository.size()
 5:    while (ExpSize < Nexp) do
 6:       similarity = createSWRLRuleForSimilarity()
 7:       Nacq = getMinimumNumberOfAcquaintances(N)
 8:       AcqSize = AcquaintanceList.size()
 9:       if (AcqSize < Nacq) then
10:          pdm = createPDM(similarity)
11:          newAcquaintances = propagateMessage(pdm);
12:          AcquaintanceList.add(newAcquaintances)
13:          AcqSize = AcquaintanceList.size()
14:          while (AcqSize < Nacq) do
15:             ram = createRAM(similarity)
16:             newAcquaintances = propagateMessage(ram);
17:             AcquaintanceList.add(newAcquaintances)
18:             AcqSize = AcquaintanceList.size()
19:          end while
20:       end if
21:       rem = createREM(similarity)
22:       newExperiences = propagateMessage(rem);
23:       ExperienceRepository.add(newExperiences)
24:       ExpSize = ExperienceRepository.size()
25:    end while
26:    SelectServiceProvider(ExperienceRepository)
27: end if
```

Figure 3.10. Algorithm of consumer agents

The service consumer populates it acquaintance list through PDM and RAM messages (Lines 10-12 and Lines 15-17). After having sufficient number of acquaintances, the consumer uses REM to collect new experiences (Lines 21-24). An REM message also contains a rule for expressing similar demand criteria of the sender. When service consumer Y gets an REM message from service consumer X, it evaluates its demands in its experiences using the similarity criteria in the REM. Later, it can send its experiences to X if the experiences

have similar demands with respect to similarity criteria encapsulated in the REM, so that X can populate its experience repository with these experiences. After collecting sufficient number of experiences, X uses the experiences in its repository for decision making (Line 26). Next section describes this procedure.

The protocol that is summarized above is also used by our rating-based approach in Section 3.1 to gather context-aware ratings. The only difference is that REM messages are used to request context-aware ratings instead of experiences.

3.3.4. Service Selection Using Experiences

In experience-based service selection, first a consumer collects related experiences from other consumers as explained in Section 3.3.3. For example, if the consumer needs to buy a notebook, it searches for the experiences that are related to "buying a notebook". After collecting related experiences, the consumer evaluates each experience using its satisfaction criteria. Each consumer has an internal taste function F_{taste} (namely satisfaction criteria) to evaluate its transactions with the service providers in the context of its service demands. In real life, the taste of a consumer may change over time. Hence, this function should be time dependent. We assume that taste function of the consumer is unknown to other consumers. In a real-life application, a consumer agent can easily elicit its taste function from its human user using a user interface. Once the consumer has the taste function, it can easily compute its expected level of satisfaction for a specific transaction given the service demand and the supplied service within the transaction. Hence, using the taste function, the consumer can also interpret an experience and compute its level of satisfaction using the data in the experience. In other words, the consumer can produce its expected level of satisfaction for the experience by asking itself how satisfied it would be, had it lived the experience under consideration. Example 5 demonstrates how experiences can be interpreted differently by different consumers.

Example 5: Consider the experience in Figure 3.9 (explained in Example 2) and assume that there are two different consumers (*Bob* and *Lucy*) who received this experience. For Bob,

delivery duration and price are crucial whereas customer support or being refundable are not important. On the other hand, for Lucy, being refundable and having customer support are indispensable. Therefore, for Bob, *TechnoShop* is a very good provider and deserves a good rating, because it delivers products within one week without requesting any extra money. However, for Lucy, *TechnoShop* is not preferable. However, by plain ratings, Bob's positive ratings of *TechnoShop* would have misled Lucy.

Information in the experiences can be used for the modeling of provider behaviors for different service demands. For this purpose, parametric classification methods such as the Multivariate Gaussian Model (GM) can be used. Experience data can also be used by non-parametric methods such as Case-Based Reasoning (CBR) for service selection. In this section, we explain how these methods can be used to select appropriate service providers.

3.3.4.1. Decision Making Using GM. In this approach, a service consumer models each service provider using the experience data available in its repository and selects a provider with the highest probability to satisfy its needs. For this purpose, the consumer uses parametric classification and builds a multidimensional Gaussian model for each service provider [62], as follows.

Demand and service specifications within experiences are received in the form of ontologies, but then they are converted into the internal representation of the service consumer. Demand and commitment information in each experience is represented as a vector. Each field in this vector is extracted from the experience ontology. These fields correspond to property values in the experience ontology such as service price. Then, supplied service for this demand is classified as *satisfied* or *dissatisfied* with respect to satisfaction criteria of the consumer using the taste function and ontological reasoning [22]. After that the *(vector, class)* pairs are used as training set. For each class, covariance and mean values are extracted from the training set. Then, a discriminant function is defined to compute the probability of satisfaction [62]. The service consumer performs this computation for every service provider and chooses the provider with the highest satisfaction probability.

The equations below formulate this computation. In these equations, C_i refers to the i^{th} class. Note that there are two classes; the first class is *satisfied* and the second class is *dissatisfied*. For the i^{th} class, mean and covariance are represented by μ_i and Σ_i, respectively. Equation 3.2 formulates the class likelihood $p(X|C_i)$; the probability that the demand X is observed in class C_i. Equation 3.3 formulates the posterior probability $p(C_i|X)$; the probability that the demand X is in class C_i. In Equation 3.3, $p(X)$ refers to the probability that demand X is observed and it is computed as $p(X) = p(X|C_1) + p(X|C_2)$ in this case. Similarly, $p(C_i)$ refers to the prior probability that the class C_i is observed. Lastly, the discriminant function for the i^{th} class, $g_i(X)$, is formulated as in Equation 3.4. The higher the computed $g_1(X)$ value is, the more likely the provider under consideration satisfies X.

$$p(X|C_i) = \frac{exp\left[-\frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i)\right]}{(2\pi)^{2/d}\left|\Sigma_i\right|^{1/2}} \tag{3.2}$$

$$p(C_i|X) = \frac{p(X|C_i)p(C_i)}{p(X)} \tag{3.3}$$

$$g_i(X) = log[p(C_i|X)] + log[p(C_i)] \tag{3.4}$$

Consider that Bob in Example 5 wants to buy a notebook. For this purpose, first he collects experiences about the notebook providers and then estimates the probability of satisfaction for each provider as described above. In this example, Bob needs to compute the probability that *TechnoShop* produces a satisfactory service. Initially, Bob uses his satisfaction criteria to evaluate the supplied services within the collected experiences about *TechnoShop*. He labels each experience as *satisfied* or *dissatisfied*. Using a Gaussian distribution function, Bob estimates the probabilities that his current demand is observed among the satisfied demands and dissatisfied demands as in Equation 3.2. Those probabilities are denoted as $p(X|C_1)$ and $p(X|C_2)$ respectively, where X is the data vector representing Bob's current demand about buying a notebook. Then, using Bayes' rule, Bob estimates the probability that *TechnoShop* produces a satisfactory service given his current service demand, denoted as $p(C_1|X)$. Lastly, Bob calculates the discriminant function $g_1(X)$ to quantify the preferability of *TechnoShop* using Equation 3.4 and uses this value to decide about *TechnoShop*.

3.3.4.2. Decision Making Using CBR. CBR is an approach for problem solving and learning. In CBR, existing problems and their solutions are encapsulated into a case structure and are stored in a case-base. When a new problem is encountered, the most similar past cases are retrieved from the case-base and their solutions are used or modified to conform to the new situation [63]. The intuition is that if two problems are similar, the solutions to these problems will probably be similar, too.

The most important challenge in the CBR is the selection of metrics for the similarity, since the performance of CBR systems critically depends on these metrics. Also, most CBR approaches are centralized. This implies that illy constructed metrics for the similarity could drastically affect the performance of the whole system. The proposed approaches in the previous sections can be combined to construct a context-aware, flexible and distributed CBR approach for the service selection. In this approach, each consumer uses its consumer society as a distributed case-base. Additionally, unlike the classic CBR systems, each consumer can represent its own similarity metrics using an OWL ontology and SWRL rules. Using these well-defined similarity metrics, the consumer queries the consumer society for similar experiences (cases) using the procedure explained in Section 3.3.3. After retrieving the similar experiences, the consumer computes a score for each retrieved experience. The computation depends on the following factors:

1. Recency: The new experiences are preferred over old experiences since they are likely to hold again in the near future. For this reason, each experience is assigned a recency value. The newer the experience, the larger the recency value.
2. Similarity: This is a factor that measures the similarity of the current demand with the examined experience. The similarity value ranges between 0 and 1, where 0 denotes total difference and 1 denotes identical demands.
3. Satisfaction: This is an important factor that measures how satisfied the current consumer agent would be, had it lived the examined experience. The consumer evaluates the supplied service depending on its current service demand and its own satisfaction criteria and obtains its expected degree of satisfaction.

We combine these factors using the formula below:

$$s_i = recency_i \times sim_i \times sat_i \tag{3.5}$$

where, s_i is the computed score for the experience i, $recency_i$ is the recency factor, sim_i is the similarity factor and sat_i is the satisfaction factor. After computing the scores for each experience, the consumer picks the experience with the highest score and selects the provider supplying the service within this experience. The proposed CBR system uses OWL ontologies for the representation of cases (experiences) and distributes the case-base such that it can be searched with individually defined similarity metrics.

3.4. Evaluation of Experience-Based Approaches

In this section, we evaluate the proposed experience-based service selection techniques. In the simulations, we devise two more strategies that use experience-based service selection:

1. Service provider selection using Gaussian Model (SPS_{GM}): This strategy is proposed in Section 3.3.4.1.
2. Service provider selection using CBR (SPS_{CBR}): This strategy is proposed in Section 3.3.4.2.

By varying the aforementioned factors (i.e., service demand and service satisfaction), we are interested in understanding the strengths and weaknesses of the proposed strategies, especially in terms of the ratio of satisfaction and the required computational time for selecting services. The simulation environment is set up as explained in Section 3.2. In our simulations, commitments and contracts between the consumers and the providers are not included. Simulations are repeated 10 times in order to increase the reliability. Only the mean values of the results are demonstrated in this section. The simulation results are analyzed with *t-test* for 95% confidence interval [58]. Our tests show that with 95% probability, the true mean values deviate at most 3% from the calculated mean values.

3.4.1. Experience-Based Approach: GM

This section summarizes the results of the simulations for SPS_{GM}. Figure 3.11 shows the simulation results for P_{CD}=0, and R_{subj}=0.5. In these settings, consumers do not change their service demands over time, but for each consumer, approximately half of the consumers having similar service demands will have satisfaction criteria conflicting with that of the consumer. Hence, these consumers provide misleading ratings. Figure 3.11 indicates that only around 50% of the services lead to satisfaction of service consumers if these service consumers use SPS_{SR}. On the other hand, service consumers using SPS_{GM} are almost always satisfied with the supplied services, because unlike rating-based approaches, it does not depend on the subjective opinions of the consumers.

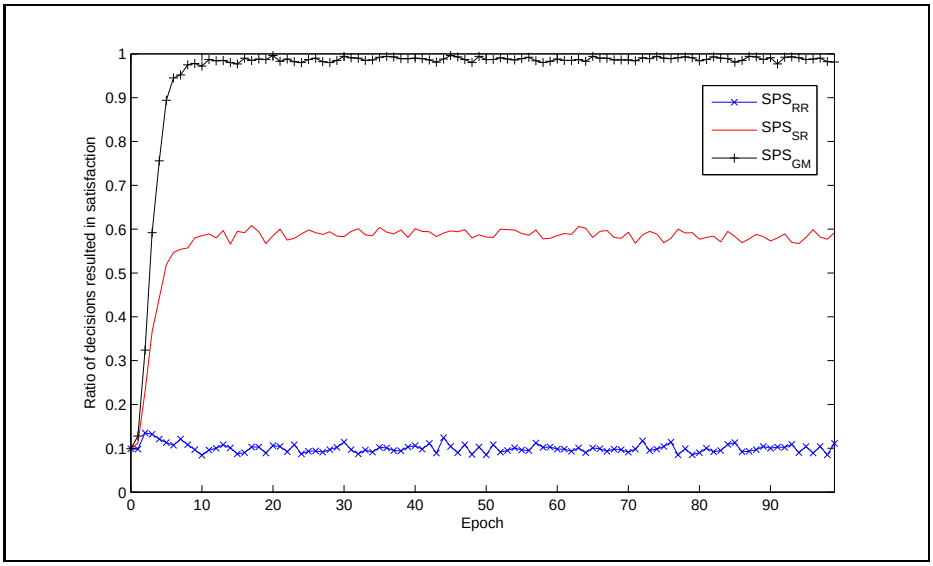

Figure 3.11. Half of the consumers having similar demands have conflicting satisfaction criteria (R_{subj}=0.5), but the consumers do not change their service demands over time (P_{CD}=0)

Figure 3.12 shows simulation results for parameters P_{CD}=0.2 and R_{subj}=0. The value of P_{CD} (0.2) implies that the consumers change their service demands with probability 0.2 after making a service decision. For these settings, SPS_{GM} leads to decisions with 100% satisfaction but satisfaction ratio of SPS_{SR} decreases over time as consumers change their service demands. Simulation results of SPS_{CAR} for the same settings (see Figure 3.4) are the same as the results of SPS_{GM} in Figure 3.12. This means that performances of SPS_{CAR}

and SPS_{GM} are similar when there is no subjectivity in the environment (R_{subj}=0). That is, both of these approaches are equally successful when the consumers do not have conflicting satisfaction criteria for similar demands.

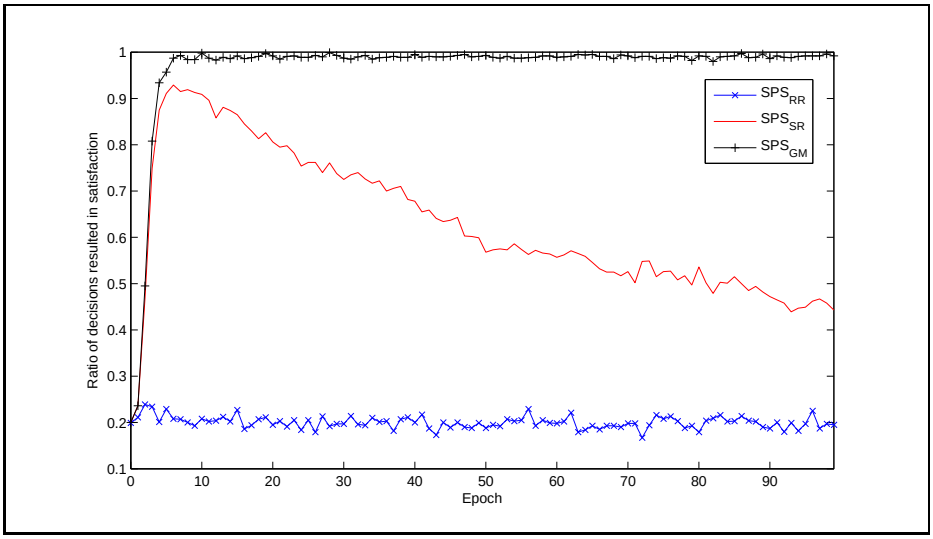

Figure 3.12. Consumers change their demands with probability 0.2 (P_{CD}=0.2), but have overlaping satisfaction criteria (R_{subj}=0)

Figure 3.13 shows simulation results when there is subjectivity and variation in context (P_{CD}=0.2 and R_{subj}=0.5). For these settings, performance of SPS_{SR} decreases further. However, for SPS_{GM}, satisfaction ratio is around 1.0 after 7^{th} epoch (before 7^{th} epoch, there are not enough experiences accumulated in the environment for the modeling of service providers).

In order to see the influences of subjectivity and context variations on the satisfaction ratios achieved by SPS_{GM} and SPS_{SR} strategies, simulations are repeated for different R_{subj} and P_{CD} values. Average ratios of satisfactions for these simulations are shown in Table 3.2 and Table 3.3. The tables indicate that the performance of SPS_{SR} decreases considerably with an increase in the value of R_{subj} or P_{CD}. Combinatorial effect of these parameters on the performance of SPS_{SR} is a more dramatic decrease in the service selection performance. However, the simulations show that the proposed method, SPS_{GM}, is robust to subjectivity and variation on context, because it is context-aware and does not depend on the subjective opinions of others (e.g., ratings). Unlike SPS_{SR}, the performance of SPS_{GM}

does not change with changing R_{subj} and P_{CD} values, as well as the achieved satisfaction is around 100% if service consumers use SPS_{GM} for decision making.

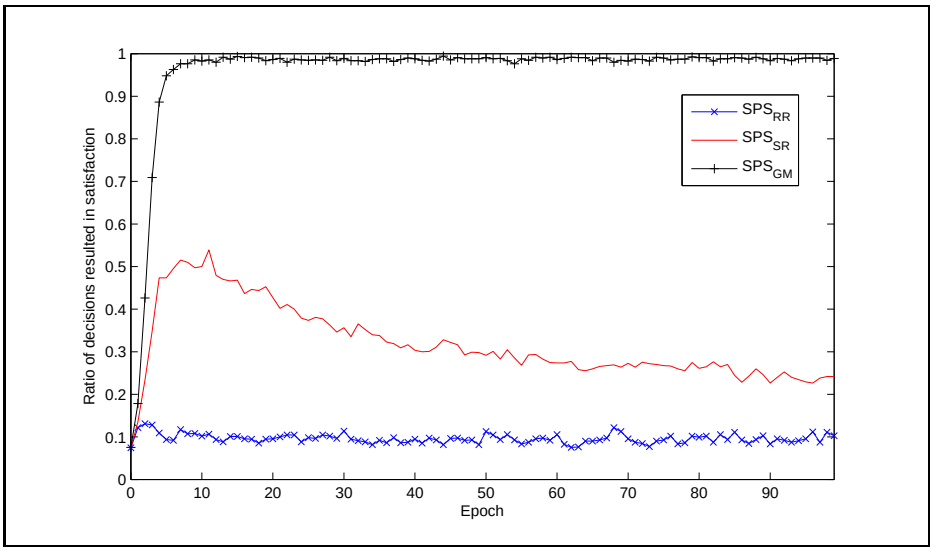

Figure 3.13. Consumers change their service demands (P_{CD}=0.2) and their satisfaction criteria are conflicting (R_{subj}=0.5)

Table 3.2. Average ratio of satisfaction for different R_{subj} values (P_{CD} is set to 0)

R_{subj}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
SPS_{GM}	0.96	0.97	0.97	0.96	0.97	0.96	0.97	0.96	0.96	0.97	0.96
SPS_{SR}	0.96	0.85	0.76	0.68	0.61	0.56	0.38	0.28	0.19	0.09	0.003

3.4.2. Experience-Based Approach: CBR

Both SPS_{GM} and SPS_{CBR} use the experiences as the representation of consumers' past dealings with the providers. However, they use different decision making processes. In order to see the effect of these different decision making processes in service selection, we compare their performances through simulations.

This section summarizes results of the simulations for SPS_{CBR} and compares its performance with that of SPS_{GM}. Initially, there are two primary variables in the simulations: P_{CD} and R_{subj}. We measure the average satisfaction ratio of the strategies when R_{subj} equals

Table 3.3. Average ratio of satisfaction for different R_{subj} and P_{CD} values

P_{CD}	SPS_{GM}		SPS_{SR}	
	$R_{subj} = 0$	$R_{subj} = 0.5$	$R_{subj} = 0$	$R_{subj} = 0.5$
0.0	0.96	0.96	0.96	0.56
0.1	0.97	0.96	0.79	0.36
0.2	0.97	0.96	0.63	0.32
0.4	0.97	0.96	0.50	0.26
0.6	0.97	0.96	0.44	0.18
0.8	0.97	0.98	0.42	0.18
1.0	0.97	0.97	0.40	0.19

0 and 0.5 as well as when P_{CD} varies from 0 to 1. In Table 3.4, we immediately note that for all values of P_{CD}, both SPS_{GM} and SPS_{CBR} achieve equally high average satisfaction ratio, while SPS_{SR} achieves a fluctuating value as seen Table 3.3, because of its sensitivity to context change (P_{CD}) and subjectivity (R_{subj}). Unlike the rating-based approaches such as SPS_{SR}, SPS_{GM} and SPS_{CBR} are robust to R_{subj} and P_{CD}. Moreover, the performance of these two strategies are impressive and equal.

Table 3.4. Average ratio of satisfaction for different R_{subj} and P_{CD} values

P_{CD}	SPS_{GM}		SPS_{CBR}	
	$R_{subj} = 0$	$R_{subj} = 0.5$	$R_{subj} = 0$	$R_{subj} = 0.5$
0.0	0.96	0.96	0.96	0.96
0.1	0.97	0.96	0.97	0.96
0.2	0.97	0.96	0.97	0.96
0.4	0.97	0.96	0.97	0.96
0.6	0.97	0.96	0.97	0.96
0.8	0.97	0.98	0.97	0.98
1.0	0.97	0.97	0.97	0.97

Although the performance of SPS_{GM} and SPS_{CBR} are the same, the time they use to select the providers are different. Table 3.5 presents the average time consumed by each approach during our simulations. The table shows that SPS_{CBR} is much more efficient than SPS_{GM} in terms of time consumption. Moreover, we can conclude that SPS_{CBR} is better

than SPS_{GM} in these settings, because it can find the satisfactory providers much faster.

Table 3.5. Time consumption of SPS_{GM} and SPS_{CBR} in milliseconds

P_{CD}	SPS_{GM}		SPS_{CBR}	
	$R_{subj} = 0$	$R_{subj} = 0.5$	$R_{subj} = 0$	$R_{subj} = 0.5$
0.0	1250	1081	448	270
0.1	3016	2879	680	651
0.2	5376	4292	982	674
0.4	7015	7976	1078	1287
0.6	10154	8163	1586	680
0.8	14983	21610	2555	3360
1.0	20377	17711	2583	1272

3.4.3. Additional Simulation Factors

So far, we have deliberately assumed that providers always provide the same quality of service consistently. However, some providers may have nondeterministic nature and may supply marginally different services at different instances of time for the same service demand and conditions. Now, we introduce a new variable PI that denotes the probability of nondeterminism on the provider side. This probability denotes how much providers deviate from their expected behavior. Consider a provider that usually produces unsatisfactory services for a specific service demand. If this provider produces a perfect service for this service demand in a transaction with a consumer, this kind of nondeterminism may mislead the consumers in their future decisions.

Since we have not modeled this situation in the previous experiments, we have implicitly assumed that PI equals 0, which means that there is no nondeterminism in the behavior of providers. In other words, for a particular service demand, a service provider will either be satisfactory or dissatisfactory independent of when the service is demanded. In the settings where behaviors of providers are predictable and free of nondeterminism, SPS_{CBR} can easily replace SPS_{GM}. Moreover, in terms of computational efficiency, SPS_{CBR} outperforms SPS_{GM} in these settings.

