Ontology-Based Service Representation and Selection
Murat Şensoy and Pınar Yolum

Abstract—Selecting the right parties to interact with is a fundamental problem in open and dynamic environments. The problem is amplified when the number of interacting parties is high, and the parties’ reasons for selecting others vary. 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. Previous approaches to service selection are usually based on capturing and exchanging the ratings of consumers to providers. Rating-based approaches have two major weaknesses. 1) ratings are given in a particular context. Even though the context is crucial for interpreting the ratings correctly, the rating-based approaches do not provide the means to represent the context explicitly. 2) The satisfaction criteria of the rater is unknown. Without knowing the expectation of the rater, it is almost impossible to make sense of a rating. We deal 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 ratings with detailed experiences of consumers. The experiences are represented with an ontology that can capture the requested service and the received service in detail. When a service consumer decides to share her experiences with a second service consumer, the receiving consumer evaluates the experience by using her 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.

Index Terms—Multiagent systems, ontology design, electronic commerce.

1 INTRODUCTION

Most interesting applications on the Web rely on the participating entities to interact. However, selecting with whom to interact is difficult in open environments such as the Web, mostly because open systems are not operated by a central authority that can monitor all participants’ activities and ensure that everyone acts in the best interest of others [3], [4]. Consider a consumer interested in receiving a service. The openness of the Web implies that for a given service description, a plethora of service providers with substantially different service offerings may exist. How can a consumer learn about the providers’ past dealings, combine information from multiple sources, and finally decide on one service provider?

Service selection has been widely studied 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. 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 [5]. E-bay [6] is a well-known Web site that uses a reputation system. Whereas in closed settings, finding a central authority (such as the company itself) is easier because there is no such authority for open systems. Hence, the traditional reputation systems are not directly applicable in open systems [7].

Since the 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 [8], [7], [9]. Trust captures a trustee’s expectation from a trustee for a particular service. Whereas different formalizations of trust exist, most formalizations are not expressive enough. That is, typically, a trustee’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 [10]. For example, a service consumer may give a low rating to a service provider who delivers a book two days late. If the delivery date is not significant for a second service consumer, then the first service consumer’s low rating will not be significant either. Hence, it is important that ratings are evaluated within their scope. The scope of a rating is the context in which the rater has experienced the service. For different service interests, the same service consumer can give different ratings for the same service. Similarly, for exactly the same service interest and the same service supplied for this service interest, different consumers may give marginally different ratings, depending on their satisfaction criteria. This takes place because ratings reflect the satisfaction criteria and taste of the raters.

Accordingly, this paper develops two novel approaches that allow 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 by using the objective data in the experience.

The rest of this paper is organized as follows: Section 2 explains the proposed rating-based approach for the context-aware service selection. Section 3 experimentally evaluates the rating-based approach. Section 4 explains the proposed experience-based approach for the context-aware service selection in detail and proposes two different decision-making schemes. Section 5 presents