In the origin of the CBR approach, there is an assumption that if a provider satisfies a service demand that is similar to or the same as the current demand of a consumer, the provider will probably satisfy the consumer's current demand, too. When PI is set to zero, this assumption always holds. Providers produce similar services for the same or very similar service demands. These services deviate insignificantly from each other so that the deviation does not affect the consumers' degree of satisfaction. However, in realistic environments, some providers may infrequently provide marginally different services for the same or similar service demands. The experiences containing these service instances may be misleading for the consumers. In order to simulate such situations, PI is set to very small probability values. In the following simulations, with these probabilities, each provider deviates from its usual service offering in favor of consumers by producing perfect services for the consumers.

Figure 3.14 and Figure 3.15 show simulation results for PI = 0.001 and PI = 0.01, respectively. The main result is that when the achieved satisfaction is considered, SPS_{CBR} is sensitive to the PI parameter such that the achieved satisfaction decreases with an increase in the value of PI. However, the performance of SPS_{GM} does not change with variations in PI and is constant around 100% satisfaction.

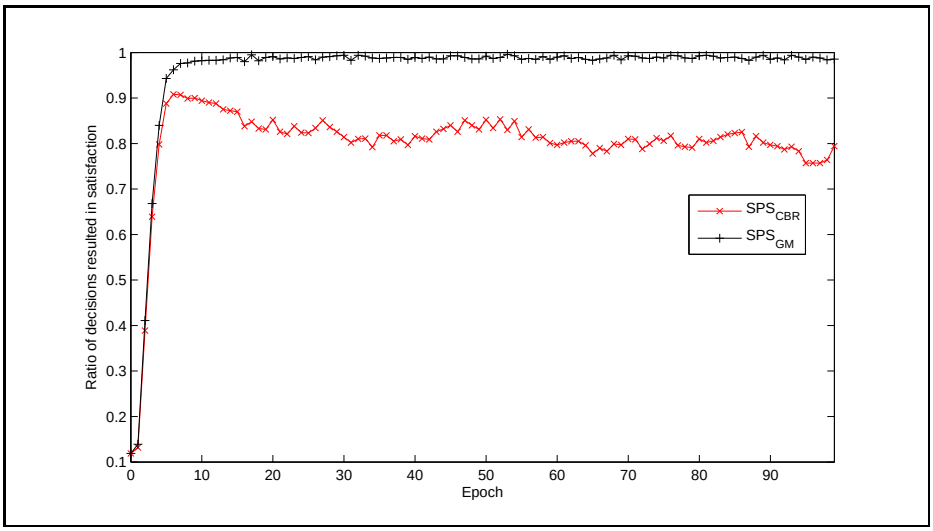

Figure 3.14. Performance of SPS_{CBR} decreases when PI = 0.001

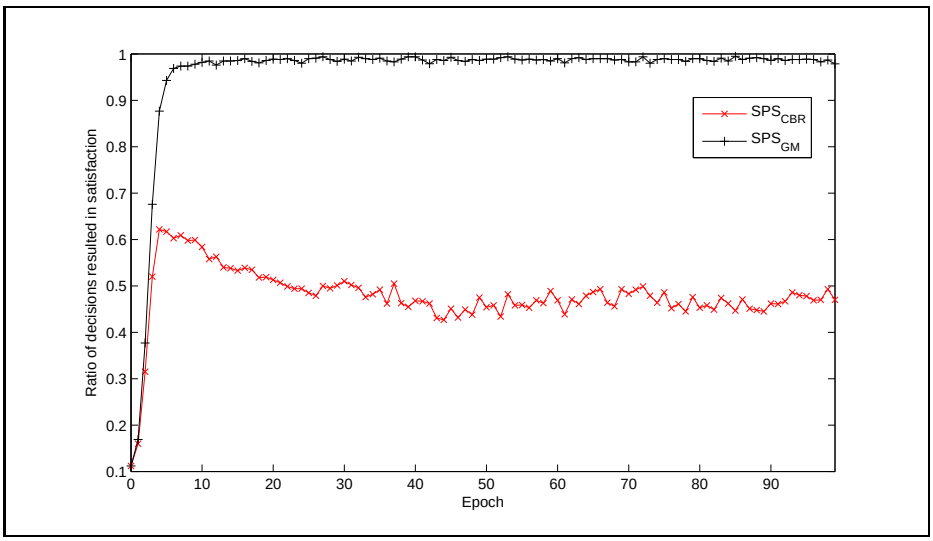

Figure 3.15. Performance of SPS_{CBR} decreases even further when PI = 0.01

3.4.4. Using Different Classification Methods for Service Selection

In Section 3.3.4.1, we propose to use parametric classification with multivariate Gaussian model for decision making. The resulting experience-based service selection approach is called as SPS_{GM}. The main idea in SPS_{GM} is the computation of the probability that a provider produces a satisfactory service for a given demand. Once a training set is derived as explained in Section 3.3.4.1, other classification methods can also be used to compute this probability. Hence, these methods can also be used for experience-based service selection. In this section, we measure the performances of *Naive Bayes* and *C4.5 Decision Tree* classifiers when they are used for experience-based service selection. Table 3.6 shows our results for the performances of these approaches for different values of P_{CD}, R_{subj}, and PI. In the table, SPS_{NB} and SPS_{DT} refer to the experience-based service selection approaches that use *Naive Bayes* and *C4.5 Decision Tree* classifiers, respectively. The table shows that using different classifiers for experience-based service selection does not change the service selection performance considerably. In other words, service selection performances of SPS_{GM}, SPS_{NB}, and SPS_{DT} are almost the same and do not significantly change with the different values of P_{CD}, R_{subj}, and PI. Hence, we can conclude that different classification methods can be used with experiences to make context-aware service selections that are robust to subjectivity and nondeterminism.

Table 3.6. Performances of different classification methods for service selection

P_{CD}	R_{subj}	PI	SPS_{GM}	SPS_{NB}	SPS_{DT}
0	**0**	**0**	0.96	0.94	0.92
0.2	**0**	**0**	0.97	0.95	0.93
0	**0.5**	**0**	0.95	0.95	0.95
0.2	**0.5**	**0**	0.96	0.96	0.95
0.2	**0.5**	**0.001**	0.97	0.95	0.94
0.2	**0.5**	**0.01**	0.96	0.95	0.95

3.5. Discussion

Selecting an appropriate service provider is a must in open settings. However, identifying the correct service providers is difficult. Previous research on service provider selection is mainly based on recording and aggregating ratings of consumers. Most of the previous rating-based approaches are similar to SPS_{RR} in the sense that they do not differentiate between the raters based on the raters' previous service interests. Alternatively, this chapter presents our research on ontology-based service selection and proposes a framework for the service selection problem. We show that better service providers can be chosen when consumers represent their experiences with ontologies.

3.5.1. Usage

The services that are mentioned here can easily be thought of as Web services. In this research, we have not been concerned about how Web services describe the access to their services. We implicitly assume that consumers can locate the service producers and access their services. However, in real life, the consumers would need to benefit from a mechanism that identifies how the service providers can be accessed. To this end, standard description languages such as WSDL can be used [64].

In the protocol that we have used in Section 3.3.3, the agents use others' experiences to

find service providers. However, two interesting cases can take place. First, the agents may know each other but not know the service providers. This will mean that initially nobody will have enough experiences to guide others. To cope with this problem, directories such as UDDI registries can be used [65]. Agents may initially look up the service providers from these registries and have interactions with these service providers. When enough experiences have been accumulated, then the system will continue to function as explained above. Second, the agents may not even be aware of each other. This is the well-known problem of bootstrapping. This means that the agents will need a mechanism to find acquaintances. In order to get some initial acquaintances, service consumers may sign in to another directory called consumer directory. They may provide their identity and IP address to the directory and the consumer directory may provide the consumer with a small number of initial acquaintances. This is a common method for P2P systems. After locating some acquaintances, the system will continue functioning as explained in Section 3.3.3.

The proposed context-aware ratings and context-aware experiences both assume that the agents share a common ontology. In open systems, one way to enable this is to allow the agents to download the ontology from a well-defined resource. The base ontology will be the same for all the domains. However, the domain ontology will differ based on the domain. For different domains, we expect the domain experts to come up with ontologies that capture the specifics of the domains.

After receiving a service, an agent needs to create feedback information about the service. This information corresponds to the Service class in the base-level ontology. Some of the necessary information about the service may be generated by the agent, whereas some information may need to be generated by the user of the agent. For example, the agent may know when the requested service has actually been received, but may not have enough knowledge to evaluate the quality of the service. For such cases, the agent's user must be eager to provide information to the agent through some predefined software interface.

3.5.2. Summary of Results

We have introduced and studied two main approaches for service selection. The first proposed approach, SPS_{CAR}, is based on ratings and it enriches the rating data with the context information using ontologies. The second proposed approach is based on capturing the experiences of consumers through ontologies. Through simulations, we have shown that the proposed approaches improve the decisions of the service consumers and increase the overall satisfaction significantly compared with the previous rating-based service selection approaches.

SPS_{SR} is a variant of distributed collaborative filtering approach for rating-based service selection. It uses plain ratings from those consumers that have had a similar service demand in the past. Ontologies are used to determine these consumers. Then, these consumers are contacted for their ratings. SPS_{SR} outperforms SPS_{RR}. The performance of SPS_{SR} decreases over time as the consumers change their demand characteristics and drops to the level of SPS_{RR}.

Unlike SPS_{SR}, SPS_{CAR} is robust to changes in the demand characteristics of the consumers. The performance of SPS_{CAR} is always better than the performance of SPS_{RR} and SPS_{SR}. However, it is vulnerable to changes in the parameter R_{subj}, because ratings reflect the satisfaction criteria and taste of the raters. Even though the consumers have the same service demands, their ratings may be highly different for the same service provided for this service demand. Our finding is that any rating-based approach will experience the same vulnerability.

The best approach in terms of achieved ratio of satisfaction is the experience-based approach. When service providers do not change the quality of their services, both SPS_{GM} and SPS_{CBR} perform equally well in finding service providers (they achieve about 100% satisfaction). However, SPS_{CBR} finds the service providers in a shorter time than SPS_{GM}. On the other hand, if the service providers vary their service offerings even a small percentage, then SPS_{CBR} performs much worse than SPS_{GM}.

Although experience-based approach is the best service selection approach in this chapter, it requires service consumers to record their experiences in substantial detail. This may be exhaustive or such information may not be available. Under such circumstances, the proposed rating-based approach, SPS_{CAR}, can be preferred by service consumers over experience based approach even though SPS_{CAR} has some disadvantages with respect to experience-based approach in terms of achieved ratio of satisfaction.

3.5.3. Related Work

Current service provider selection strategies accept ratings as first-class citizens, but do not allow more expressive representations like we have here. Whereas rating-based approaches assume that the ratings are given and taken in similar contexts (e.g., in response to similar service demand), we can make the context explicit. This allows agents to evaluate others' experiences based on their needs. Thus, the use of context information and experiences improves the satisfaction rate of the consumers.

FIRE [18] is a trust and reputation model consisting of four components: interaction trust, role-based trust, witness reputation and certified reputation. Witness reputation component is directly related to our approach since it allows agents to locate others by making use of other agents' past experiences. However, in FIRE the past experiences are captured only as ratings. However, in our approach, agents exchange their experiences in the form of ontologies so that they can represent their demands, received services, and so on in more depth.

Sen and Sajja [66] develop a reputation-based trust model that is used for selecting processor agents for processor tasks. Each processor agent can vary its performance over time. Agents are looking for processor agents to send their tasks to using only evidence from others. Sen and Sajja propose a probabilistic algorithm to guarantee finding a trustworthy processor. In our framework, service demands among agents are not equivalent; hence a provider that is trustworthy for a consumer need not be so for a different consumer. Hence, each agent can select a different provider for its needs.

Yolum and Singh study properties of referral networks for service selection, where referrals are used among the service consumers to locate the service providers [14]. Current applications of referral networks also rely on exchanging ratings. Hence, the suffer from circulation of subjective information. However, it would be interesting to combine referral networks with the ontology representation here so that agents can exploit the power of ontologies for knowledge representation as well as referrals for accurate routing.

Zhang *et. al.* propose a multiagent approach for distributed information retrieval task [42]. In their work, each agent has a view of its environment called agent-view. The agent-view structure of an agent contains information about language models of documents owned by each agent. An agent-view reorganization algorithm is run to dynamically reorganize the underlying agent-view topology. An agent-view is analogous to acquaintance list structure in our work. Zhang *et al.*'s protocol does not use ontologies or description logic reasoners during information retrieval. However, if their protocol is modified to accommodate the experience ontology and DL reasoners, their protocol can be used for retrieving experiences instead of the protocol that we have used in Section 3.3.3.

Soh and Chen propose a multiagent approach to improve distributed information retrieval performance by balancing ontological and operational factors [67]. In this work, collaborating agents enhance their performance by learning ontologically and operationally. Soh and Chen show that their proposed approach is able to improve the quality of the collaborations in terms of the response time, quality of the retrieved results, the number of neighbors contacted and message complexity. A similar approach can also be applied to our work in order to improve the quality of collaboration.

Maximilien and Singh develop a QoS ontology to represent the quality levels of service agents and the preferences of the consumers [15]. Their representation of QoS attributes is richer (such as availability, capacity, and so on), however, their ontology does not represent commitments and thus business contracts as part of the ontology. Further, their system does not allow reasoning by agents individually as we have developed here.

CBR is used in centralized recommendation systems to automatically estimate con-

sumer preferences. Aguzzoli *et. al.* propose a collaborative cased based recommendation system for the music market [68]. The proposed system is hosted by an online shopping site. During their online shopping, consumers choose sound tracks and add them to their shopping chart, which is called a partial compilation. The system inspects the partial compilation of a consumer and recommends new sound tracks using a case-base. This case-base is composed of the recorded compilations of consumers, who have previously visited the web site and used this system. Matching of sound tracks between the partial compilation and the compilations in the case-base is used for the computation of similarity between compilations. Then, the sound tracks that are included in the similar compilations are recommended to the consumer. A similar approach for recommending restaurants are proposed by Burke [69]. This system is hosted by a website, which is similar to a catalog for the restaurants. The system records the browsed restaurants as cases and recommends new restaurants to the users depending on their browsing histories.

Limthanmaphon and Zhang propose a web service composition approach [70]. They use CBR for service discovery. The definition of preassembly composite service cases are stored in a case-base. The definition of a composite service includes the set of services it includes and relationships between these services. When a user has a new request for a composite service, similarity measure is used to find the closest cases in the case-base. As similarity measure, matching between the definitions of composite services are used. Then, the preassembly service with the highest similarity value is suggested to the user. However, none of these case-based approaches use an ontology in conjunction with a case-base as we have done here.

So far, we have assumed that agents exchange their experiences or context-aware ratings honestly. However, some consumers may want to defame some providers because of personal or commercial reasons by disseminating deceptive information about the providers. This situation imposes the requirement of using a trust mechanism so that some consumers could be ranked as trustful and others could be ranked as distrustful (cheater or slanderer). As a result, in Chapter 4, we develop an approach for handling deceptive information and integrate it into the experience-based service selection.

4. CONTEXT-AWARE SERVICE SELECTION UNDER DECEPTION

In Chapter 3, we have showed that when consumers speak the truth, the proposed experience-based approach outperform the rating-based approaches in terms of providing consumer satisfaction, especially in populations where the satisfaction criteria of the users vary [1,3]. However, in many settings, consumers may prefer to be dishonest about their experiences. As a result, if a consumer cannot differentiate between honest and deceptive experiences of others, deceptive experiences may mislead the consumer [6]. Hence, one needs to identify deceptive experiences and disregard them when selecting a service provider. In this chapter, we propose a method to allow consumers to model the trustworthiness of other consumers. Unlike the many deceptive information filtering in the literature [50,52], this method combines consumers' personal observations about the providers and public knowledge of the others.

In summary, in this chapter, we propose an integrated approach for context-aware service selection in deceptive environments. This approach effectively combines: (1) the experience-based service selection approach in Chapter 3 that makes context-aware service selections using the shared consumer experiences, and (2) an information filtering method that computes trustworthiness of the consumers and identifies deceptive experiences. This method filters out deceptive experiences, before they are used for the selection of service providers. We evaluate the performance of the proposed approach using extensive experiments in different settings. We compare it with two recent rating-based service selection approaches from the literature. Our experiments show that using the proposed approach, service consumers can successfully select satisfactory service providers even if a significant ratio of consumers are liars, and even if the satisfaction criteria and the context of consumers vary considerably over time. Moreover, our integrated approach significantly outperforms the rating-based approaches in those settings.

The rest of this chapter is organized as follows. Section 4.1 describes the effects of deceptive information in the context-aware service selection proposed in Chapter 3. Section 4.2 presents our method for the computation of consumers' trustworthiness and filtering out deceptive experiences received from untrustworthy consumers. Section 4.3 experimentally evaluates our integrated approach for context-aware service selection with comparisons to well-known rating-based service selection approaches. We offer concluding remarks in Section 4.4.

4.1. Effects of Deceptive Experiences in Context-Aware Service Selection

In experience-based approaches, consumers use the experiences of others. Experiences are more expressive than ratings. A negative rating may be given by a consumer because of various reasons such as late delivery, insufficient consumer support or the worst case; *defraudation* where the consumer pays for a service, but never receives the paid service. Ratings do not distinguish between these; a negative rating means only the dissatisfaction of the rater. However, experiences can express the actual story; what is requested and what is actually received. It is apparent that defraudation is much more serious reason for the consumer dissatisfaction. When a consumer encounters an experience including defraudation made by a provider , the consumer will probably never consider the provider for a transaction. However, experiences are produced by consumers and some consumers may want to defame some providers because of personal or commercial reasons. This situation imposes the requirement of using a trust mechanism so that some consumers could be ranked as trustful and others could be ranked as distrustful (liar). As a result, our framework should have an integrated collaborative trust mechanism.

In this section, we describe how the expressiveness of experiences can be abused by some malicious consumers to mislead others during service selection. For this purpose, we first describe the behavior of malicious consumers, and then we empirically analyze the possible effects of deceptive experiences in the context-aware service selection.

4.1.1. Behavior of Liars

In the rating-based service selection literature, behavior of a liar is usually defined using a simple formula; $r' = r - 1$, where $0 \leq r \leq 1$ is the true rating of the liar and $0 \leq r' \leq 1$ is the deceptive rating that the liar shares with others. This way, the liar is assumed to mislead others as much as possible (it gives bad ratings for good providers and good ratings for bad providers). In this chapter, we follow the same way, but we redefine this behavior for the case of experience-based service selection, where consumers share their ontology-based past experiences instead of ratings.

In experience-based service selection, liars modify their experiences before sharing, so that they mislead the other consumers the most. This is achieved by disseminating awful experiences about the good providers and perfect experiences about the bad providers. That is, if the liar demanded a notebook within 7 days from a provider in the past and it is delivered on time, the liar states in its experience that the merchandise was not received even though the money is paid out or the notebook was delivered after 120 days. This way, extremely negative experiences about the good service providers are disseminated by the liars. On the other hand, if an experience contains an unsatisfactory service, the liar modifies the experience before sharing so that the received service looks like it has satisfied the demand of the customer. For example, if the liar demanded a notebook within 7 days from a provider in the past, but delivery was made after 30 days, the liar states in its experience that the notebook was delivered within 7 days.

4.1.2. Influence of Extremely Negative Experiences

Since there are many highly rated sellers in the marketplaces such as e-Bay, buyers have become very sensitive to any negative feedback. Cabral and Hortacsu examine e-Bay to figure out the effects of negative and positive reviews on the behaviors of buyers and sellers [71]. They show that when a seller in e-Bay receives a negative review for the first time, his sales' growth decreases by 14%. Similarly, using data on e-Bay, Lucking-Reiley demonstrates that negative feedback (e.g., negative user reviews) has a statistically significant effect on users' biddings during auctions, but positive feedback does not [72]. In

other words, once a seller has negative feedback, the number of bids put for the seller drops significantly, while a similar effect is not observed for positive feedback.

These researches reveal that consumers are dramatically affected by the negative reviews about the providers in real life, even though there are a significant number of positive reviews for those providers. The liars defined in this chapter disseminate extremely negative experiences about the good service providers. Those negative experiences are different from ordinary negative experiences, because they contain extremely repulsive information about the service providers, such as defraudation. In real life, such information may have a great impact on the consumers' service decisions. In order to imitate this, we implement service consumers so that they prefer not to select a provider if they receive extremely negative experiences about the provider. That is, the number of positive experiences about a provider does not matter, if a consumer receives several extremely negative experiences about the provider and these experience are not filtered out, the consumer does not choose the provider. This type of sensitivity to deceptive information makes successful service selection much harder for experience-based service selection, because even a few unfiltered deceptive experiences may seriously affect the performance of service selection. However, rating-based service selection approaches do not have the same sensitivity to deceptive information, because they do not have any information about the rational of a negative rating.

4.1.3. Empirical Analysis of Deception

In this section, we present the performance of the experience-based service selection approach in Chapter 3 when there are liars in the environment. For our evaluations in this section, we select SPS_{GM}, which is shown to be the best experience-based service selection method in Chapter 3. In our experiments, parameter R_{liar} defines the ratio of liars in the consumer society. We use the same environmental setup and parameters in Chapter 3. For each value of R_{liar}, simulations are run for 100 epochs and average ratio of successful service selections is calculated for each R_{liar} value. Results of our simulations are summarized on Figure 4.1. In order to see the effect of deceptive information better, we set the parameters other than R_{liars} to zero ($PI = 0$, $P_{CD} = 0$ and $R_{subj} = 0$) throughout the simulations.

Figure 4.1 shows that, when there are no liars in the environment, experience-based service selection leads to successful service selections almost all the time (success at around 97% of the cases on the average). However, when 10% of the consumers are liars, performance of service selection sharply decreases below 0.35. This is the effect of extremely negative experiences disseminated by the liars about the satisfactory providers. For higher ratio of liars, the performance of service selection approach reaches to 0.0, because the consumers mostly select providers among the bad ones. This is intuitive, because in this setting there are many extremely negative experiences about the good providers and considerable amount of positive experiences about the bad providers. Hence, the sincere experiences disseminated by the honest consumers are overwhelmed by the experiences disseminated by the liars in the environment. As a result, mostly the bad providers are considered for service selection while the ratio of liars among the consumers increases.

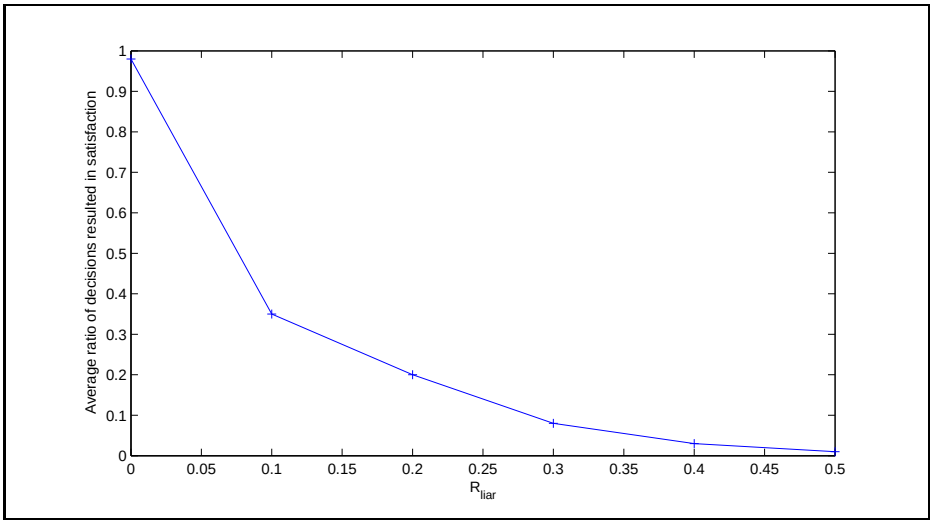

Figure 4.1. Drop in the performance of experience-based service selection (SPS_{GM}) when $0 \leq R_{liars} \leq 0.5$, where $PI = 0$, $P_{CD} = 0$ and $R_{subj} = 0$

4.2. Filtering out Deceptive Experiences

In this section, we describe how deceptive experiences are determined and filtered out during service selection. From now on, we call consumers that share their experiences with others "advisors". This terminology is used to comply with the *trust and reputation* literature, where agents that share their ratings with others are usually called "advisors". A

consumer can estimate the trustworthiness of an advisor by combining two different sources of information; private and public credits of the advisor. The private credit of the advisor is calculated by the consumer, based on the experiences the advisor supplies of providers with whom the consumer has already had some personal experiences. If private credit cannot be calculated with confidence, a public credit is calculated, based on the advisor's experiences with all providers in the environment. A weighted combination of the private and the public credits is derived, based on the estimated reliability of the private credit value. This combined value then represents the trustworthiness of the advisor. After that, the experiences received from the less trustworthy consumers are finally regarded as deceptive and filtered out during service selection. Note that during those calculations, we only consider the experiences related to the current demand of the consumer, because only those experiences are used for the service selection, so the context of those experiences is the same as the current one. In other words, trustworthiness of advisors is calculated in a context-dependent way. This enables an advisor to be regarded as trustworthy in one context while the advisor may be regarded as untrustworthy in another context.

4.2.1. Private Credit of Advisors

Our approach allows a consumer C to evaluate the private credit of an advisor A by comparing their experiences for their commonly encountered providers $\{P_0, P_1, ..., P_m\}$. For each of the commonly encountered provider P_i, A has the experience vector E_{A,P_i} and C has the experience vector E_{C,P_i}. The experiences in E_{A,P_i} and E_{C,P_i} are ordered according to their recency. The experiences are then partitioned into different elemental time windows. The length of an elemental time window may be fixed (e.g. three days) or adapted by the frequency of the transactions with the provider P_i, similar to the way proposed in [73], where the length is smaller when the frequency of the transactions is high, and larger when the frequency is low. It should also be considerably small so that there is no need to worry about the changes of providers' behavior within each elemental time window.

We define a pair of experiences (e_{A,P_i}, e_{C,P_i}), such that e_{A,P_i} is one of the experiences in E_{A,P_i}, e_{C,P_i} is one of the experiences in E_{C,P_i}, and e_{A,P_i} *corresponds* to e_{C,P_i}. Two experiences, e_{A,P_i} and e_{C,P_i}, are correspondent only if the experience e_{C,P_i} is the most recent

experience in its time window, and the experience e_{A,P_i} is the closest and prior to the experience e_{C,P_i}. We consider experiences provided by C after those by A, in order to incorporate into C's experiences anything learned from A, before taking an action. According to the solution proposed in [74], by keeping only the most recent experiences, we can avoid the issue of advisors' "flooding" the system. No matter how many experiences are provided by one advisor in a time window, we only keep the most recent one. Then, we count the number of experience pairs for P_i, denoted as N_{P_i}. The total number of experience pairs for all commonly encountered providers (N_{all}) will be calculated by summing up the number of experience pairs for each commonly encountered provider as follows:

$$N_{all} = \sum_{i=0}^{m} N_{P_i} \tag{4.1}$$

For each pair of experience (e_{A,P_i}, e_{C,P_i}), the consumer C converts e_{A,P_i} and e_{C,P_i} to its satisfaction levels based on its own taste function F^C_{taste} as follows:

$$l_{A,P_i} = F^C_{taste}(e_{A,P_i}), \quad l_{C,P_i} = F^C_{taste}(e_{C,P_i}) \tag{4.2}$$

We define the experience pair (e_{A,P_i}, e_{C,P_i}) as a positive experience pair if l_{A,P_i} is the same as l_{C,P_i}. Otherwise, the pair is called as a negative experience pair.

We examine experience pairs for all commonly encountered providers. Suppose there are N_p number of positive pairs. The number of negative pairs will be $N_{all} - N_p$. The private credit of the advisor A is estimated as the probability that A will provide truthful experiences to C. By truthful experiences, we mean the experiences whose converted satisfaction levels are the same as the ones of the personal experiences of C. Because there is only incomplete information about the advisor, the best way of estimating this probability is to use the expected value of the probability. The expected value of a continuous random variable is dependent on a probability density function, which is used to model the probability that a variable will have a certain value. Because of its flexibility and the fact that it is the conjugate prior for distributions of binary events [75], the beta family of probability density functions is commonly used to represent probability distributions of binary events (see, e.g. the gener-

alized trust models BRS [76] and TRAVOS [52]). Therefore, the private credit of A can be calculated as in Eqauation 4.5.

$$\alpha = N_p + 1 \tag{4.3}$$

$$\beta = N_{all} - N_p + 1 \tag{4.4}$$

$$R_{pri}(A) = E[Pr(A)] = \frac{\alpha}{\alpha + \beta} \tag{4.5}$$

In Equation 4.5, $Pr(A)$ is the beta probability density function for the probability that A will provide truthful experiences to C, and $E[Pr(A)]$ is the expected value of this probability density function.

4.2.2. Public Credit of Advisors

If the consumer C has a few or no personal experiences about the providers that the advisor A has experience with, then private credit of A cannot be computed by C with confidence. In this case, the consumer C calculates A's public credit in addition to its private credit. For this purpose, experiences given by A is examined to determine if they are consistent with the majority of the experiences given by the other advisors for the same providers. Consistency of an experience e_{A,P_i} with the majority is computed as follows. First, the consumer C determines the experiences provided by other advisors about the same provider, P_i. Suppose that n other advisors ($A_0 \dots A_{n-1}$) also have given C their experiences about the provider P_i. Let one of the experiences given by the advisor A_j be e_{A_j,P_i}, where $0 \leq j < n$ and e_{A_j,P_i} corresponds to e_{A,P_i}. In other words, similar to the calculation of private credit, e_{A,P_i} and e_{A_j,P_i} are within the same time window, e_{A_j,P_i} is prior to e_{A,P_i}, and they are the most recent experiences in the corresponding time window. Hence, we guarantee that the conflicts between the experiences in our calculations are not due to the behavior change of the providers, but instead due to dishonest reporting. Second, those experiences provided by other advisors about P_i are converted to the consumer C's satisfaction levels, using Equation 4.2. In this case, we use 1 to represent satisfactory experiences and 0 to represent dissatisfactory experiences. Then, we calculate the average satisfaction level (avg) as in

Equation 4.6. The experience e_{A,P_i} of the advisor A is considered a consistent experience if $|F^C_{taste}(e_{A,P_i}) - avg| \leq \phi$; otherwise, e_{A,P_i} is considered as an inconsistent experience. In our calculations, $0 < \phi < 0.5$ is the maximum acceptable deviation from the majority.

$$avg = \frac{\sum_{j=0}^{n-1} F^C_{taste}(e_{A_j,P_i})}{n} \tag{4.6}$$

Suppose that the advisor A provides in total N'_{all} experiences for the current demand of C. If there are N_c consistent experiences among those experiences, the inconsistent experiences provided by A will be $N'_{all} - N_c$. In a similar way as estimating the private credit, the public credit of the advisor A is estimated as the probability that A will provide consistent experiences for the current demand of C. It can be calculated as in Eqauation 4.9.

$$\alpha' = N_c + 1 \tag{4.7}$$

$$\beta' = N'_{all} - N_c + 1 \tag{4.8}$$

$$R_{pub}(A) = \frac{\alpha'}{\alpha' + \beta'} \tag{4.9}$$

Eqauation 4.9 indicates that public credit of an advisor is high as long as it gives experiences consistent with the experiences of the majority.

4.2.3. Trustworthiness of Advisors

In order to estimate the trustworthiness of the advisor A, we combine the private credit and the public credit values. The private credit and the public credit values are assigned different weights. The weights are determined by the reliability of the estimated private credit value. For this purpose, we first determine the minimum number of experience pairs needed for C to be confident about the calculated private credit of A. The Chernoff Bound theorem [61] provides a bound for the probability that the estimation error of private credit exceeds a threshold, given the number of pairs. Accordingly, the minimum number of pairs

can be determined as in Equation 4.10.

$$N_{min} = -\frac{1}{2\varepsilon^2} ln \frac{1-\gamma}{2} \tag{4.10}$$

In Equation 4.10, ε is the maximum error that can be accepted by C, and γ is the confidence measure. If the total number of experience pairs used for the calculation of the private credit is larger than or equal to N_{min}, the consumer C is confident about the calculated private credit value. Hence, this value is used as the trustworthiness of A. However, if the used experience pairs are less than N_{min}, the consumer C combines the private and the public credit values as a weighted sum. The weight (or reliability) of the private credit value can be measured as follows:

$$w = \begin{cases} \frac{N_{all}}{N_{min}} & \text{if } N_{all} < N_{min}; \\ 1 & \text{otherwise.} \end{cases} \tag{4.11}$$

The trustworthiness of A is calculated by combining the private and public credit values as follows:

$$Tr(A) = wR_{pri}(A) + (1-w)R_{pub}(A) \tag{4.12}$$

4.3. Evaluation

We extend our simulator in Chapter 3 to measure the performance of our approach in selecting an appropriate service provider in deceptive environments. In the implementation of the proposed approach, we set the maximum acceptable deviation from the majority $\phi = 0.1$, the acceptable level of error $\varepsilon = 0.4$ and the confidence measurement $\gamma = 0.6$ during the calculations of advisors' trust values. This setting is purposefully chosen to give more weight to the personal observations (private credit) rather than the information from others (public credit). After calculating the trust values using private and public credits, we regard an advisor as a liar if its trustworthiness is less than 0.5. We integrate the proposed deceptive information filtering approach into SPS_{GM} to measure the achieved performance

improvement in deceptive environments. We call the integrated experience-based service selection approach as SPS^*_{GM}. We also implement two different service selection approaches from the literature and compare them with SPS^*_{GM}. Those approaches are explained briefly in the next section (for details see Section 2.3).

4.3.1. Service Selection Approaches for Benchmarks

There are many rating-based service selection approaches in the literature. We use three of those approaches to make benchmark comparisons with our approach. Those approaches are shortly explained below. In order to make more reliable comparisons, the rating-based approaches and SPS^*_{GM} use the same information sources in our experiments. While SPS^*_{GM} uses *experiences*, the rating-based approaches use *ratings* from the same sources (advisors).

The beta reputation system (BRS) is proposed by Jøsang and Ismail [76]. It estimates reputations of service providers using a probabilistic model. This model is based on the beta probability density function, which can be used to represent probability distributions of binary events. In this approach, consumers propagate their ratings about providers. A rating of the consumer c to the provider p is in the form of $r = [g, b]$, where g is the number of c's good interactions with p and b is the number of c's bad interactions with p. Ratings from different consumers about the same provider are combined by simply computing the total number of good interactions and the total number of bad interactions with the provider. These two numbers are used to compute the parameters of a beta distribution function that represents the reputation of the provider. To handle unfair ratings provided by other consumers (advisors), Whitby *et al.* extend the BRS to filter out those ratings that do not comply with the significant majority of the ratings by using an *iterated filtering approach* [50]. Hence, this approach assumes that significant majority of the advisors honestly share their ratings (see Section 2.3.1 for the detailed description).

Trust and reputation in the context of inaccurate information sources (TRAVOS) is proposed by Teacy *et al.* [52]. Similar to BRS, it uses beta probability density functions to compute consumers' trust on service providers. The main difference between BRS and

TRAVOS is the way they filter out unfair ratings. While BRS uses the majority of ratings to filter out unfair ratings about the providers, TRAVOS uses the personal observations about those providers to detect and filter out unfair ratings. Hence, unlike BRS, TRAVOS does not assume that the majority of ratings are fair (see Section 2.3.3 for the detailed description).

4.3.2. Simulation Environment

We use the same simulation environment in Section 4.1.3. Only one of the service providers can satisfy a given service demand. When the simulations start, agents do not have any prior experiences with service providers. At each epoch, with a probability of 0.5, a consumer requests a service for its current service demand. Then, it collects experiences related to similar service demands from other consumers in order to use for service selection. In our simulations, we force consumers to make service decisions based on the information from others rather than their own previous experiences. In this way, we can test the abilities of our approach better against subjectivity, unreliability and context-awareness.

In this section, we use rating-based service selection approaches for benchmark comparisons. In this case, liars give bad ratings to good service providers and good ratings to bad service providers, analogous to the experience-based case described in Section 4.1. Formally, if the true rating of a liar to a provider is r, the liar modifies the rating as $r' = 1 - r$ before sharing with the other consumers to mislead the others as much as possible [20].

Our main performance metric is success in service selection. We measure it as the ratio of service decisions resulted in satisfaction as in Chapter 3. Intuitively, in deceptive environments, the success in service selection should be correlated with the amount of filtered deceptive information during service selection. As the amount of unfiltered deceptive information increases, the performance of service selection approaches is expected to decrease. Our supplementary performance metric is the error in identifying liars among the advisors during service selection.

4.3.3. Experimental Results

In this section, we evaluate the performance of our integrated service selection approach in three steps. First, we examine our approach in deceptive environments when there is no subjectivity and variation on context. Second, we investigate the performance change when subjectivity is included in the experiments. Lastly, we consider reliable environments with no subjectivity and inspect the performance of our approach when consumers are allowed to change the context of their service demands.

4.3.3.1. Deceptive Environments without Subjectivity and Variation on Context. In this setting, consumers do not change their service demands ($P_{CD} = 0.0$). Therefore, the context of their service selections does not change during the experiments. Moreover, consumers with the similar demands have the same taste ($R_{subj} = 0.0$), so the consumers with similar service demands are satisfied with the same providers. In order to learn the effect of deception in this setting, we repeat our experiments for different percentages of liars.

Figure 4.2 shows the ratio of successful service selections in one of our experiments where only 20% of consumers are liars ($R_{liar} = 0.2$). In this setting, performances of SPS^*_{GM}, TRAVOS and BRS are almost the same. Those approaches can successfully determine satisfactory service providers. Figure 4.3 shows another experiment in the same setting where 50% of the consumers are liars ($R_{liar} = 0.5$). As shown in the figure, performances of SPS^*_{GM} and TRAVOS decrease slightly, when the percentage of liars is increased from 20% to 50%. However, in this case, the performance of BRS decrease dramatically.

Those experiments show that some of the service selection approaches are significantly affected by deceptive information disseminated by the liars in the society. In order to see the effect of deceptive information more clearly, we conduct simulations for different ratios of liars, by varying the value of the R_{liar} parameter. Figure 4.4 shows the average ratio of successful service selections through the experiments for different ratios of liars. The figure shows that the performances of SPS^*_{GM} and TRAVOS do not decrease significantly as the percentage of liars increases in the society. Although TRAVOS is slightly more sensitive to

the ratio of liars than SPS^*_{GM}, both of these approaches have a very good performance in determining satisfactory service providers. Unlike SPS^*_{GM} and TRAVOS, BRS is extremely sensitive to the percentage of liars and eventually its performance approaches to 0.0 as the ratio of liars approaches 0.8.

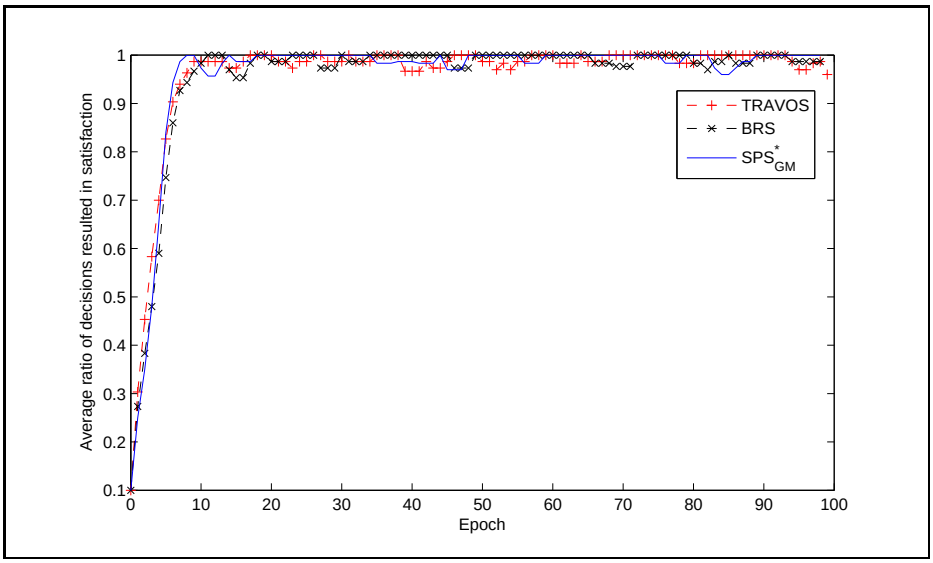

Figure 4.2. TRAVOS, BRS and SPS^*_{GM} have similarly good performances when $R_{liar} = 0.2$ and there is no subjectivity or variation on context during the experiment ($R_{subj} = 0.0$ and $P_{CD} = 0.0$)

Figure 4.4 shows that performance of BRS for $R_{liar} > 0.2$ decreases considerable with respect to TRAVOS and SPS^*_{GM}. In order to understand the reasons behind this observation, in Figure 4.5 we plot the average error in determining liars for BRS, TRAVOS and SPS^*_{GM}. As shown in Figure 4.5, when the ratio of liars becomes greater than 0.2, BRS's error in determining liars dramatically increases. This means that BRS starts misclassifying liars as honest and honest consumers as liars when the ratio of liars increases. This is an expected result because BRS is designed for environments where a significant majority of the consumers are honest. The high amount of error in determining liars implies the usage of more ratings from liars and fewer ratings from honest consumers. Therefore, in the case of BRS, filtering ratings may lead to less successful service selection. Unlike BRS, TRAVOS and SPS^*_{GM} have very low ratios of error. SPS^*_{GM} and TRAVOS fail to determine liars in at most 5% and 6% of the cases respectively, while BRS's error in determining liars approaches 100%. In this part of our evaluations, we show that BRS fails in service selection when there

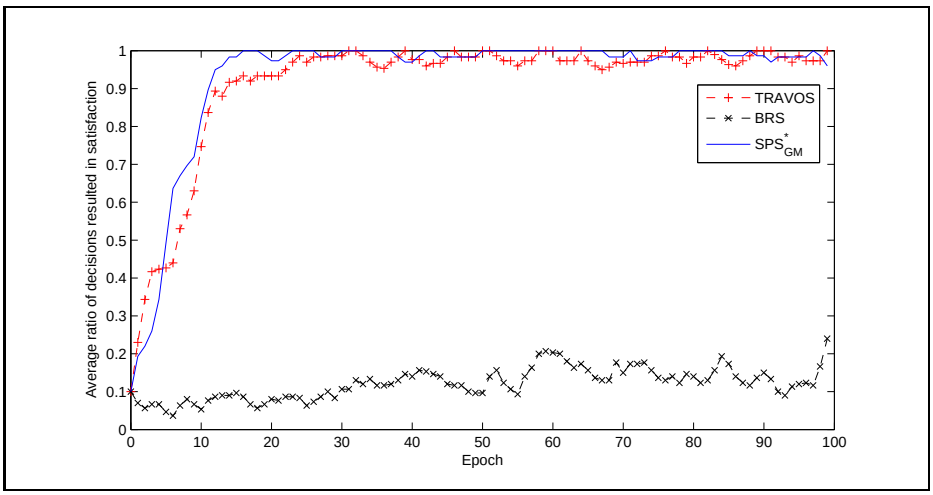

Figure 4.3. Performance of BRS sharply decreases when the ratio of liars is increased to 0.5 ($R_{liar} = 0.5$, $R_{subj} = 0.0$ and $P_{CD} = 0.0$)

are a significant number of liars in the environment. Next, we repeat our experiments for the case where consumers have different tastes for similar service demands.

4.3.3.2. Deceptive and Subjective Environments without Variation on Context. In this setting, consumers do not change their service demands ($P_{CD} = 0.0$) as in the previous setting. However, this time, half of the consumers having similar service demands have conflicting satisfaction criteria ($R_{subj} = 0.5$). Figure 4.6 demonstrates the results of an experiment where there are no liars among the consumers ($R_{liar} = 0.0$). The figure shows that performances of rating-based approaches are significantly lower than the performance of SPS^{*}_{GM} in general during the simulations. This is the effect of subjectivity on the rating-based service selection, because in the case where there is no subjectivity ($P_{CD} = 0.0$, $R_{liar} = 0.0$ and $R_{subj} = 0.0$), performances of the four service selection approaches are the same as shown in Figure 4.4. Vulnerability of rating-based approaches to subjectivity is expected, because rating-based approaches assume that there is no subjectivity among the consumers [20]. That is, they assume that every consumer gives good ratings to "good" providers and "bad" ratings to bad providers. However, in the case of subjectivity ($R_{subj} = 0.5$), the definition of "good" and "bad" depends on each consumer and may change significantly from consumer to consumer as in real life.

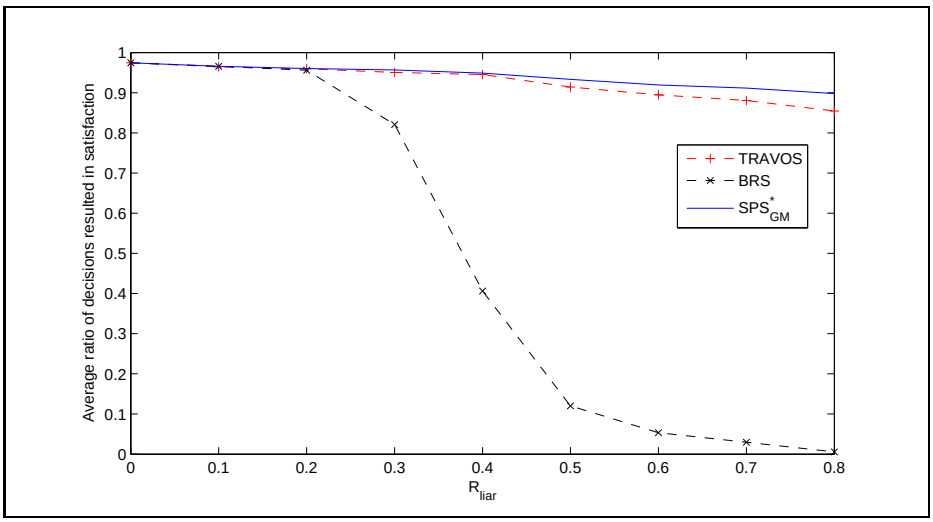

Figure 4.4. As the ratio of liars increases, the performance of BRS decreases considerably while the performances of SPS^*_{GM} and TRAVOS are slightly affected ($R_{subj} = 0.0$ and $P_{CD} = 0.0$)

In this setting, the performance of TRAVOS is much better than the performance of BRS. The main reason for this performance difference is that TRAVOS can successfully identify advisors whose ratings conflict with personal observations. In other words, TRAVOS labels advisors with conflicting taste as liars and it removes their ratings during service selection. In this way, it enables consumers to use ratings from others with similar taste. Although TRAVOS is not proposed to handle subjectivity, its mechanism of filtering out unfair ratings works well for removing subjectivity during service selection. This is because both subjectivity and deception ultimately result in consumers disseminating conflicting ratings for the same providers. Note that when subjectivity is high as in our setting, BRS has the worst performance.

Half of the consumers are liars in Figure 4.3 and have different tastes in Figure 4.6. These figures show that the effect of subjectivity is more severe than that of the deception for TRAVOS and BRS. The main reason for this observation is the fact that it is harder to determine consumers (advisors) with different taste than the liars. That is, ratings of a honest consumer and a liar for the same providers always conflict. However, if two consumers are both honest but their satisfaction criteria are different as in the case of subjectivity, their ratings conflict only for the providers that satisfy one of those consumers. On the other hand,

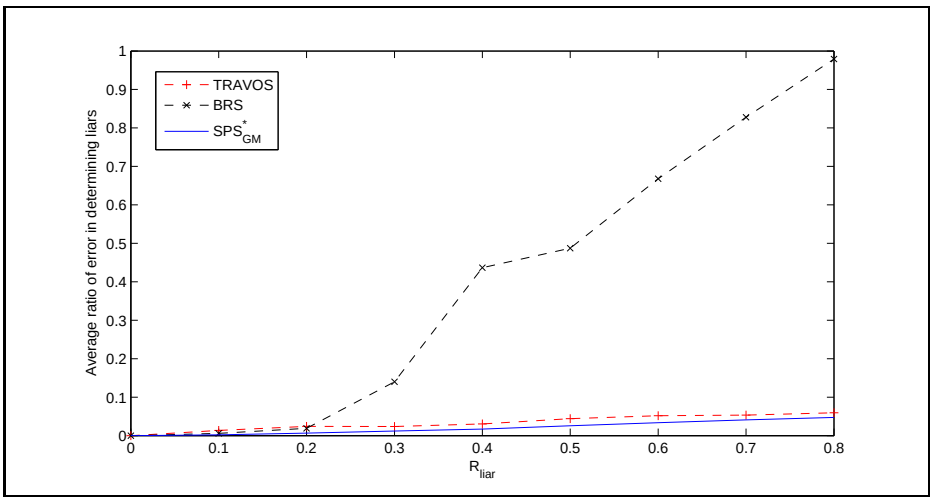

Figure 4.5. As the ratio of liars increases, BRS's error in determining liars increases dramatically with respect to the errors of TRAVOS and SPS^*_{GM} ($R_{subj} = 0.0$ and $P_{CD} = 0.0$)

ratings of those consumers are consistently negative for the other providers (ones which do not satisfy any of those consumers). For example, in our experiments, two consumers with different tastes give conflicting ratings only for two of the providers while their ratings are consistently negative for the rest. Therefore, determining consumers with different taste is more difficult than determining liars.

In many real-life settings, deceptive and subjective information exist together. In order to see the combined effect of subjectivity and deception during service selection, we change the ratio of liars when there exists subjectivity in our experiments. We show our results in Figure 4.7. Our experiments show that for SPS^*_{GM}, increasing the ratio of liars from 0 to 0.8 results in a small decrease in the ratio of successful service selections; the decrease is only from 0.96 to 0.86. If we compare the performance of TRAVOS, and BRS, we see that TRAVOS has the best performance in terms of the success in service selection. However, TRAVOS is more sensitive to deception and subjectivity than SPS^*_{GM}; its performance decreases from 0.85 to 0.42 in Figure 4.7 when the ratio of liars is increased from 0 to 0.8. Therefore, we can confidently state that SPS^*_{GM} is much more robust to deception and subjectivity than TRAVOS, and BRS.

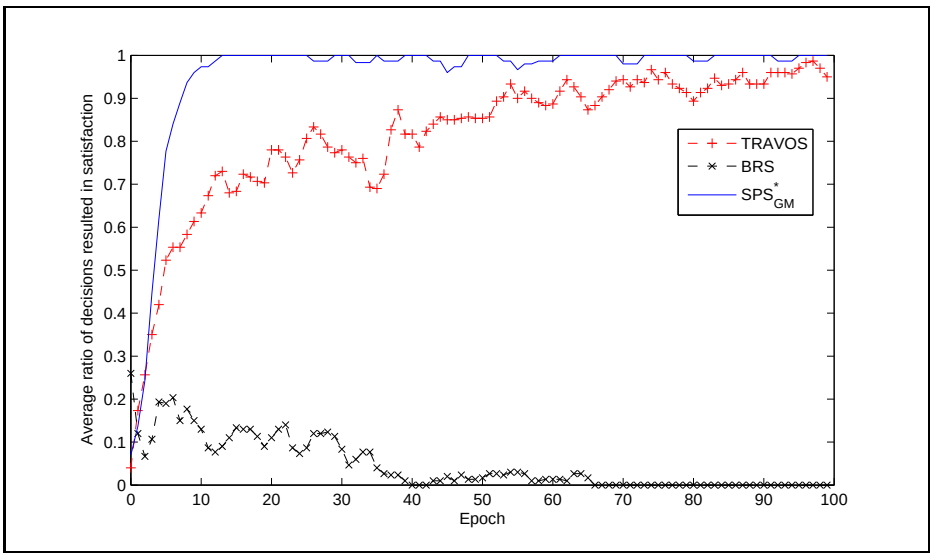

Figure 4.6. Performance of the rating-based approaches decreases in the case of subjectivity ($R_{subj} = 0.5$), even though all of the consumers are honest ($R_{liar} = 0.0$) and there is no variation on context ($P_{CD} = 0.0$)

4.3.3.3. Reliable Environments with Variation on Context and No Subjectivity. Unlike the previous settings, in this setting, consumers change their service demands with probability P_{CD} after receiving a service. Moreover, all of the consumers are honest ($R_{liar} = 0.0$) and their satisfaction criteria are similar if their service demands are also similar ($R_{subj} = 0.0$). Figure 4.8 summarizes the average ratio of successful service selections for different approaches when the value of P_{CD} is varied from 0.0 to 1.0. Figure 4.8 indicates that the performance of the rating-based approaches decreases sharply when $P_{CD} > 0$, whereas the performance of the proposed approach is near to 1.0. This sharp performance decrease is intuitive, because ratings of a consumer reflect the aggregation of its past transactions with the providers. Assume that a provider *BookHeaven* is an expert on selling books, but not competent in selling music CDs. Assume that *Bob* recently made 5 transactions for 5 items from *BookHeaven*: 2 books and 3 CDs. Because *BookHeaven* is an expert on book selling, the transactions related to the books were successful, but the transactions related to the CDs were not. In this case, the overall rating of *Bob* for *BookHeaven* is bad, because number of unsuccessful transactions is higher than that of the successful transactions. If another consumer wants to buy a book, the rating of *Bob* for *BookHeaven* will be misleading. In other words, as consumers change their demands, their ratings about the providers become more

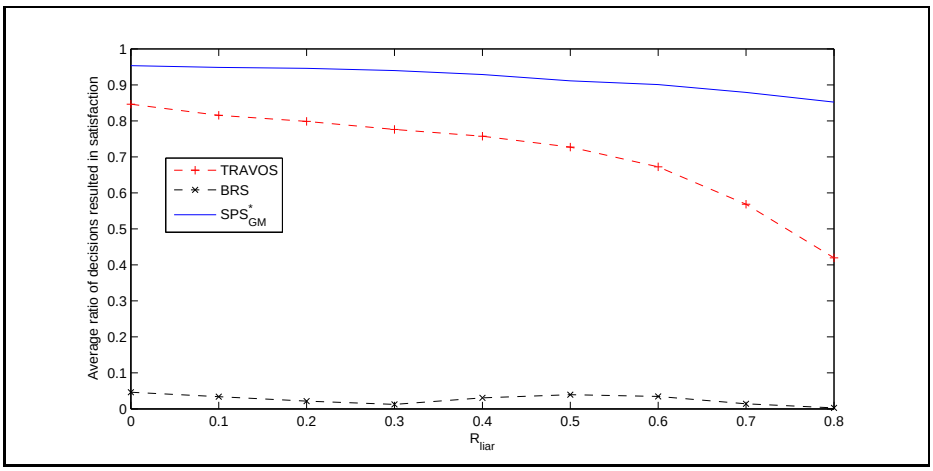

Figure 4.7. When the consumers have different tastes ($R_{subj} = 0.5$) and there is no variation on context ($P_{CD} = 0.0$), ratio of successful service selections decreases much more sharply for TRAVOS and BRS as R_{liar} increases

misleading, depending on the variation in the expertise of the providers. However, SPS^*_{GM} differentiates between the experiences belonging to different contexts. It can easily recognize that *BookHeaven* can provide a satisfactory service if a book is demanded, but it cannot produce a satisfactory service if a music CD is asked for. Hence, as seen in the Figure 4.8, SPS^*_{GM} almost always leads to satisfactory service decisions. Its ratio of successful service selections does not go below 0.94 while the performances of the rating-based approaches decrease to 0.3.

4.3.4. Summary of Results

We compare our approach with two rating-based service selection approaches from the literature; BRS and TRAVOS. In environments where there is no deception, variation on context and subjectivity, these rating-based approaches have the same performance as SPS^*_{GM}. All of the service selection approaches can successfully determine the most satisfactory service providers in this case. Unfortunately, this setting is far from being realistic in many real-life scenarios. In the case where there are liars among the consumers, the performance of BRS dramatically decreases. The decrease is sharp when the ratio of liars increases. However, TRAVOS and SPS^*_{GM} almost always make satisfactory service selections, even if most

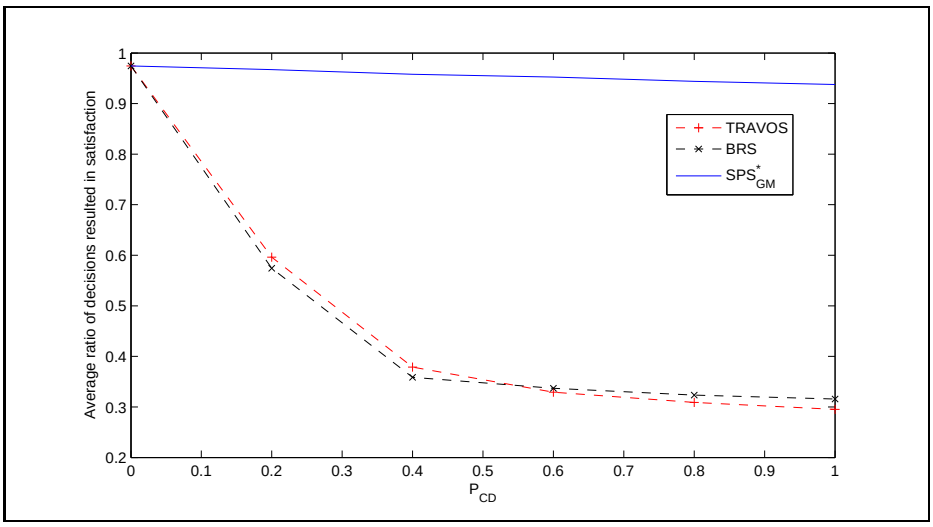

Figure 4.8. Average ratio of successful service selections decreases dramatically for TRAVOS and BRS, when the context is allowed to vary during service selection ($R_{subj} = 0.0$ and $R_{liar} = 0.0$)

of the consumers in the society are liars.

When the consumers are allowed to have different tastes for the same service demands, rating-based approaches suffer from the subjectivity. Although, subjectivity dramatically affects the performance of BRS, the performance of TRAVOS decreases only 10%. TRAVOS achieves relatively better performance than BRS by determining and eliminating ratings from the consumers having different tastes. Unlike the rating-based approaches, SPS^*_{GM} is not considerably affected by the subjectivity. This is intuitive because, unlike rating-based approaches, SPS^*_{GM} does not depend on the subjective *opinions* of other consumers during service selection. If there is not only subjectivity but also liars among the consumers, the performance of TRAVOS decreases dramatically while SPS^*_{GM} is only affected slightly in this setting.

Consumers may vary the context of their service demands regularly as in many real-life settings. Even if consumers have the same taste and are always honest, variation in context may result in serious decreases in the performances of service selection approaches. Our experiments show that rating-based approaches are very sensitive to the variation of context.

If consumers frequently change their service demands, their ratings become more confusing than before. As a result, TRAVOS and BRS fail in selecting the satisfactory service providers. On the other hand, SPS^{*}_{GM} can differentiate between different experiences depending on their contexts. Hence, our approach is not affected by the frequent changes in the context of service demands and almost always selects satisfactory service providers.

4.4. Discussion

In this section, we first compare our approach with the related work and then conclude with an overview of the chapter.

4.4.1. Related Work

Because current service selection approaches are based on ratings, deceptive information filtering problem in service selection is referred as unfair ratings in the literature [20]. Unfair ratings problem is receiving more attention with the establishment of the notion of open systems, where there is no central authority monitoring the behavior of participants in the system.

There are several approaches developed to handle unfair ratings in service selection (details can be found in Chapter 2). However, these approaches are designed to be effective only in limited situations. For example, Whitby *et al.* [50] extend the beta reputation system proposed by Jøsang and Ismail [50] to cope with unfair ratings by filtering out the ratings that do not comply with the ratings of the majority. This approach works only in the situations where the majority of the agents are trustworthy. Dellarocas [73] develops a cluster filtering approach to separate unfairly high ratings and fair ratings. However, this approach cannot handle unfairly low ratings [77]. Teacy *et al.* [52] propose the TRAVOS model to cope with inaccurate ratings provided by agents about the service providers. However, this model uses only the personal observations about the service providers and disregards the valuable public information. In the case of insufficient personal observations, this approach cannot be used. Zhang and Cohen [78] have proposed an approach to allow consumers to model the trustworthiness of other consumers. This centralized rating-based method combines consumers'

personal observations about providers and public knowledge of the others held centrally by the system. In this chapter, we adapt this approach to evaluate the trustworthiness of consumers in a decentralized way in a distributed setting on the basis of the consumers' shared *experiences*, which is context-aware and able to handle consumer subjectivity.

Our work is distinguished from the literature as follows. Our approach does not depend on ratings, which are subjective and uninformative about the context; instead our approach uses ontology-based experiences, which are objective and capture contextual information. Therefore, our approach enables the computation of trustworthiness for a specific context. Similarly, current deceptive information filtering approaches consider consumers as liars if these consumers have different satisfaction criteria. This problem is called subjectivity problem in rating-based systems. Unlike rating-based approaches, our approach can successfully handle subjectivity while computing trustworthiness. Therefore, honest consumers having conflicting satisfaction criteria are not regarded as liars in our approach.

4.4.2. Overview of the Contributions

As the number of service providers increases dramatically on the Web, it gets harder to select an appropriate provider for a particular service demand. Traditional approaches to service selection are usually based on the exchange of ratings among consumers in a multiagent system. The main challenge here is the fact that consumers' tastes and expectations may vary considerably for the same service. Therefore, ratings may be significantly misleading if the raters and the consumers using their ratings do not share similar tastes. Moreover, the problem of service selection becomes more challenging when some of the consumers disseminate deceptive information about the providers.

In this chapter, we enhance our service selection framework so that it is not only consumer-oriented and context-aware, but also robust to deceptive information disseminated by malicious consumers. In our framework, service consumers semantically describe their past experiences with service providers, so that any consumer can interpret the experiences of others using its own satisfaction criteria and context. If these experiences are not indicated truthfully, they may be misinterpreted by consumers, as they make decisions about

which service providers to select.

In order to cope with deception, we propose an approach for modeling the trustworthiness of consumers in a multiagent system – one that allows for a weighted combination of personal credit and public credit of the consumers that share their experiences, in order to determine whether the consumer is a liar. In order to be sensitive to the different satisfaction criteria that consumers have, we evaluate the experiences that are shared by consumers according to the taste function (F_{taste}) of each consumer, as part of the reasoning about deception.

We then integrate this method for detecting deception into our service selection framework, in order to filter out deceptive information. The result is an overall method for service provider selection that offers definite improvements over other methods that do not adequately account for subjectivity, context-awareness and untruthfully shared experiences together. We experimentally show that better service providers can be chosen using our approach, even if consumers have different tastes, they change context of their service demands over time or a significant portion of them are liars.

Although the proposed experience-based service selection approaches in Chapter 3 and Chapter 4 are successful in identifying satisfactory service providers, these approaches have a weakness: assumption of a static, shared service ontology among agents. This assumption cannot account for the fact that a shared service ontology may become insufficient over time, while service interests of the consumers evolve in many real-life settings. In Chapter 5, we propose a distributed approach for the evolution and maintenance of service ontologies [7,9]. This approach is not only useful for service selection in agent mediated e-commerce, but also useful for other service selection domains. Therefore, we explain it with a broader perspective considering the services in a general sense.

5. COOPERATIVE EVOLUTION OF SERVICE ONTOLOGIES

In the previous chapters, we propose different service selection approaches that are based on semantic description of consumers' past experiences, instead of ratings. While these approaches are successful in identifying service providers, they assume common service semantics. That is, these approaches assume that the agents share a service representation, such as an ontology, through which the agents can represent their service needs and past experiences in the best way possible. This assumption cannot account for the fact that service interests of the users continuously evolve in real life. This means that agents representing those users may need to use new service concepts to describe the new service interests. This requires agents to dynamically modify their vocabulary.

The need for new services stems from several facts. One, the consumer may be in need of composed services. Individual services may be expressible by the common ontology, but if the consumer always requests multiple services together, it may prefer to express its service need compositely. Example 6 demonstrates such a composite service need. Two, the service needs of the consumer may evolve in time. Thus, the consumer may actually construct a new service description that does not exist before and start demanding this service. Example 7 demonstrates such a service need.

Example 6: A consumer wants to purchase a book and wants it to be delivered to an address. Assume that there is no service concept composed of both purchasing and delivering, so the consumer creates two service instances: one for purchasing and one for delivering. Then, it collects information from other agents related to these service needs. Using the collected information, it discovers that $ProviderA$ sells books and $ProviderB$ makes deliveries. Since the consumer requests both services, it is in need of a provider that does both services for itself. However, since its ontology does not have a specific concept to represent the composed service, the consumer is left to make two individual service requests.

Example 7: Assume that a concept like valet parking is not known by any of the consumers

in the society. Assume that a consumer represents a human user which is is very busy and does not want to waste her time searching for an appropriate place to park her car. Instead, she wants a new type of parking service in which an attendant parks her car on the behalf of her. The consumer has formulated a new service to express her parking needs.

The above examples show how a consumer may become aware of a new service need and why it would prefer to add it into its ontology. However, addition of a new service concept into the local ontology of a consumer does not only affect the consumer's local ontology but also it gradually affects local ontologies and preferences of other consumers that it interacts with, as in the real life.

The need for individual ontologies to evolve on their own brings in a major problem of ontology alignment. If the consumers do not exchange information about their ontologies frequently, they can end up with disjoint ontologies, which will obstruct their communication. Hence, if consumers are allowed to add new service concepts into their ontologies, they should also be offered a mechanism with which they can notify others about their ontology changes. It is up to other consumers to decide whether these changes are of interest to them or not. That is, no consumer needs to keep track of all changes in others' ontologies. By cooperatively exchanging information about their service ontologies, consumers can build on the service concepts evolved by others, rather than reinventing the same services individually.

Accordingly, this chapter proposes a distributed approach for the creating and dissemination of new service concepts and evolution of service ontologies in the context of service selection. If a consumer cannot formulate its service need using the existing service concepts in its ontology, it first queries other consumers to find a suitable service concept that corresponds to its current service need. If such a service concept does not exist, the consumer can generate a new service concept to express its service need and inserts the concept into its service ontology. While using the new service concept when interacting with others, the consumer first checks if the correspondent agent is aware of the concept that is used in the interaction. If the correspondent consumer does not know the concept, then the consumer

sends a description of the concept to the correspondent consumer. The correspondent consumer can then add the concept into its ontology using this description and if it finds the service concept useful, it can use it in its forthcoming interactions with others. This way, the semantics of new service concepts circulate and get established in the consumer society. With this approach, we investigate the following questions:

1. How well can the consumers' service ontologies evolve so that the service needs of the consumers can be represented as concisely and accurately as possible?
2. How much of the ontology evolutions are due to individual efforts and how much are generated by cooperation?
3. How much of the learned services are useful for the consumers and how much of the useful services do they learn?

The rest of this chapter is organized as follows. In Section 5.1 and Section 5.2, we explain our insight of service ontologies. Then, in Section 5.3, we present our approach to enable consumers to agree upon the services and in Section 5.4 we propose an algorithm to add new service concepts to ontologies. Lastly, we experimentally evaluate our approach in Section 5.5 and discuss our work with references to the literature in Section 5.6.

5.1. Describing Services

Each consumer has access to a common meta-ontology that contains primitive concepts and properties. This ontology is static and does not contain any service concept. This ontology is public; i.e., may be downloaded from a well-defined resource. It constitutes a grounding for describing service concepts and sharing these description between the consumers.

Each consumer has a local service ontology. This ontology contains the service concepts known by the consumer in a specific service domain. Each service concept has a description. This description is made using only the concepts and the relations from the common meta-ontology. Therefore, each consumer can interpret this description correctly and understand what the service does. Description of a service concept contains the properties

of the service. Each property of the service is defined as a combination of an *OWL Property* and its range or value and denoted in the rest of the chapter as *(OWLProperty;RangeOrValue)*. A service concept gets its semantic meaning from its properties and from nothing else. That is, semantic meaning of a concept is not related to the syntax or lexical properties of its name. This is intuitive, because services are already defined fully by their properties in real life. That is, usually properties of a service uniquely identify it. Example 8 illustrates the description of services from their properties. As a characteristic of taxonomies in ontologies, a service concept inherits the semantic meaning of its ancestors in the service ontology. Let S_A and S_B be two service concepts with different names. These services are semantically equivalent if their sets of properties are equal. Similarly, let S_C and S_D be two semantically different service concepts. S_D is a specialization of S_C if it has every property that S_C has.

Example 8: Assume that the parking service already exists in the service ontology and its description has only one property; $(hasAction; Park)$. This means that any service that contains $(hasAction; Park)$ in its description is also a parking service. The valet parking service is described using only two properties; $(hasAction; Park)$ and $(hasActor; Attendant)$. The second property contains a restriction on the range of the OWL property *hasActor*. This restriction states that actor of the parking action must be an attendant, and not the owner of the car. If another service concept is described using exactly the same properties with the valet parking service, one can easily infer that this service is equivalent to ”valet parking”, even if it has another name.

While interpreting service descriptions, a consumer uses ontological reasoning. For example, assume that *hotel valet parking* service is described using two properties; *(hasAction;Park)* and *(hasActor;HotelAttendant)*. Again, assume that $HotelAttendant$ is a subconcept of $Attendant$ in the common meta-ontology. Therefore, the consumer can easily infer a new property *(hasActor;Attendant)* from the property *(hasActor;HotelAttendant)*. This implies that the description of *hotel valet parking* actually contains three properties; *(hasAction;Park), (hasActor;HotelAttendant)* and implicitly *(hasActor;Attendant)*. Therefore, this service is a specialization (sub-concept) of *valet parking* service, because it has the every property of valet parking.

Each consumer has a unique identifier such as a URI and a unique name-space. For example, the consumer representing *John Doe* has a unique identifier *http://agent.johndoe* and its name-space contains every name starting with this URI. When the consumer creates a new service concept such as *Valet Parking*, it gives a name to this service concept within its name-space such as *http://agent.johndoe/valetparking*. This way, name conflicts between the service concepts that are created by different consumers are prevented.

5.2. Similarity Between Services

Two service concepts can be compared to each other in terms of semantic similarity [79]. Several metrics for measuring the semantic similarity exist. We start with Tversky's similarity metric [80]. It considers the common and different properties of two concepts in the computation of similarity and assumes that two concepts are similar as much as they have common properties. However, we think that each property does not have the same importance in the computation of similarity. Therefore, we apply a weighting scheme to the properties of the concepts. In this work, similarity between two concepts a and b is computed using Equation 5.1.

$$Sim(a,b) = \frac{\sum_{p \in U(a) \cap U(b)} w_p^a}{\sum_{p \in U(a)} w_p^a + \sum_{p \in U(b) - U(a)} w_p^a} \tag{5.1}$$

In Equation 5.1, $U(a)$ is a set that contains the properties of service a including the ones inherited from its ancestor services and w_p^a is the importance of the property p for the service a. The importance of a property is determined by its level of inheritance. That is, a property is more important if it is inherited from the higher levels of the hierarchy. The intuition behind this assumption is as follows. In an ontology, if two concepts are different in terms of their most general properties (those that are inherited from higher levels in the hierarchy), these concepts will be apart in the ontology hierarchy. However, if they are different only in their most specific properties (those that are not inherited), they will be close in the ontology hierarchy. For instance, sibling concepts are different in their most specific properties. Therefore, importance of the property p for the service a (namely w_p^a) is

computed as $1 + \sum_{c \in P(a)} f(c, p)$, where $P(a)$ is the set of all ancestors of a in the service ontology, and $f(c, p) = 1$ if the service concept c has the property p ($p \in U(c)$); otherwise $f(c, p) = 0$. This means that the weight of the property p will be bigger if more ancestors of the service a has it.

5.3. Interactions of Consumers

Equipped with the above, a consumer can compare its service description to other service descriptions and find out if they are equivalent or similar. This is useful in two settings: One, when a consumer formulates a new service description, it can contact other consumers and ask them if they have previously formulated the same service description. If so, the existing URI for the service can be used, removing the necessity to create a new identifier. Two, an consumer can query other consumers for the same or similar service descriptions to find out other consumers that have had the same or similar service needs in the past. When the consumer gets hold of this information, it can contact the consumers to discuss potential service providers.

In order to get information related to its current service interest, the consumer interacts with its *neighbors*, which are defined as those consumers that are interested in the same or similar services. To select its neighbors to contact, each consumer models other consumers in the society by keeping track of their service interests from previous interactions. To do this, each consumer keeps a table called *Service Interest Table* for maintaining the list of known service concepts and the identifiers of the consumers that have used each of these concepts in their interaction with the consumer. When the consumer needs information related to a specific service, it finds table entries belonging to the most similar service concepts in its service interest table. Similarity between service concepts is computed using their descriptions as explained in Section 5.2. The consumers that are referenced in these entries are selected as the neighbors of the consumer. Then, the consumer interacts with those neighbors. During these interactions, the consumer should properly express its current service need to the other party in order to get the most related information.

5.3.1. Emergence of New Service Concepts

If a consumer encounters difficulties in representing its current service need using the service concepts in its service ontology, a new service concept referring to the service need is required. This new concept may either already exist in the society but the consumer may not be aware of it or the concept may be totally new to the entire society. In order to differentiate between these two cases, the consumer first creates a description of its desired service concept and assigns a unique name to it using its name-space. Example 9 demonstrates a simple case.

Example 9: John Doe wants to use valet parking service in a hotel during his travel. Therefore, his agent looks for the hotels that provide a valet parking service. Assume that the agent does not have a valet parking concept in its ontology. Hence, it cannot express its service need directly. As a result, it creates a description for its desired service. In this case, this description is composed of two properties; *(hasAction;Park)* and *(hasActor;HotelAttendant)*. Then, it chooses a name for the desired service concept; for example, it calls the desired concept as *http://agent.johndoe/valetparkinginhotel*.

After describing the desired service concept, the consumer sends a *Service Inquiry Message* to its neighbors to find out if the service concept is already known to its neighbors. This message contains the description of the desired service and the name assigned by the consumer. Upon receiving the service inquiry message, a neighbor inspects the service concepts in its ontology to find a semantic match with the desired service concept and informs the requesting consumer if there is a match or not. If a semantic match is found, then the neighbor sends the name of the matched concept in its ontology to the requesting consumer. Therefore, the consumer can add the desired service concept into its ontology with this name as described in Section 5.4. This way, the consumer and its neighbor address this service concept with the same name in their ontologies.

If the consumer receives different names from its neighbors for the same service inquiry message, it notes that these names are synonyms, because they refer to the same service

concept. If none of the neighbors locates the desired service concept within their service ontologies, the consumer concludes that this service concept is not known by any of its neighbors. In this case, the consumer places the concept into its local ontology with the name that is stated in the service inquiry message. Note that concept names are created within the name-spaces of the consumers. That is, it is not possible for two consumers to create two different service concepts and give the same name to them. Therefore, each concept name is unique and associated with only one service concept in the agent society. By giving unique names to the new service concepts, we remove the probability of name conflicts.

At the end of the procedure above, the consumer adds the new service concept into its service ontology. During its cooperation with its neighbors, the consumer gathers important information about the service concept. The consumer shares the gathered information with its neighbors by sending a *Service Consolidation Message*. This message contains the description of the service concept, and its name in the consumer's ontology. It also contains the identifiers of the neighbors that already know the service concept and the names of this service concept within their ontologies (referred as "synonyms"). When a neighbor receives a service consolidation message, it adds the described service concept into its ontology with the referred name if its ontology does not contain the service concept yet. Furthermore, the neighbor stores the referred synonyms to remember how the same service concept is addressed by others. Therefore, the consumer and the neighbors can understand each other during their future communications regarding this service concept.

In many approaches to ontology evolution, agents do not cooperate while creating and adding new concepts into their ontologies. Therefore, they independently add the semantically equivalent concepts with different names into their ontologies. This results in the requirement of finding mappings between the concepts in different ontologies for communicating properly. However, in the proposed approach, when a consumer generates a new service concept to represent its new service needs, it teaches this service concept to its neighbors by sharing the description of the concept or the neighbors inform the consumer about the service concept if the concept is already known by them. This leads to an interactive learning of new services. Hence, mutually understood service concepts emerge as a result of consumers' social interactions.

Moreover, this approach leads to cooperative evolution of service ontologies. When a consumer learns a useful concept from its neighbors, it can directly use it or create another concept that builds on the learned concept to describe its service needs better. Hence, more accurate concepts that describe the service needs are cooperatively and iteratively created.

5.3.2. Discovering Others

The approach explained above depends on the social interactions of a consumer with other consumers that have similar service interests. In this approach, when it needs to learn a new service concept, the consumer communicates with others that have used a similar service concept in their interactions. For example, if the consumer needs a new service that is a type of valet parking offered by hotels, it communicates with the consumers that are interested in valet parking services or hotel services. These consumers are determined using the service interest table of the consumer. In many cases, the agent may need to expand this table by discovering new consumers with a specific service interest. In this section, we propose a simple P2P search mechanism for this purpose.

In order to get the identifiers of the consumers with a specific service interest, the consumer generates a *Search Message*. This message contains the identifier of the message originator, the name of the service concept that represents the service interest, the desired number of search results that should be returned by the receiver and lastly a time-to-live (TTL) value to define how long the message should be forwarded. Then, using its service interest table, the consumer chooses a subset of its neighbors to whom the search message will be sent as explained in Section 5.3.1. Figure 5.1 shows a search message. In this message, the message originator states that it is looking for five consumers that are interested in *HotelValetParking* or a similar service. In the figures and tables throughout this chapter, we omit the name-space prefixes for the service concept names (e.g., http://agent.johndoe/), because of space limitations.

When another consumer receives this message, it checks whether the service concept in the message is known or not. If this concept is not known by the receiver yet, it requests for the description of the service concept from the message originator and adds the concept

into its service ontology as explained in Section 5.4. The receiving consumer processes the message as follows. First, it computes the similarity of the service concept in the message to the service concepts in its service interest table. Then, it sends the table entries belonging to the most similar concepts along with the concept names to the originator of the search message and the message originator updates its service interest table using these entries. The receiver decides the number of entries to be sent using the number defined in the search message. The receiver also updates its service interest table by adding the name of the message originator to the related service concept entry. As a result, the receiver remembers the service interest of the message originator and uses this information in the future. If TTL value of the message is greater than one, the receiver decrements the TTL value of the message and forwards the message to its neighbors that are most related to the service concept in the message.

Using these simple interactions, consumers learn service interests of each other, update their service interest tables, and use these tables in their future interactions. During the interactions of the consumers, if unknown service concepts are encountered by a consumer, the consumer requests for the description of these concepts from the consumers that have used these concepts in their interactions. Then, these concepts are added to the ontology of the consumer using the procedure in Section 5.4.

Received Search Message	
Originator	agent.johndoe
Service Concept	HoteValetParking
DesiredResults	5
TTL	3

Service Interest Table	
ConceptName	**Consumers**
Parking	Cons0;Cons3;Cons10;Cons12
ValetParking	Cons0;Cons16
HoteValetParking	Cons11;Cons23;Cons34
HotelService	Cons11,Cons20

Similarity to HotelValetParking	
Parking	0.4286
ValetParking	0.7143
HoteValetParking	1.0
HotelService	0.5714

Returned Search Results	
ConceptName	**Consumers**
HoteValetParking	Cons11;Cons23;Cons34
ValetParking	Cons0;Cons16

Figure 5.1. An example search message, service interest table of the receiver, the computed similarities, and the returned search results

5.4. Updating Service Ontologies

A new service concept is added into a service ontology using its description and ontological reasoning. In a service ontology, each service concept is described in terms of its properties. An example service ontology is shown in Figure 5.2. Each service concept inherits the properties of its ancestors. A property implies another one if it is a specialization of the latter. For example, in Figure 5.2, *HotelValetParking* concept inherits *(hasActor;Attendant)* from *ValetParking* and *(hasActor;HotelAttendant)* from *HotelService*. The latter property implies the former, because *HotelAttendant* is defined as a sub-concept of *Attendant* in the shared meta-ontology. Therefore, *HotelValetParking* can be described using only two properties; *(hasActor;HotelAttendant)* and *(hasAction;Park)*.

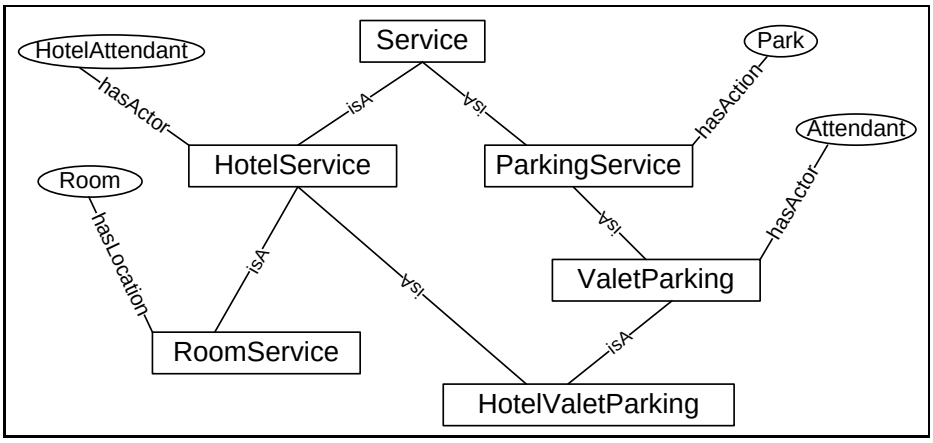

Figure 5.2. Service ontology of a neighbor

In this chapter, we assume that service concepts are precisely described by their properties. Hence, if the properties of two service concepts are equivalent, then they are semantically equivalent. Consider Example 9. The agent in the example describes its desired service concept using two properties; *(hasActor;HotelAttendant)* and *(hasAction;Park)*. When the neighbor referred in Figure 5.2 gets this description, it can trivially match the desired concept with *HotelValetParking* concept in its ontology.

A consumer learns new service concepts using their descriptions. When the consumer receives a description of a service concept S_{new}, it searches for a semantic match in its ontology using the approach above. If the consumer cannot find a semantically equal service

concept in its ontology, S_{new} should be added as a new concept into the service ontology of the consumer. For this purpose, the consumer first defines the parent(s) of S_{new} in its ontology. A service concept S_P in the consumer's ontology is a parent of S_{new}, if S_{new} is a specialization of S_P, but not that of S_P's any subconcept. As stated before in Section 5.1, S_{new} is a specialization of S_P if every property of S_P can be inferred from the properties of S_{new}. Example 10 shows how parents of a new service concept are determined.

Example 10: Agent of *John Doe* generates a new service concept and gives this concept a name, *http://agent.johndoe/restaurantvaletparking*. Then, the agent wants to teach this concept to the agent referred in Figure 5.2. It defines this concept using two properties; *(hasAction;Park)* and *(hasActor;RestaurantAttendant)*. Assume that *RestaurantAttendant* is a sub-concept of *Attendant* in the shared meta-ontology. When the agent referred in Figure 5.2 gets this description, it can conclude that this service is a specialization of *Service*, *ParkingService*, and *ValetParking* concepts. However, only the most specific one, namely *ValetParking*, is the parent of the new service concept. As a result, the agent adds the new concept into its ontology as a sub-concept of *ValetParking* with the referred name.

After defining the parents of S_{new}, it is placed into the ontology as a sub-concept of these parents. Some of the properties of S_{new} are also owned by its parents. Therefore, the new concept inherits these properties from its parents. However, S_{new} may have some other properties that are not owned by its parents in the ontology. These properties are directly added to the new concept in the ontology. Example 11 shows how a new service concept is placed into the ontology.

Example 11: The agent in Example 10 adds the new service concept as a subconcept of *ValetParking* and uses the referred name *http://agent.johndoe/restaurantvaletparking* as its name. This new service concept inherits *(hasAction;Park)* from its ancestors, but not *(hasActor;RestaurantAttendant)*. So this property is attached to the new service concept in the ontology.

Lastly, we may need to modify some of the existing parent-child relationships of S_P,

which is a parent of S_{new}. Let S_X be a sub-concept of S_P. If S_X is also a specialization of S_{new}, by definition S_P cannot be the parent of S_X any more. Instead, S_{new} should be the new parent of S_X in the ontology. Therefore, we modify the parent-child relationships accordingly.

Consider an example scenario in which the neighbor referred in Figure 5.2 teaches service concepts from its ontology to a consumer. This consumer has only one service concept in its ontology. This concept is named *Service* and does not have any property. Figure 5.3 shows the initial service ontology of the consumer and its change after the addition of *HotelValetParking*, *ValetParking* and *HotelService* concepts, respectively.

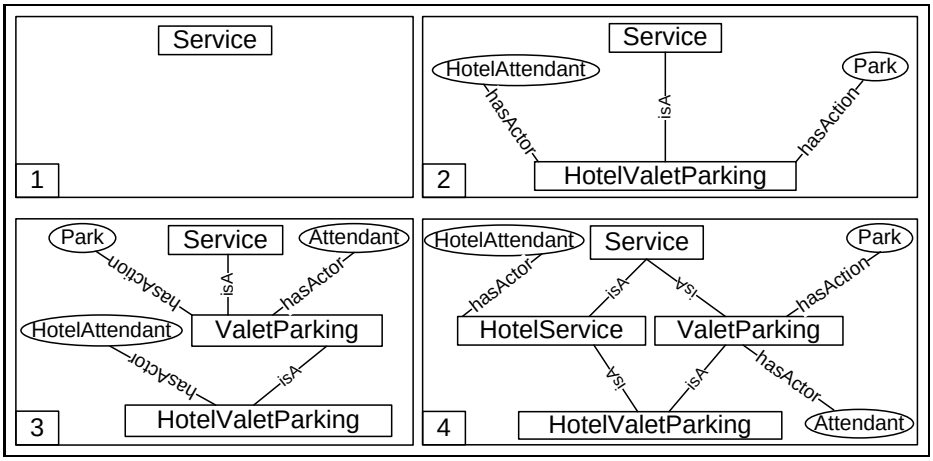

Figure 5.3. Service ontology of the consumer; (1) initial, (2) after adding *HotelValetParking*, (3) after adding *ValetParking*, (4) After adding *HotelService* concept

When the consumer receives the description of *HotelValetParking* from the neighbor, it adds *HotelValetParking* as a subconcept of *Service*. Note that any service concept is trivially a specialization of *Service*, because *Service* does not have any property. In this example, the new service concept does not inherit any of its properties from its parent in the ontology, so all of its properties are directly attached to it as shown in the second sub-figure of Figure 5.3.

When the consumer receives the description of *ValetParking* from the neighbor, it determines that *Service* is the parent of this new service concept. Hence, *ValetParking* is also added into the ontology as a subconcept of *Service*. Description of *ValetParking* contains two properties; *(hasAction;Park)* and *(hasActor;Attendant)* . These properties can be in-

ferred from the properties of *HotelValetParking*. Therefore, *HotelValetParking* is set as a sub-concept of *ValetParking*. Note that, now *HotelValetParking* inherits *(hasAction;Park)* from its parent (see the third sub-figure of Figure 5.3).

Lastly, the consumer receives the description of *HotelService* from the neighbor. It has only one property; *(hasActor;HotelAttendant)*. This service concept is also added into the ontology as a subconcept of *Service*. In this time, previously added service concept *HotelValetParking* is its specialization. Therefore, *HotelValetParking* is set as a sub-concept of *HotelService* as shown in the fourth sub-figure of Figure 5.3. Notice that now *HotelValetParking* has two parents and inherits all of its properties from those parents.

5.5. Experimental Results

We have conducted simulations to evaluate our approach. In the simulations, 100 consumer agents are used. In the beginning of the simulations, each consumer agent is given identities of three randomly chosen consumers to overcome bootstrapping. Initial interactions are done with these agents. Throughout the simulations, service inquiry or search messages are sent to at most two neighbors and TTL values of the search messages are set to 1, because of the small size of the simulated agent society.

Consumer agents are implemented in Java and JENA[10] is used as OWL reasoner. Simulations are run on a computer with a 1.66 GHz Intel Core Duo CPU and 1.0 GB RAM under Windows OS. Next, we describe the simulation environment and the results of our simulations. Simulations are repeated 10 times in order to increase the reliability. Only the mean values of the results are demonstrated in this section.

5.5.1. Simulation Environment

For the simulations, we select *Food and Beverage Services* domain. In real life, different restaurants have different service offerings, which are usually called *Food Menus*. Each service concept in this domain is described by a list of foods and beverages that are served

[10]http://jena.sourceforge.net

or delivered to a consumer together for a meal. For example, *KFC Hot Wings Menu* is an instance of *Chicken Wing Menu* concept that contains a number of fried chicken wings, a bunch of fried potatoes, and a cup of drink.

To facilitate meaningful generation of service needs, we have designed roles for agents. These roles are similar to the real life roles such as student, parent, and so on. These roles define the behaviors of consumers by specifying their service needs and characteristics. For example, for dinner, the consumers playing *vegetarian* role usually demand a cup of vegetable soup, some pasta or rice, and a salad. In our simulations, there are 10 distinct roles and each agent is assigned exactly one role. Each agent has a personality ontology that contains information about the role that the agent plays. The consumer agent can reason about its service needs using this ontology and shape its behaviors and preferences appropriately. In other words, the roles enable generation of service needs during the simulation. Roles of the consumers are set at the beginning and properties of the roles do not change during a simulation. This means that consumers playing the same role continuously have the same *service needs* during the simulations.

For the simulations, we extended W3C's food ontology [25] and we use the resulting ontology as the shared meta-ontology. This ontology has more than 200 concepts, 1020 individuals and 60 properties. In the beginning of simulations, each consumer agent has the same service ontology. This ontology contains only 10 shared service concepts such as pizza service, salad service and so on. Each of these service concepts contains only one type of food. For example, an instance of pizza service contains an instance of pizza and nothing else. However, each service need that is imposed by the roles is composed of several food types. Hence, none of these shared service concepts is enough to represent a service need on its own. Therefore, in the beginning of the simulations, a consumer having a service need must request several services together to satisfy its service need. For example, in the beginning, the consumer playing *vegetarian* role must demand several services together such as soup service, pasta or rice service and salad service, because there is not a service concept that contains a soup, rice or pasta, and a salad together. This may result in problems in real life. For example, you want to have some soup and pasta. However, there is not a food service that contains soup and pasta together. Instead, there is a soup service offered by a

restaurant and a pasta service that is offered by another restaurant. Hence, you have to make two different orders from two different restaurants. If the pasta arrives much earlier than the soup, you should either eat pasta before the soup or wait for the soup and let the pasta get cold.

Table 5.1 shows the service concepts and the roles that use these concepts in the beginning of the simulations. During the simulation, each consumer tries to express its service need as concisely as possible (i.e., using a single concept). To do so, at each epoch, the consumer tries to create a new service concept with a small probability (0.1). A new service concept is created by either combining two existing service concepts as in Example 6 or adding a new property to an existing service concept as in Example 7. If the service concept to be created is learned from others, the consumer uses the learned service concept rather than recreating it.

Table 5.1. Service concepts and roles using them in the 0^{th} epoch

Service Concept Name	Roles Using The Service Concept
SoupService	Role5, Role4, Role2, Role0
DessertService	Role8, Role7
AlcoholicBeverageService	Role8, Role5, Role4, Role0
NonAlcoholicBeverageService	Role9, Role8, Role7, Role6
PizzaService	Role7, Role6, Role2, Role0
PastaService	Role9, Role6, Role2
RiceService	Role4, Role3, Role2, Role1
SaladService	Role5, Role4, Role3, Role0
SeaFoodService	Role6, Role5, Role4, Role2, Role1
MeatService	Role8, Role6, Role5, Role3, Role1, Role0

5.5.2. Simulation Results

With the above setup, we first investigate whether the consumers have actually been successful in formulating their service needs concisely as desired. Table 5.1 show the initial

configuration of the environment, where consumers use around four service concepts to describe their service needs. However, in our simulations, this number sharply decreases over time and on the average it becomes 1 in the 15^{th} epoch. That is, after this point, new service concepts emerge so that every consumer can and does use only one service concept to describe its service need. The first plot in Figure 5.4 shows that starting from the $15th$ epoch, the ratio of new service concepts used by a consumer reaches 1; meaning that any service concept used by a consumer to describe its service need is a new service concept (i.e., did not exist in the starting service ontology).

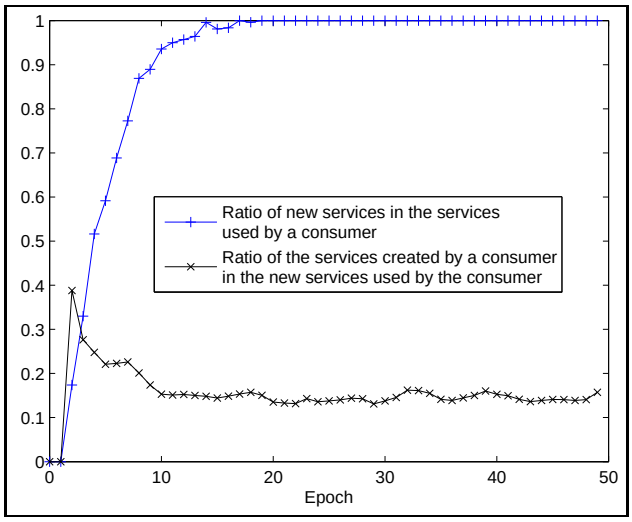

Figure 5.4. Average ratio of adoption and creation of new service concepts by the consumers

For example, the consumers playing $Role7$ cooperatively create a new service concept before 15^{th} epoch and use only the instances of this new service concept to describe their needs thereafter. This new service concept contains a dessert, an alcohol-free beverage, and a pizza. This confirms that our approach can enable consumers to create new and more expressive service descriptions to satisfy their service needs.

Next, we study whether consumers create accurate service concepts based only on their experiences or whether they have benefited from interactions with others. This is a crucial point, since the proposed approach is meant to help agents evolve their ontologies *cooperatively*. The second plot in Figure 5.4 shows the ratio of the service concepts created by a consumer to the new service concepts used by the consumer. In the beginning, this ratio

is 0.4, which means that on the average 40% of the new service concepts used by a consumer are created by the consumer. On the other hand, this ratio rapidly decreases and approaches 0.1, which means that only 10% of the new service concepts used by a consumer is created by the consumer. The remaining service concepts are created by other consumers and adopted by this consumer. This proves that there is a powerful cooperation between the consumers. They teach each other new service concepts and furthermore the taught service concepts are adopted and used to describe service needs of the consumers in the society. As stated before, this situation leads to the cooperative creation of new service concepts, because new service concepts are created using the learned ones.

Another interesting question is how well each agent becomes aware of the service concepts created by other agents. To study this point, we make a distinction between *unique* and *replica* service concepts. A service concept is unique at the time it is created if nobody in the system has created another service concept with the same meaning. A service concept is a replica if an equivalent service concept has been previously created by another agent in the system before (but not known by the creator of the service concept).

Figure 5.5 shows plots related to this question. The first plot is the number of new service concepts created through the simulations. After the 15^{th} epoch, there are 92 new service concepts in the environment. This number includes the unique service concepts as well as the replicas. The second plot, on the other hand, shows only the number of unique service concepts, which is equal to 75. Put together, these numbers reveal that 82% of the new concepts in the environment are unique and only 18% of them are replicas. This result confirms that consumers become aware of the useful service concepts instead of reinventing them by creating replicas.

Next, we study how much of the service concepts are learned by an agent on the average and whether the agent misses out important service concepts in the society. The third plot in Figure 5.5 is the average number of unique service concepts known by a consumer. Consumers do not have the knowledge of whether a service concept is unique or replica, because each consumer has a limited knowledge of the environment. If the consumer knows two service concepts with the same meaning, it considers the one that is learned earlier as

unique and the other service as replica (synonym). The dashed line in the figure shows that the average number of unique service concepts known by a consumer stabilizes at 11 after the 15^{th} epoch. This shows that the consumers know only a small portion of the service concepts in the environment.

The last result shown in the figure is the average number of known unique service concepts that are useful for the consumer. In this case, we count the unique service concepts known by a consumer only if these service concepts are useful for the consumer in defining its service needs. For example, for the consumers playing $Role7$, a service concept containing only a pizza and an alcohol-free beverage is useful, so we count this service concept for the consumers playing $Role7$. However, we do not count this service concept for the consumers playing $Role2$, because these consumers do not demand alcohol-free beverages with pizza (see Table 5.1). Figure shows that the number of useful unique service concepts for a consumer is around 8 out of the 11 unique service concepts it knows. These two observations show that (1) consumers learn a small proportion of the service concepts in the environment, and (2) this small proportion is enough for agents to represent their service needs. Hence, they learn only enough to express their needs, and no more.

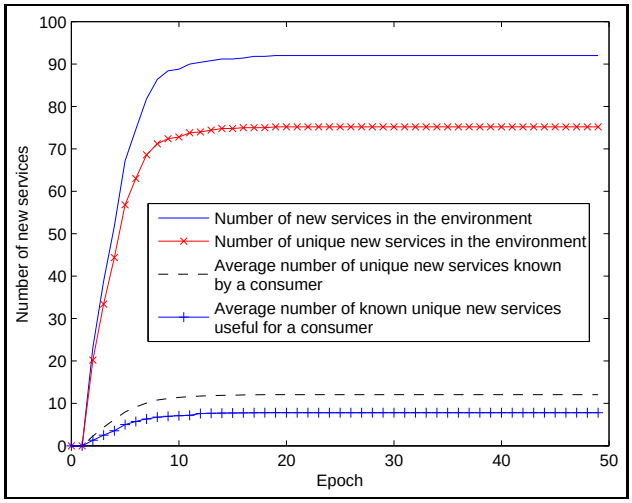

Figure 5.5. Number of new service concepts, unique new service concepts, known service concepts, and useful known service concepts for a consumer

A concrete conclusion of Figure 5.5 is that most of the service concepts learned by consumers are useful. An immediate question is how much of the useful and frequently used

service concepts are known by a consumer. This is important to find out since knowing useful service concepts that are not used in communication is rarely of value. Figure 5.6 plots the average ratio of *useful* unique concepts known by a consumer to the *useful* unique concepts in the environment. To calculate this, we weight every service concept in the environment by considering whether the service concept is useful for describing the consumer's service needs as well as how frequent the service concept is used. The frequency of service concepts is calculated by counting the number of occurrences of the service concepts in communications of all the consumers. As Figure 5.6 shows the ratio of known useful service concepts dramatically increases and approaches 0.9, which means that each consumer knows most of the useful unique service concepts, which are frequently used by the consumers in the society.

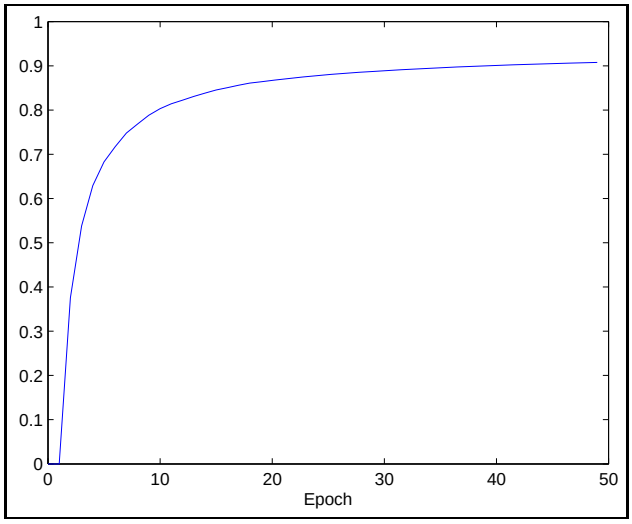

Figure 5.6. Average ratio of known useful service concepts

5.6. Discussion

This chapter proposes a novel approach for cooperative evolution of service ontologies. Our simulations show that consumer ontologies evolve considerably so that at the end of the simulations each consumer is able to represent its service need as concisely as possible. The evolutions are not only due to individual effort but mostly a result of cooperation. That is, most of the service concepts used by each consumer are devised by others in the society. Furthermore, the consumers learn a small portion of the service concepts that emerge in

the society, making sure that most of the learned service concepts are useful for them in representing their service needs. Finally, consumers learn most of the useful, frequently used service concepts in the society. Thus, through cooperation, agents can have different but evolving ontologies, yet they can communicate with those that are similar to them to represent their needs.

5.6.1. Extensions to the Proposed Approach

Main contribution of our research is the techniques that are proposed in Section 5.3 for the cooperative evolution of service ontologies in a P2P setting. In this chapter, we assume that service concepts can be described by the agents using a *shared meta-ontology* that is composed of fundamental concepts and properties. This ontology should carefully be designed by domain experts so that it can be used to describe any new service concept in the domain. This may require considerable amount of human effort and may not be possible in some settings. The proposed techniques in this chapter can also be used together with other concept description models. However, each concept description model has its own advantages and disadvantages over the description model that is used in this chapter. For example, in the example-based concept learning literature, a concept is described using its instances (positive examples) and its non-instances (negative examples) [81, 82]. Although this concept description model does not require a shared meta-ontology, it requires instances to be shared among agents (e.g., through a public directory). We can easily integrate this example-based description model into the proposed framework.

For example, in Web Services domain, we can describe *Book Selling* service concept using its positive and negative examples. In this case, UDDI [65] entries related to book selling concept (e.g., Amazon book selling service) can be used as the positive examples and UDDI entries that are not related to book selling (e.g., New York Times newspaper selling service) can be used as the negative examples. When an agent gets this description, it can learn the *Book Selling* concept using the examples in the description as explained in [81, 82]. Furthermore, it can compute the semantic similarity or equivalence between two service concepts using these examples and machine learning techniques as described by Doan *et. al.* [83]. Doan *et. al.* also offer a method for testing subsumption between two concepts

using examples. This is equivalent to testing whether a concept is a specialization of another concept or not. Therefore, a service concept can easily be placed into an ontology using its example-based description, using a method similar to the method in Section 5.4. Using the approaches above, we can easily replace our concept description model with the example-based concept description model. Similarly, we can switch between different concept description models to use our framework in a broader range of real-life applications.

Example-based concept learning can easily be integrated into this work as shortly described above. Although the quality of concept learning highly depends on the quality of the given examples, previous approaches do not specifically address the problem of how to select the most useful examples for the learner. In [8], as an extension of the research in this chapter, we extend current example-based concept learning approaches using a novel active learning approach. The proposed approach enables agents to teach each other concepts from their ontologies using examples. Unlike other example-based concept learning approaches, our approach enables the learner to elicit the most informative and useful examples interactively from the teacher. Hence, the learner participates to the learning process actively. We empirically compare the proposed approach with the previous concept learning approaches. Our experiments show that using the proposed approach, agents can learn new concepts successfully and with fewer examples.

5.6.2. Related Work

The necessity for individual ontologies to evolve on their own results in a major problem of ontology alignment. There are substantial amount of research related to ontology alignment and reconciliation in the literature [82–86]. Similarity-based approaches for ontology alignment are powerful and flexible enough for aligning ontologies expressed in languages like OWL. In these approaches, similarity between the concepts from two different ontologies is computed and the concepts that are similar to each other are mapped to align these two ontologies. Some other approaches use syntactic or lexical properties of concepts' names in addition to semantic similarity metrics. They use string matching algorithms or lexical databases like WordNet. We believe that semantic similarity is much more important than syntactic similarity, because similarity or dissimilarity between the names of concepts

may be highly misleading.

Afsharchi *et. al.* [82] use an instance based approach for learning concepts. In that setting, when an agent confronts an unknown concept, it chooses teacher agents among its neighborhood and these teachers teach the concept to the agent by providing positive and negative examples of the concept. Then, the agent uses a machine learning approach to learn the properties of the new concept. This approach is similar to the approach of Sen and Kar [81] in the sense that agents teach each other concepts by providing examples. This approach is not practical for cases where there is not a sufficient number of instances shared by the agents.

Stephens *et. al.* [84] propose an approach for the reconciliation of independent ontologies. They argue that if two ontologies share no concept in common, they cannot be reconciled. However, if they share concepts with a third ontology then the third ontology might provide a semantic bridge to relate these ontologies. Their approach makes use of different techniques such as string matching and lexical databases in order to measure the semantic distance between two concepts. However, they do not allow ontologies to evolve cooperatively as we have done here.

Williams [85] proposes a methodology and algorithms for improving the mutual understanding of two agents. In this approach, agents develop a common feature description of a particular concept using knowledge sharing and machine learning techniques in a peer-to-peer setting. So, they gradually arrive at consensus on the concepts and they develop mappings between the concepts in their ontologies. However, Williams' approach does not support cooperative ontology evolution.

Steels [87] allows differences in the ontologies of the agents. Steels uses a method called language games to minimize these differences and generate shared ontologies from the differentiated ontologies. The language games depend on a trial and error phase in the communication. In practical applications, this may not be possible all the time.

Aberer *et. al.* [86] propose an approach for the global semantic agreements. They

assume that mappings between two different ontologies are already made by skilled human experts. These mappings are exchanged by the agents and global semantic agreements are reached using the properties of the exchanged mappings. Laera *et. al.* [88] use argumentation over concept mappings to reach global semantic agreements. Similar to Aberer *et. al.*, in Laera *et. al.*'s approach, mappings between concepts are assumed to be made beforehand by a mapping engine. Each mapping has a confidence value for different agents. By using argumentation theory together with these mappings, an agreement over heterogeneous ontologies are reached dynamically.

Our work distinguishes from the literature in several ways. In the other approaches, ontologies of agents evolve independently. Each agent creates new concepts on its own and adds them into its ontology. This results in highly different ontologies. Then, the mentioned approaches are used to align these ontologies. In our approach, ontologies evolve cooperatively. That is, a consumer learns new service concepts from its neighbors and furthermore creates new ones using the learned service concepts. As a result, not only more useful service concepts emerge over time, but also service ontologies of the consumers having similar service needs become aligned over time. Additionally, our approach has a proactive nature. In our approach, a consumer prohibits its future communication problems by informing its neighbors about the created service concepts before using them.

In Chapter 3, we develop several service selection approaches and augment these approaches with mechanishs for filtering deceptive information and ontology evolution in Chapter 4 and Chapter 5, respectively. Current research on service selection in the literature show that there is no single solution for the problem of service selection that works perfectly in every setting and for every consumer. Each solution has different strengths and weaknesses for different settings. For example, the proposed context-aware service selection approaches in this dissertation are better than rating-based approaches in terms of the achieved satisfaction, but they require consumers to record their experiences objectively using an ontology. This results in the problem of choosing the best service selection approach for a particular consumer and its environment. In Chapter 6, we introduce this problem in detail and propose an approach for consumers to learn how to choose the most useful service selection mechanisms among different alternatives in dynamic environments [10].

In this approach, consumers continuously observe outcomes of different service selection mechanisms. Using their observations and a reinforcement learning algorithm, consumers learn to choose the most useful service selection mechanism with respect to their trade-offs. Moreover, this approach enables consumers to switch between different service selection mechanisms intelligently, as the properties of the environment changes dynamically.

6. ON CHOOSING AN EFFICIENT SERVICE SELECTION MECHANISM

In a real-life implementation of consumer agents, there should be at least one service selection mechanism embedded in the agent. This service selection mechanism is used by the agent to decide on a service provider to interact with. However, which service selection mechanism to implement in an agent is a difficult task for an agent developer for at least two reasons.

First, each mechanism has its own advantages and disadvantages over the others under different settings. In our previous research (see Chapter 3), we have observed that if the consumers in the environment have almost the same taste, providers satisfying one consumer tend to satisfy another consumer, too. Rating-based mechanisms are simple and fast as well as effective in terms of achieved satisfaction in this setting. On the other hand, if the tastes of consumers are highly different, rating-based mechanisms may have a bad performance in terms of achieved satisfaction [3, 4] (they are inversely affected by the subjectivity). This means that when the environment is static and consumers have similar preferences, rating-based mechanisms may give perfect results in a fast way in such an environment. On the other hand, an experience-based mechanism may be preferable if the consumers' preferences and tastes vary. Hence, under different situations, the same service selection mechanism has highly different utilizations for the same consumers.

Second, the expectations of consumers from a service selection mechanism may vary. For example, one consumer may trade off time for performance. That is, the consumer may prefer to find service providers faster even when that means the best service provider will not be found. On the other hand, another consumer may prefer the slower mechanism because of the fact that its achieved satisfaction is higher.

Ideally, the consumer agent should be able to choose the service selection mechanism to use based on its current trade-offs as well as its observation of the environment. To accomplish this, we propose agents to learn the pros and cons of service selection mechanisms

to use in dynamic environments and to decide on the mechanism at run time. As a learning technique, we employ reinforcement learning (RL) since it allows agents to learn from their actions in the environment [89]. Through simulations, we show that using our proposed approach, consumers can successfully learn to select the most useful mechanisms for service selection under different configurations of the environment.

The rest of this chapter is organized as follows: Section 6.1 explains the proposed approach, with necessary background knowledge. Section 6.2 gives a brief overview of service selection mechanisms that are used in this work and provides their performance evaluation. Section 6.3 provides our experimental results. Finally, Section 6.4 summarizes our contribution and compares it to relevant literature.

6.1. Learning to Choose Among Service Selection Mechanisms

In real life, given a number of service selection mechanisms, we expect service consumers to pick a mechanism that is most suitable for their current situation. For example, if the consumer is in a hurry, it might prefer a service selection mechanism that will find it a service provider quickly; whereas at other times, the quality of the found service provider may be more important. Making choices between service selection mechanisms would be trivial if agents could observe and characterize their environment perfectly. However, this is rarely the case. Most often, agents are not aware of the service demands of other agents, those agents' past experiences, their expectations and so on. Interestingly, the performance of most service selection mechanisms (time and quality-wise) are dependent of these characteristics of the environment. Therefore, we need to provide consumer agents with means to detect which mechanism to use in their environment to satisfy QoS constraints such as the time required for service selection or desired ratio of satisfaction.

In different configurations of the environment, a service selection mechanism may have different utilizations for a consumer. Therefore, if the agent finds out that its environment is changing, then the consumer should switch to another service selection mechanism that has better utilization for that specific setting. For this purpose, the consumer should continuously observe the environment and should learn the best service selection mechanism for itself

under different configurations of the environment. This intuition becomes more concrete when we analyze the different parameters of an environment.

6.1.1. Reinforcement Learning

Reinforcement learning (RL) [89] is an ideal learning technique to enable agents to learn the environment and thus decide on which strategy to use in that particular situation. That is why we propose to use RL for choosing a service selection mechanism in dynamic environments.

Let us review the basic structure of an RL algorithm. In RL, there is a learning agent and an environment that is observed by this agent. The environment has a set of discrete states. At each state, there exists a set of available actions for the agent. When the agent takes an action in a state, it receives a reward in turn and also the observed state of the environment may change as a result of this action. The action to be taken in each state is determined by the policy adopted by the agent. After a set of trial-and-error interactions with the environment, the agent learns an optimal policy for choosing the best action in a given state of the environment so that the total reward is maximized [89].

In RL, the environment is defined by a finite set of states $S = \{s_1, s_2, \ldots, s_N\}$, and the agent has a finite set of actions $A = \{a_1, a_2, \ldots, a_M\}$. In this setting, the agent observes the state of the environment $s_t \in S$ in time t, and chooses an action $a_t \in A$, and receives a scalar reward r_{t+1} after executing a_t. Each state has a value in terms of the maximum discounted reward expected in that state. Value of a state is formulated in Equation 6.1. Thus, the purpose of RL is to construct an optimal action policy that maximizes the expected discounted reward in each state. In the equation, γ, ($0 < \gamma < 1$), is a discount factor so that distant rewards are less important. The equation expresses that expected value of a state, s_t, is the weighted sum of the rewards received when starting in the state s_t and following the current policy.

$$V(s_t) = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+1}\right] \tag{6.1}$$

The action selection at each step is based on Q-values, which are related to the goodness of the actions. The Q-value, $Q(s, a)$, is the total discounted reward that the agent would receive when it starts at a state s, performs an action a, and behaves optimally thereafter. The Q-values can be estimated by Temporal Difference Learning (TD) [90] methods such as Q-Learning [91] and SARSA [89]. In this work, we choose to use SARSA reinforcement learning algorithm [89] (shown in Figure 6.1). SARSA has two important advantages compared to other approaches [92]. First, it has better convergence guarantees compared to other RL approaches such as Q-learning. Secondly, it learns much more rapidly and in the early part of learning, its average policy is better than that of the other RL approaches.

6.1.1.1. SARSA Algorithm. In the algorithm shown in Figure 6.1, initial Q-values are set to zero and the current state s is determined (Line 1). While the current state is not the terminal state of the agent (Line 2-9), the agent selects an action a using a policy (π) derived from the Q-values (Line 3). There may be different ways of deriving a policy from Q-values. One of the most popular approaches is $\epsilon - greedy$. In this approach, with probability ϵ, we choose an action uniformly randomly among all possible actions. This is called exploration. As well as, with probability $1 - \epsilon$, we choose the action having the maximum Q-value ($\pi(s) = max_{a_i} Q(s, a_i)$) and this is called exploitation. We do not want to explore indefinitely, so we start with a high ϵ value and continuously decrease it at each time step (e.g., $\epsilon = 1/t$). After determining a using π, a is executed (Line 3). As a result an immediate reward is taken and a new state is observed s' (Line 4). SARSA algorithm is different from other TD approaches such as Q-Learning in a way that it uses the current policy to update Q-values. Therefore, in the algorithm, the action that should be taken in state s' is also determined using the policy π (Line 5). Then, s, a, r, s', and a' are used to update $Q(s, a)$ (Line 7). In the update formula of Q-values, γ is the discount factor and η is the learning rate ($0 < \eta < 1$). Discount factor is used to weight immediate rewards more heavily than future rewards. The closer γ is to 1 the greater the weight of future rewards. The learning rate controls how much weight we give to the recent reward, as opposed to the old Q-value estimate. We typically start with a high learning rate and lower the learning rate as time progresses. After updating Q-value, s and a are updated properly for the next time step (Line 8).

1: Initialization: initialize all Q(s,a) value to zero and observe the current state s
2: **while** (s is not terminal state) **do**
3: Select an action a using policy derived from Q (e.g., using $\epsilon - greedy$) and execute it.
4: Observe immediate reward r and the new state s'
5: Choose the action a' in state s' using policy derived from Q (e.g., using $\epsilon - greedy$)
6: Update $Q(s, a)$:
7: $Q(s, a) \leftarrow Q(s, a) + \eta \times (r + \gamma \times Q(s', a') - Q(s, a))$
8: $s \leftarrow s'$, $a \leftarrow a'$
9: **end while**

Figure 6.1. SARSA reinforcement learning algorithm

6.1.1.2. Reward Function. The reward function (r) that we use with SARSA algorithm is given in Equation 6.2. In the equation, $E(S_{action})$ and $E(T_{action})$ denote the expected ratio of satisfaction and the expected time required for the chosen action, respectively. This reward function reflects the trade-offs of the service consumers. According to the reward function, a consumer is given a negative reward after choosing an action if there is another action with an expected ratio of satisfaction that is at least 10% better than that of the chosen action. Even though there is no such action, the consumer gets a negative reward again if the chosen action is at least 10% slower than another action whose ratio of satisfaction is at most 1% worse than that of the chosen action. If none of these cases occurs, the agent gets a positive reward. Using this reward function, consumers learn to choose fast service selection strategies without trading off the ratio of satisfaction more than 10%.

We select this reward function deliberately because of two reasons; (i) in many settings, cost of choosing a bad service provider may be higher than choosing a good provider in long time. In the first case, a consumer loses money, while in the second case it loses time. Hence, consumers may prefer much slower service selection strategies if these strategies are significantly more successful than the others (e.g., their success rate is at least 10% better than others) (ii) for many consumers, time is also very valuable. Therefore, they prefer a faster service selection strategy over others if its service selection performance is not considerably lower than that of the other strategies (e.g., at most 1% lower). Altough the agents can use any reward functions reflecting their trade-offs, we believe that the reward function in Equation 6.2 successfully represents the general considerations cited above.

During its life-time, the consumer continuously computes and revises $E(S_{action})$ and $E(T_{action})$. For each action, the consumer records the time required for the action and the result of the action in terms of the degree of satisfaction. By simply averaging these values over time, the consumer computes $E(T_{action})$ and $E(S_{action})$, respectively. In order to handle highly dynamic environments, the consumer uses only the recent information in its computations (i.e., information from the last 20 time steps).

$$r(act) = \begin{cases} -1 & \text{if } \exists X | \frac{(E(S_X) - E(S_{act}))}{E(S_{act})} > .1 \\ \\ -1 & \text{if } \exists X | abs(\frac{(E(S_X) - E(S_{act}))}{E(S_{act})}) < .01 \wedge \frac{(E(T_{act}) - E(T_X)))}{E(T_X)} > .1 \\ \\ +1 & \text{otherwise} \end{cases} \tag{6.2}$$

6.1.2. Discretization of Continuous State Space

In our setting of RL, states of the environment are different configurations of the environment and the actions are different service selection strategies. We can model the states of the environment using the P_{CD}, R_{subj}, and PI parameters, as described in Chapter 3. However, agents cannot observe these parameters directly, but they can observe the result of their actions. Intuitively, there is a correlation between the performance of the service selection strategies and these parameters. This intuition enables us to describe states of the environment using the expected ratio of satisfaction of the different service selection strategies, instead of the parameters like P_{CD}, R_{subj}, or PI. For example, if we consider two service selection mechanisms A and B, a state can be characterized by a consumer using the expected success of A and B. Therefore, the consumer observes the states of the environment as the pairs of different $E(S_A)$ and $E(S_B)$ values. However, this state space is continuous and needs to be converted into a discrete state space for RL.

By dividing continuous state space to discrete states, we are trying to map different configurations of the environment to a number of discrete states. If the number of discrete states is too small, then highly different configurations of the environment will be mapped to

the same discrete state. This will decrease the accuracy of the RL algorithm considerably. If the number of states is set to a number that is much bigger than the optimal number of states, then it will take too much time to train the RL algorithm and at the end, some states will have almost identical Q-values, because of the redundancy of the states. To tackle this problem, we propose to decide on the number of discrete states dynamically. Initially, we start with only one discrete state. We continuously observe the environment and try to estimate the possible states that the environment may be in. Then, we group these possible states into clusters by using *k-means* clustering algorithm. Each cluster is represented by one discrete state. If any cluster of the possible states is too diverse to be represented by only one discrete state, then this cluster is divided into two clusters and a new discrete state is created to represent the new cluster of possible states. Q-values of the discrete state representing the divided cluster is used as the initial estimates of the new state's Q-values. This way, we incrementally increase the number of discrete states only if it is required. This approach has several advantages. First, it prohibits the introduction of redundant states. Second, this approach does not require the knowledge of maximum number of states. Third, this approach increases the convergence rate of Q-values by incrementally introducing the new states and by using pre-computed Q-values as the initial estimates of these states' Q-values. Algorithm in Figure 6.2 presents a modified version of SARSA algorithm with the proposed discretization of continuous state space.

$$wcv_i = \frac{\sum_{L \in c_i} \left[\sum_{j=0}^{n-1} |L[j] - center_i[j]| \right]}{|c_i|} \tag{6.3}$$

In this algorithm, every state has a label. This label is an n-tuple, where n is the number of available service selection mechanisms and each element of the n-tuple is of $(E(S_{X_i})$: the expected ratio of satisfaction of the service selection mechanism X_i. Initially we have only one state and its label is set to $(0, \ldots, 0)$ (Line 1). During the life time of the agent, it continuously computes expected ratio of satisfaction for each action and creates a label $Label_t$ at time t using computed $E(S_{X_i})$ values (Lines 5–6). $Label_t$ is added to *state-space set* (Line 7). This set is initiated as an empty set (Line 1) and populated with $Label_t$ at each time step. Labels of the states and $Label_t$ are used to determine the next state after taking an action. After taking an action, the distance between computed $Label_t$ and the labels of

the states are compared. The state having a label closest to $Label_t$ is set as the current state, s' (Line 9). With a probability P_{update}, we update state labels by running k-means algorithm [62] on state-space set. We initiate k-means algorithm with k centers (k is the number of states). These initial centers are the current labels of the states (Line 15). New clusters and cluster centers are defined after running k-means algorithm. Each state s_i has a corresponding cluster c_i. We compute within-cluster variance, (wcv_i) for each cluster c_i using the center of the cluster and the labels within the cluster (see Equation 6.3). If, for any cluster c_i, wcv_i exceeds a predefined threshold (Line 16), a new state is created (s_{new}) (Line 17). One of the labels in c_i is set as the label of s_{new} (Line 18) and Q-values of s_i is copied to s_{new}'s Q-values (Line 19). After creation of new states, we run k-means algorithm again (Line 21). This procedure is repeated until wcv for each cluster is below the threshold. Then, we set the center of each cluster as the new label of the corresponding state (Lines 23–25). Using this approach, we increase the number of states only it is required. Furthermore, increasing the number of states does not affect the performance of the RL algorithm much, because we initialize Q-values of new states with the Q-values of the most similar states. As time advances, Q-values of new states are differentiated by SARSA algorithm.

6.2. Comparison of Service Selection Mechanisms

To evaluate the proposed approach, we perform simulations using four service selection mechanisms that the agents can choose from at run time according to their trade-offs. These service selection approaches are described in Chapter 3 in detail. The first two mechanisms are the rating-based service selection approaches; *Selective Ratings* (SPS_{SR}) and the proposed *Context-Aware Ratings* (SPS_{CAR}). The second two mechanisms are the proposed experience-based service selection approaches based on *Gaussian Model* (SPS_{GM}) and *Case-Based Reasoning* (SPS_{CBR}).

In Chapter 3, we demonstrate the performance of these service selection mechanisms in terms of their success in service selection. In this section, we also tabulate their average time requirements during service selection. We have two metrics to evaluate each service selection mechanism. Our first metric is the average ratio of decisions resulted in satisfaction (S) and our second metric is the average time required for selecting a service provider (T).

1: Initialization: Create s_{init} with Label $\{0, \ldots, 0\}$, initialize all $Q(s_{init}, a)$ values to zero, $States \leftarrow \{s_{init}\}$, $StateSpace \leftarrow \{\}$
2: $s = s_{init}$
3: **while** (alive()) **do**
4: Select an action a in state s using policy derived from Q, and execute it
5: Update $E(R_{X_i})$ for all of n mechanisms using the data in the last N time steps
6: $Label_t \leftarrow \left\{E(R_{X_0}), \ldots, E(R_{X_{(n-1)}})\right\}$
7: add $Label_t$ to $StateSpace$
8: $r \leftarrow getReward()$
9: $s' \leftarrow min_{s_i} \sum_{j=0}^{n-1} |s_i.Label[j] - Label_t[j]|$
10: Choose the action a' in state s' using policy derived from Q
11: Update $Q(s, a)$:
12: $Q(s, a) \leftarrow Q(s, a) + \eta \times (r + \gamma \times Q(s', a') - Q(s, a))$
13: $s \leftarrow s'$, $a \leftarrow a'$
14: **if** $(Rand() < P_{update})$ **then**
15: KMeans(StateSpace,States)
16: **while** $(\exists c_i | wcv_i > Threshold)$ **do**
17: Create new state s_{new}
18: select one label from c_i and assign it as the Label of s_{new}
19: $s_{new}.QValues \leftarrow s_i.QValues$
20: add s_{new} to $States$
21: KMeans(StateSpace,States)
22: **end while**
23: **for** each $s_i \in States$ **do**
24: $s_i.Label \leftarrow c_i.center$
25: **end for**
26: **end if**
27: **end while**

Figure 6.2. Modified version of SARSA algorithm with the proposed discretization of continuous state space

For different values of the parameters P_{CD}, R_{subj}, and PI, Table 6.1 and Table 6.2 show the performance of each mechanism in terms of the achieved satisfaction and time requirements, respectively. Table 6.1 summarize that service decisions of a consumer usually results in satisfaction of the consumer if the consumer uses SPS_{GM}. Moreover, the performance of SPS_{GM} in terms of ratio of satisfaction does not change much with different values of P_{CD}, R_{subj}, and PI. On the other hand, as shown in Table 6.2, time requirements of SPS_{GM} is much more than that of other mechanisms. Performance of each service selection mechanism is equally good in terms of the achieved satisfaction when P_{CD}, R_{subj}, and PI are set to zero. However, increasing P_{CD} results in a serious decrease in the performance of SPS_{SR} in terms of the ratio of satisfaction. Performance of the mechanisms other than SPS_{SR} is not affected by P_{CD} considerably. Performances of SPS_{SR} and SPS_{CAR} in terms of the ratio of satisfaction decrease considerable as R_{subj} increases. SPS_{SR} is faster than SPS_{CAR} in all cases. Unlike the experience-based mechanisms, the rating-based mechanisms achieve a

low ratio of satisfaction as R_{subj} increases [3], because satisfaction criteria of the consumers get more differentiated as R_{subj} increases. Table 6.2 reveals that time requirements of the rating-based mechanisms are much less than that of the experience-based mechanisms, because aggregation of experiences is much more costly than aggregation of ratings in terms of time. Performance of SPS_{CBR} in terms of the ratio of satisfaction is quite well for $PI = 0$ and does not change significantly when P_{CD} or R_{subj} changes. Furthermore, in terms of time performance, SPS_{CBR} outperforms SPS_{GM}. On the other hand, the performance of SPS_{CBR} in terms of the ratio of satisfaction significantly decreases if nondeterminism is observed ($PI > 0$).

In summary, SPS_{GM} is the best service selection mechanism when the ratio of satisfaction is concerned. Unlike the performance of SPS_{GM}, performance of other mechanisms in terms of the ratio of satisfaction is badly affected as P_{CD}, R_{subj}, or PI increases. P_{CD} affects the performance of SPS_{SR}, R_{subj} affects the performances of SPS_{SR} and SPS_{CAR}, and lastly PI affects the performance of SPS_{CBR}. Order of the mechanisms from the fastest one to the slowest one is SPS_{SR}, SPS_{CAR}, SPS_{CBR}, and SPS_{GM}. This order does not change as P_{CD}, R_{subj}, or PI changes.

Table 6.1. Service selection performances of different mechanisms

P_{CD}	R_{subj}	PI	S_{SR}	S_{CAR}	S_{CBR}	S_{GM}
0	0	0	0.9617	0.9616	0.9616	0.9616
0.2	0	0	0.6287	0.9671	0.9670	0.9670
0	0.5	0	0.5687	0.5930	0.9551	0.9551
0.2	0.5	0	0.3189	0.4838	0.9599	0.9599
0	0	0.01	0.9636	0.9271	0.6369	0.9630
0.2	0	0.01	0.6135	0.9229	0.6421	0.9650
0	0.5	0.01	0.5687	0.5798	0.4892	0.9579
0.2	0.5	0.01	0.3064	0.4965	0.4785	0.9592

Table 6.2. Average time required for service selection mechanisms in milliseconds

P_{CD}	R_{subj}	PI	T_{SR}	T_{CAR}	T_{CBR}	T_{GM}
0	0	0	1.78	7.55	448.11	1250.75
0.2	0	0	2.26	21.24	981.59	5376.60
0	0.5	0	1.54	5.45	269.48	1080.53
0.2	0.5	0	1.96	13.78	674.10	4291.46
0	0	0.01	1.51	8.53	497.75	1623.09
0.2	0	0.01	1.63	23.47	1050.02	6163.32
0	0.5	0.01	1.33	5.66	363.83	1217.55
0.2	0.5	0.01	1.34	6.57	401.70	1737.57

6.3. Experimental Evaluation

In this section, we evaluate our approach for choosing the most suitable service selection mechanisms in different configurations of the environment using simulations. We use the same simulation settings in Chapter 3. However, each consumer agent is given four different service selection mechanisms; SPS_{SR}, SPS_{CAR}, SPS_{CBR}, and SPS_{GM}. These consumers use the proposed approach to learn how to choose the most profitable service selection mechanism in their environment. In order to make our evaluations easily understandable, we assume that consumers have the same trade-offs between the time requirements and ratio of satisfaction. These trade-offs are embedded in the reward function that is explained in Section 6.1.1.2.

For different configuration of the environment, actions of a consumer are rewarded differently. For example, in case of $R_{subj} = 0$ and $P_{CD} = 0$, a consumer should choose SPS_{SR}. In this configuration of the environment, SPS_{SR} is much faster than other mechanisms, even though performance of other mechanisms in terms of ratio of satisfaction is not significantly better than that of SPS_{SR}. Therefore, the consumer gets a positive reward if it chooses SPS_{SR}; otherwise it gets a negative reward. However, if the value of P_{CD} increases to 0.2, performance of SPS_{SR} in terms of ratio of satisfaction decreases significantly. In

this case, SPS_{SR} is still much faster than other mechanisms, but there are other mechanisms that have a ratio of satisfaction at least 10% better than that of SPS_{SR} (see Table 6.1 and Table 6.2). Therefore, choosing SPS_{SR} results in a negative reward.

The best of the four service selection mechanisms in terms of the ratio of satisfaction is SPS_{GM}. Hence, we measure the performance of our approach in terms of the performance of SPS_{GM}. When a consumer has a service demand, it uses the proposed approach to chooses a service selection mechanism using the computed policy as explained in Section 6.1. Then, the consumer selects a service provider using the chosen mechanism. We measure the average ratio of satisfaction (S_{RL}) and the average time required for service selection (T_{RL}) when the proposed approach is used. We also measure the average ratio of satisfaction (S_{GM}) and the average time required for service selection (T_{GM}) if the consumers had used only the SPS_{GM} in their service selections.

Table 6.3 shows the results of our simulations. The first three columns refer to the parameters that are used to configure the simulation environment. Next two columns are the average ratio of satisfaction for SPS_{GM} and the proposed RL approach, respectively. The last column is the ratio of T_{GM} to T_{RL}. In the first eight rows, there is only one environment configuration during the simulations. On the other hand, in the last row, environment gradually changes starting from the first configuration to the eighth configuration during the simulation.

In the first configuration of the environment ($P_{CD} = 0$, $R_{subj} = 0$, $PI = 0$), all of the service selection mechanisms have almost the same ratio of satisfaction. Therefore, the proposed approach chooses SPS_{SR} and SPS_{CAR} most of the time, because performance of these two mechanisms are very good in terms of both the achieved satisfaction and the time required for service selection. In this setting, using the proposed approach, the consumers can make as satisfactory service selections as they do using SPS_{GM}. Moreover, using the proposed approach the consumers make service selections 74 times faster than they do using SPS_{GM}.

In the second configuration of the environment ($P_{CD} = 0.2$, $R_{subj} = 0$, $PI = 0$),

Table 6.3. Performance of the reinforcement learning approach

Configuration ID	P_{CD}	R_{subj}	PI	S_{GM}	S_{RL}	T_{GM}/T_{RL}
1	0	0	0	0.9616	0.9616	74.43
2	0.2	0	0	0.9670	0.8940	45.09
3	0	0.5	0	0.9551	0.9227	8.79
4	0.2	0.5	0	0.9599	0.9040	6.29
5	0	0	0.01	0.9630	0.9562	86.25
6	0.2	0	0.01	0.9650	0.8598	9.72
7	0	0.5	0.01	0.9579	0.8989	2.84
8	0.2	0.5	0.01	0.9592	0.8588	1.17
9	0-0.2	0-0.5	0-0.01	0.9751	0.9123	4.5

SPS_{SR} is not as good as other strategies any more. Hence, the proposed approach does not prefer SPS_{SR}. Although SPS_{CAR}, SPS_{CBR} and SPS_{GM} have almost the same performance in terms of achieved satisfaction, the proposed approach chooses SPS_{CAR} most of the time. This is due to the fact that SPS_{CAR} is much faster than SPS_{CBR} and SPS_{GM}. As a result, using the proposed approach, consumers select satisfactory services in around 90% of their service selections. This is achieved in a way that 45 times faster than SPS_{GM}.

In the third and fourth configurations, performances of SPS_{SR} and SPS_{CAR} are worse than that of SPS_{CBR} or SPS_{GM} in terms of achieved satisfaction, because both of them are badly influenced as R_{subj} increases. As a result, the proposed approach chooses either SPS_{CBR} or SPS_{GM} in service selection. The primary choice of our approach is SPS_{CBR}, because it is much faster than SPS_{GM}. As a result, the proposed approach achieves satisfaction in 92% and 90% of the cases in a way 8.79 and 6.29 times faster than SPS_{GM}, respectively for these two configurations.

In the fifth configuration ($P_{CD} = 0$, $R_{subj} = 0$, $PI = 0.01$), SPS_{CBR} has a low ratio of satisfaction, because SPS_{CBR} is badly influenced as PI increases. Therefore, our approach chooses between SPS_{SR}, SPS_{CAR} and SPS_{GM}, which have almost the same

ratio of satisfaction in this setting. Most of the time, SPS_{SR} is chosen by our approach, because it is much faster than SPS_{CAR} and SPS_{GM}. As a result, the proposed approach achieves almost the same ratio of satisfaction as SPS_{GM} does in a way 86.25 times faster than SPS_{GM}.

In the sixth configuration of the environment ($P_{CD} = 0.2$, $R_{subj} = 0$, $PI = 0.01$), only SPS_{CAR} and SPS_{GM} have a desirable performance in terms of the ratio of satisfaction. In this case, the proposed approach chooses the faster mechanism, SPS_{CAR}, rather than SPS_{GM}. Hence, the proposed approach achieves satisfaction in 86% of the cases. In this setting, the proposed approach results in service selections that are 9.72 times faster than that of SPS_{GM}.

In the seventh and eighth configurations, the performance of SPS_{SR}, SPS_{CAR} and SPS_{CBR} are much worse than that of SPS_{GM} in terms of the ratio of satisfaction. As a result, our approach chooses mostly SPS_{GM}. In these settings, the proposed approach achieves satisfaction in 90% and 86% of the cases, respectively for these two configurations.

We lastly conduct simulations in which configuration of the environment is changed from the first configuration to the eighth configuration. In this challenging case, using the proposed approach, consumers select satisfactory services in 91% of their service selections in a way 4.5 times faster than SPS_{GM}.

The worst performance of the proposed approach in terms of achieved satisfaction is in the eighth configuration. In this case, S_{RL} is around 10% worse than S_{GM}. Hence, in any configuration of the environment, the proposed approach is not outperformed significantly more than 10% by any service selection mechanism, in terms of the ratio of satisfaction. This is the trade-off exactly stated in the reward function.

6.4. Discussion

Service selection mechanisms vary in their success rate as well as time requirements. Existence of different service selection mechanisms gives rise to a major problem: how to

choose the most appropriate service selection mechanism among a range of alternatives. This problem is amplified when the environment is dynamic. Our proposed approach allows agents to learn how to choose the most useful service selection mechanism among different alternatives in dynamic environments. The proposed approach examines the environment and learns to differentiate between different service selection mechanisms with respect to the trade-offs of the consumers. Our experimental results show that consumers choose the most useful service selection mechanism using the proposed approach. The performance of the proposed approach does not significantly go below the lower-bound defined by the trade-offs of the consumers. Even in the case that the environment changes dramatically, the proposed approach exhibits a good performance in terms of both the ratio of satisfaction and the time required for service selection.

To the best of our knowledge there is no approach for choosing one selection mechanism among different alternatives. However, there are various service selection mechanisms proposed in the literature. Some of these approaches are shortly described in the previous chapters. These approaches are stand alone approaches and do not concern choosing a useful service selection mechanism among different alternatives as we do in this work.

Techniques for combining different classifier exist in the literature. For example, Ortega *et. al.* propose an approach that takes a collection of existing classifiers and learning algorithms and creates a combined classifier that takes advantage of all of these classifiers [93]. The basic idea of their approach is that each classifier has a particular sub-domain for which it is most reliable. By combining different classifiers, Ortega *et. al.* create a classifier that is reliable for a range of different sub-domains. Although this type of approaches are similar to our approach in a sense that they combine different mechanisms to come up with a better one, they are specifically proposed for the classification problems and they cannot be used directly in the service selection problem that we are addressing in this chapter.

7. CONCLUSION

This dissertation concludes with a review of the work presented with respect to the objectives set in Chapter 1. Lastly, some directions for future work are suggested.

7.1. Dissertation Overview

As the number of service providers increases dramatically on the Web, it gets harder to select an appropriate provider for a particular service demand. Although Semantic Web technologies such as ontologies and ontological reasoning [21] are promising for intelligent, consumer-oriented and context-aware selection of service providers, current approaches for service selection follow fairly traditional ways. The most widely used service selection approaches depend on capturing and manipulating ratings [20]. In rating-based approaches, the consumers rate the service providers and share their ratings with other consumers. Then, the shared ratings are aggregated to determine the most satisfactory service providers.

Ratings are simply the numbers that do not say anything about the context they belong to. Hence, rating-based service selection approaches fail when the providers have highly different service characteristics (e.g., QoS) in different context. Moreover, ratings reflect the subjective opinion of the raters. Therefore, ratings may be significantly misleading if the raters and the consumers using their ratings do not share similar tastes or the context.

In this dissertation, we develop a flexible approach for context-aware service selection in four steps. First, we propose an ontology-based representation to enhance classical ratings with semantic description of their context in Chapter 3. Using this representation, ratings can be differentiated depending on their context. This way service selection performance of rating-based approaches are increased for the cases that service consumers change the context of their service demands. However, this approach fails when consumers have conflicting satisfaction criteria. Second, we replace over-all ratings with detailed experiences of consumers. An experience roughly corresponds to machine understandable form of a user review on the Web in that it captures an episode of the customer with a provider and

can be thought of as a record of what service the customer has requested and received in return. Contrary to user reviews, an experience contains only objective information rather than subjective evaluations. For instance, while a user review in real life may state that the price of the experienced service was fair, an experience states only the actual price of the service (e.g., $100). Hence, any consumer receiving an experience can semantically evaluate the service provider according to its own satisfaction criteria using the detailed data in the experience. For example, while for one consumer, $100 is fair for the service, for another one it is exorbitant. This way, subjectiveness inherent to the ratings are removed from the service selection. Resulting service selection approach is context-aware and consumer-oriented in a way that it enables consumers to use their own satisfaction criteria and context during service selection. We empirically show that this experience-based service selection approach leads to satisfactory service selections even if the satisfaction criteria and the context of consumers vary considerably over time.

Third, we show that context-aware service selection is sensitive to deceptive information in Chapter 4. That is, when malicious consumers disseminate untruthful experiences about the providers, service selection performance decreases dramatically. To handle deception during service selection, we propose a filtering approach by which experiences received from untrustworthy consumers are filtered out during service selection. A consumer computes the trustworthiness of another consumer by comparing the experiences received from this consumer with its personal experiences and the experiences received from the other consumers in the society. This way, public and private knowledge about the providers is combined to compute trustworthiness of the consumers. We integrate the filtering approach to the proposed service selection approach. The result is an overall method for service selection. We empirically compare our approach with well know approaches from the service selection literature [50, 52] and show that it offers definite improvements over other methods that do not adequately account for subjectivity, context-awareness and untruthfully shared information together.

The proposed service selection approach is superior to the rating-based service selection approaches at a cost of using a richer representation. That is, in order to use the proposed approach, consumers should use a common semantics such as a shared ontology, which is as-

sumed to be enough for representing their service needs and past experiences. In literature, a static shared ontology among the agents is usually assumed to facilitate communication between the agents. Therefore, we make the same assumptions until Chapter 5. However, service needs of a consumer may evolve over time and the service concepts within the static ontology may rapidly become insufficient to represent the consumer's new service needs and experiences. Therefore, as the fourth step, we propose a distributed approach for evolution and maintenance of service ontologies for service selection in Chapter 5. In this approach, a consumer queries its neighborhood to learn a suitable concept if the concepts in its local ontology are not sufficient to describe its current service need. If such a concept is not known by its neighbors, the consumer creates the concept, and teaches it to its neighborhood. This way, the consumer prohibits its future communication problems by informing its neighbors about the created concept before using it. Our experiments show that the proposed approach leads consumers to have different but overlapping service ontologies as their service needs evolve. However, they can still communicate with each other, because service ontologies of the consumers having similar service interests are aligned over time.

With these contributions, we fulfill all of our objectives set in Chapter 1. The results is an overall approach for service selection that is not only flexible, consumer-oriented and context-aware, but is also robust to deceptive information disseminated by malicious consumers.

There is no single solution for the problem of service selection that works perfectly in every setting and for every consumer. Each solution has different strengths and weaknesses for different settings. For example, the proposed service selection approach in this dissertation is better than rating-based approaches in terms of the achieved satisfaction, but it requires consumers to record and interpret their experiences using an ontology. This results in the problem of choosing the best service selection approach among alternatives for a particular consumer and its environment. Therefore, apart from the objectives set in the Chapter 1, we introduce a new research problem in Chapter 6 and we lastly propose a meta method for consumers to learn how to choose the most useful service selection mechanism among different alternatives in dynamic environments with respect to their trade-offs. Through the simulations, we show that not only the consumers choose the useful and efficient service

selection mechanism using the proposed method, but also the service selection performance of the proposed method does not go below the lower-bound defined by the consumers' trade-offs.

7.2. Real-Life Applications

This dissertation develops a service selection approach that has many practical benefits for real life. In this approach, human users are represented by consumer agents that may continuously interact with their human users through graphical user interfaces in a real-life application. These agents semantically describe their users' past interactions with the service providers. Then, using these semantic descriptions of past experiences, the agents can make satisfactory service decisions depending on their own context and satisfaction criteria. Therefore, this approach paves the way to industrial applications that enable consumer-oriented, context-aware and flexible service selections.

In practice, the agents may be developed by various software companies. However, they can still communicate and cooperate to make service selection if they comply with the proposed OWL ontologies and the communication protocols. Human users can download these agents from software repositories and then these agents can interact with their human users to learn their service demands, satisfaction criteria for these demands and past experiences. Based on the gathered information, the agents can make service decisions on behalf of their human users as explained in this dissertation.

Our approach depends on agents to discover others in a P2P setting. In the proposed protocols, agents may have some initial acquaintances to start communication. In real life, this can be achieved using a directory that the agents can sign in when they enter to the system. That is, when an agent enters the system, it can register its identity to the directory and receive some initial list of acquaintances to interact with. These initial acquaintances may be chosen randomly or some heuristics may be used to choose them for a specific agent (e.g., based on location, service interests and so on). Using these initial acquaintances, the agents may discover other agents having similar service demands and may collect experiences related to these service demands by interacting with the discovered agents.

Currently, there are many centralized rating-based systems to select service providers (e.g., the reputation system of e-Bay). However, the system proposed in this dissertation does not depend on ratings. Therefore, we cannot directly use the accumulated information about the providers in these legacy systems. Hence, the proposed system may have a bootstrapping problem in real life. That is, initially nobody will have enough experiences to guide others in the system. On the other hand, in order to start up the system, we can use service registries where service providers are listed according to their service offerings (e.g., e-Bay's database of sellers). After listing service providers, the agents may select suitable service providers using the existing legacy systems (e.g., e-Bay's reputation system) and record their experiences with the selected providers as proposed in this dissertation. After enough number of experiences is accumulated in the system, the agents can start using the proposed system completely.

7.3. Future Work

In Chapter 3, we have assumed that agents exchange their experiences or context-aware ratings willingly. However, in open settings, there can be times when the agents do not prefer to cooperate with other agents. This could stem from two facts: (1) The users of the agents may not want to record their experiences as needed by the system and (2) the users may not be willing to exchange this information. Hence, incentives must be created for users to record and exchange their experiences. Incentive creation is an interesting problem that has received attention in the literature [94, 95]. Such techniques can complement our work to create incentives for exchanging experiences. Moreover, capturing complex business interactions may result in better representations of contracts in our simulations. This will enable us to benefit from the ontologies even more.

In Chapter 4, we have evaluated the performance of our approach in detecting the possible liars being fairly consistent in lying. For future work, in our evaluations, it would be worthwhile to explore the case where some liars lie only in a specific context while being honest in other contexts. It would also be worthwhile to consider other types of liars from the literature, such as *Exaggerated Positive* and *Exaggerated Negative* defined in [55]. The performance of detecting these types of liars would then be evaluated and compared against

competing approaches.

In Chapter 6, we propose a method for consumers to choose most suitable service selection mechanism among alternatives. We consider this as an important research problem, because many service selection approaches are proposed in the literature and many others will be proposed in the future. Hence, it becomes very important to choose the most proper one among alternatives with respect to the trade-offs of the consumers. As a future work, we plan to extend our approach to enable online addition of new service selection mechanisms to the learning algorithm. We also want to add our work a social dimension so that the consumers can recommend each other new service selection mechanisms. Then, the recommended mechanisms can be added to the learning algorithm on the fly. These are interesting directions for further research.

REFERENCES

1. Şensoy, M. and P. Yolum, "Ontology-Based Service Representation and Selection", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 8, pp. 1102–1115, 2007.

2. Şensoy, M., F. C. Pembe, H. Zırtıloğlu, P. Yolum, and A. Bener, "Experience-based service provider selection in agent-mediated E-Commerce", *Engineering Applications of Artificial Intelligence*, Vol. 20, No. 3, pp. 325–335, 2007.

3. Şensoy, M. and P. Yolum, "A Context-Aware Approach for Service Selection Using Ontologies", *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 931–938, 2006.

4. Şensoy, M. and P. Yolum, "A Comparative Study of Reasoning Techniques for Service Selection", *Proceedings of the Fifth International Workshop on Agents and Peer-to-Peer Computing (AP2PC)*, pp. 91–102, 2006.

5. Şensoy, M., "A Framework For Ontology-Based Service Selection in Dynamic Environments", *Proceedings of the* 22^{nd} *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1947–1948, 2007.

6. Şensoy, M. and P. Yolum, "Experimental Evaluation of Deceptive Information Filtering In Context-Aware Service Selection", *Proceedings of the* 11^{th} *International Workshop on Trust in Agent Societies*, pp. 153–165, 2008.

7. Şensoy, M. and P. Yolum, "A Cooperation-Based Approach For Evolution Of Service Ontologies", *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 837–844, 2008.

8. Şensoy, M. and P. Yolum, "Active Concept Learning for Ontology Evolution", *Proceedings of the* 18^{th} *European Conference on Artificial Intelligence (ECAI)*, 2008, To

Appear.

9. Şensoy, M. and P. Yolum, "Cooperative Evolution of Service Ontologies", *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1301–1303, 2007.

10. Şensoy, M. and P. Yolum, "On Choosing An Efficient Service Selection Mechanism In Dynamic Environments", *Proceedings of the Ninth International Workshop on Agent-Mediated Electronic Commerce (AMEC IX)*, pp. 99–112, 2007.

11. Nwana, H. S., J. Rosenschein, T. Sandholm, C. Sierra, P. Maes, and R. Guttmann, "Agent-mediated electronic commerce: issues, challenges and some viewpoints", *Proceedings of the Second International Conference on Autonomous agents (AGENTS)*, pp. 189–196, ACM Press, New York, NY, USA, 1998.

12. Ramaswamy, L., B. Gedik, and L. Liu, "A Distributed Approach to Node Clustering in Decentralized Peer-to-Peer Networks", *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, Vol. 16, No. 9, pp. 1–16, 2005.

13. Zhong, N., J. Liu, and Y. Y. Yao, "In Search of the Wisdom Web", *IEEE Computer*, Vol. 35, No. 11, pp. 27–31, November 2002.

14. Yolum, P. and M. P. Singh, "Engineering Self-Organizing Referral Networks for Trustworthy Service Selection", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. A35, No. 3, pp. 396–407, 2005.

15. Maximilien, M. and M. P. Singh, "A Framework and Ontology for Dynamic Web Services Selection", *IEEE Internet Computing*, Vol. 8, No. 5, pp. 84–93, 2004.

16. Sabater, J. and C. Sierra, "Reputation and Social Network Analysis in Multi-Agent Systems", *Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 475–482, 2002.

17. Yu, B. and M. P. Singh, "Emergence of Agent-based Referral Networks", *Proceedings of*

the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 1268–1269, 2002.

18. Huynh, T. D., N. R. Jennings, and N. Shadbolt, "FIRE: An Integrated Trust and Reputation Model for Open Multi-agent Systems.", *Proceedings of* 16^{th} *European Conference on Artificial Intelligence*, pp. 18–22, 2004.

19. Castelfranchi, C. and R. Falcone, "Principles of trust for MAS: cognitive anatomy,social importance, and quantification", *Proceedings of the Third International Conference on Multiagent Systems*, pp. 72–79, 1998.

20. Jøsang, A., R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision", *Decision Support Systems*, Vol. 43, No. 2, pp. 618–644, 2007.

21. Feigenbaum, L., I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens, "The Semantic Web in Action", *Scientific American*, Vol. 297, No. 6, pp. 90–97, 2007.

22. Pan, J. Z., "A Flexible Ontology Reasoning Architecture for the Semantic Web", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 2, pp. 246–260, 2007.

23. Gruber, T. R., "Toward principles for the design of ontologies used for knowledge sharing", *International Journal of Human-Computer Studies*, Vol. 43, No. 5-6, pp. 907–928, 1995.

24. Staab, S. and R. Studer (editors), *Handbook on Ontologies*, International Handbooks on Information Systems, Springer, 2004.

25. McGuinness, D. L. and F. V. Harmelen, "OWL Web Ontology Language Overview", http://www.w3.org/TR/owl-features, 2003.

26. Berners-Lee, T., J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American.Com*, 2001.

27. Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (editors), *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2003.

28. Horrocks, I., P. Patel-Schneider, and F. V. Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language", *Journal of Web Semantics*, Vol. 1, No. 1, pp. 7–26, 2003.

29. Hendler, J. and D. L. McGuinness, "DARPA Agent Markup Language", *IEEE Intelligent Systems*, Vol. 15, No. 6, pp. 72–73, 2001.

30. Harmelen, F. V., P. F. Patel-Schneider, and I. Horrocks, "Reference description of the DAML+OIL ontology markup language", http://www.daml.org/2001/03/reference, 2001.

31. Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "A Semantic Web Rule Language Combining OWL and RuleML", http://www.w3.org/Submission/SWRL, 2004.

32. Lloyd, J. W. (editor), *Foundations of logic programming*, Springer series in symbolic computation. Springer-Verlag, New York, 1987.

33. Steinmetz, R. and K. Wehrle (editors), *Peer-to-Peer Systems and Applications*, Springer Verlag, 2005.

34. Adamic, L., B. Huberman, R. Lukose, and A. Puniyani, "Search in power law networks", *Physical Review*, Vol. 64, 2001.

35. Unger, H. and M. Wulff, "Cluster-building in P2P-Community Networks", *Proceedings of Parallel and Distributed Computing and Systems (PDCS)*, pp. 680–685, 2002.

36. Rohrs, C., "Query routing for the Gnutella network", http://rfcgnutella.sourceforge.net, 2001.

37. Crespo, A. and H. Garcia-Molina, “Routing indices for peer-to-peer systems”, *Proceedings of 22th International Conference on Distributed Computing Systems (ICDCS)*, pp. 23–33, 2002.

38. Tsoumakos, D. and N. Roussopoulos, “Adaptive probabilistic search for peer-to-peer networks”, *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P)*, pp. 102–110, 2003.

39. Ripeanu, M. and I. Foster, “Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems”, *Lecture Notes in Computer Science*, Vol. 2429, pp. 85–93, 2002.

40. Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content Addressable Network”, *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 161–172, 2001.

41. Stoica, I., R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications”, *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 149–160, 2001.

42. Zhang, H., W. B. Croft, B. Levine, and V. Lesser, “A Multi-agent Approach for Peer-to-Peer based Information Retrieval System”, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 456–463, 2004.

43. McKnight, D. and N. Chervany, “The Meanings of Trust”, Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Reseach Center, 1996.

44. Resnick, P. and R. Zeckhauser, “Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay’s Reputation System”, Baye, M. (editor), *Advances in Applied Microeconomics: The Economics of the Internet and E-Commerce*, Vol. 11, Elsevier Science, 2002.

45. Abdul-Rahman, A. and S. Hailes, “Supporting trust in virtual communities”, *Proceedings of the Hawaii International Conference on Systems Sciences*, pp. 1–9, 2000.

46. Sabater, J. and C. Sierra, “REGRET: A reputation model for gregarious societies”, *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS)*, pp. 61–69, 2001.

47. Sabater, J. and C. Sierra, “Social ReGreT, a reputation model based on social relations”, *ACM SIGecom Exchanges*, Vol. 3, No. 1, pp. 44–56, 2002.

48. Page, L., S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web”, Technical report, Stanford Digital Library Technologies Project, 1998.

49. Yolum, P. and M. P. Singh, “Emergent Properties of Referral Systems”, *Proceedings of the Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 592–597, 2003.

50. Whitby, A., A. Jøsang, and J. Indulska, “Filtering out unfair ratings in bayesian reputation systems”, *The ICFAIN Journal of Management Research*, Vol. 4, No. 2, pp. 48–64, 2005.

51. Buchegger, S. and J. Y. Le Boudec, “A Robust Reputation System for P2P and Mobile Ad-hoc Networks”, *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

52. Teacy, W., J. Patel, N. Jennings, and M. Luck, “TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources”, *Autonomous Agents and Multi-Agent Systems*, Vol. 12, No. 2, pp. 183–198, March 2006.

53. Yu, B. and M. P. Singh, “Towards a probabilistic model of distributed reputation management”, *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, pp. 125–137, 2001.

54. Yu, B. and M. P. Singh, "An evidential model of distributed reputation management", *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 294–301, 2002.

55. Yu, B. and M. Singh, "Detecting Deception in Reputation Management", *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 73–80, 2003.

56. Fensel, D., J. Hendler, H. Lieberman, and W. Wahlster, *Spinning the Semantic Web*, The MIT Press, Cambridge, MA, 2003.

57. Hirtle, D., H. Boley, B. Grosof, M. Kifer, M. Sintek, S. Tabet, and G. Wagner, "Schema Specification of RuleML 0.91", http://www.ruleml.org/0.91, 2006.

58. Montgomery, D. C., *Design and analysis of experiments*, John Wiley and Sons, West Sussex, 2001.

59. Singh, M. P., "An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts", *Artificial Intelligence and Law*, Vol. 7, pp. 97–113, 1999.

60. Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner", *Web Semantics*, Vol. 5, No. 2, pp. 51–53, 2007.

61. Mui, L., M. Mohtashemi, and A. Halberstadt, "A Computational Model of Trust and Reputation", *Proceedings of the* 35^{th} *Hawaii International Conference on System Science (HICSS)*, pp. 2431–2439, 2002.

62. Duda, R. O., P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley and Sons, West Sussex, 2001.

63. Aamodt, A. and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *Artificial Intelligence Communications*, Vol. 7, No. 1, pp. 39–59, 1994.

64. Chinnici, R., J. J. Moreau, A. Ryman, and S. Weerawarana, “Web Services Description Language (WSDL) Version 2.0”, http://www.w3.org/TR/wsdl20, 2006.

65. Clement, L., A. Hately, C. V. Riegen, and T. Rogers, “UDDI Version 3.0.2”, http://uddi.org/pubs/uddi_v3.htm, 2004.

66. Sen, S. and N. Sajja, “Robustness of Reputation-Based Trust: Boolean Case”, *Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 288–293, 2002.

67. Soh, L.-K. and C. Chen, “Balancing Ontological and Operational Factors in Refining Multiagent Neighborhoods”, *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 745–752, 2005.

68. Aguzzoli, S., P. Avesani, and P. Massa, “Collaborative Case-Based Recommender Systems”, *Proceedings of the Sixth European Conference on Advances in Case-Based Reasoning: Lecture Notes in Computer Science*, Vol. 2416, pp. 460–474, Springer-Verlag, 2002.

69. Burke, R., “A Case-Based Reasoning Approach to Collaborative Filtering”, *Advances in Case-Based Reasoning*, pp. 370–379, Springer Verlag, 2000.

70. Limthanmaphon, B. and Y. Zhang, “Web Service Composition with Case-Based Reasoning”, *Australasian Database Conference*, pp. 201–208, 2003.

71. Cabral, L. and A. Hortacsu, “The Dynamics of Seller Reputation: Theory and Evidence from eBay”, Working Paper 10363, National Bureau of Economic Research, March 2004.

72. Lucking-Reiley, D. H., “Auctions on the Internet: What’s being auctioned, and how?”, *Journal of Industrial Economics*, Vol. 48, No. 3, pp. 227–252, 2000.

73. Dellarocas, C., “Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior”, *Proceedings of the Second ACM conference on Electronic*

commerce, pp. 150–157, 2000.

74. Zacharia, G., A. Moukas, and P. Maes, "Collaborative Reputation Mechanisms in Electronic Marketplaces", *Proceedings of the* 32^{nd} *Hawaii International Conference on System Sciences (HICSS-32)*, pp. 8026–8032, Maui, Hawaii, January 1999.

75. Russell, S. and P. Norvig, *Artificial Intelligence: A Modern Approach*, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 2002.

76. Jøsang, A. and R. Ismail, "The Beta Reputation System", *Proceedings of the* 15^{th} *Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy*, pp. 48–64, June 2002.

77. Zhang, J. and R. Cohen, "A Personalized Approach to Address Unfair Ratings in Multiagent Reputation Systems", *Proceedings of the* 10^{th} *International Workshop on Trust in Agent Societies*, 2006.

78. Zhang, J. and R. Cohen, "A Comprehensive Approach For Sharing Semantic Web Trust Ratings", *Computational Intelligence journal*, Vol. 23, No. 3, pp. 302–319, 2007.

79. Lin, D., "An Information-theoretic Definition of Similarity", *Proceedings of the* 15^{th} *International Conference on Machine Learning*, pp. 296–304, Morgan Kaufmann, San Francisco, CA, 1998.

80. Tversky, A., "Features of Similarity", *Psychological Review*, Vol. 84, No. 4, pp. 327–352, 1977.

81. Sen, S. and P. Kar, "Sharing a concept", *Working Notes of the AAAI-02 Spring Symposium*, pp. 55–60, 2002.

82. Afsharchi, M., B. Far, and J. Denzinger, "Ontology Guided Learning to Improve Communication among Groups of Agents", *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 923–930, 2006.

83. Doan, A., J. Madhaven, R. Dhamankar, P. Domingos, and A. Helevy, “Learning to match ontologies on the semantic web”, *VLDB Journal*, pp. 303–319, 2003.

84. Stephens, L. M., A. K. Gangam, and M. N. Huhns, “Constructing Consensus Ontologies for the Semantic Web: A Conceptual Approach”, *World Wide Web*, Vol. 7, No. 4, pp. 421–442, 2004.

85. Williams, A. B., “Learning to Share Meaning in a Multi-Agent System”, *Autonomous Agents and Multi-Agent Systems,* Vol. 8, No. 2, pp. 165–193, 2004.

86. Aberer, K., P. Cudre-Mauroux, and M. Hauswirth, “Start making sense: The Chatty Web approach for global semantic agreements”, *Journal of Web Semantics*, Vol. 1, No. 1, pp. 89–114, 2003.

87. Steels, L., “The Origins of Ontologies and Communication Conventions in Multi-Agent Systems”, *Autonomous Agents and Multi-Agent Systems,* Vol. 1, No. 2, pp. 169–194, October 1998.

88. Laera, L., I. Blacoe, V. Tamma, T. Payne, J. Euzenat, and T. Bench-Capon, “Argumentation over Ontology Correspondences in MAS”, *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1285–1292, 2007.

89. Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

90. Tesauro, G., “Temporal difference learning and TD-Gammon”, *Communications of the ACM*, Vol. 38, No. 3, pp. 58–68, 1995.

91. Watkins, C. J. C. H. and P. Dayan, “Q-learning”, *Machine Learning,* Vol. 8, No. 3, pp. 279–292, 1992.

92. Whiteson, S., M. E. Taylor, and P. Stone, “Empirical Studies in Action Selection for Reinforcement Learning”, *Adaptive Behavior*, Vol. 15, No. 1, pp. 33–50, 2007.

93. Ortega, J., M. Koppel, and S. Argamon, "Arbitrating Among Competing Classifiers Using Learned Referees", *Knowledge and Information Systems*, Vol. 3, No. 4, pp. 470–490, 2001.

94. Jurca, R. and B. Faltings, "An incentive compatible reputation mechanism", *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1026–1027, 2003.

95. Zhang, J. and R. Cohen, "Design of a Mechanism for Promoting Honesty in E-Marketplaces", *Proceedings of the* 22^{nd} *Conference on Artificial Intelligence (AAAI)*, pp. 1495–1500, 2007.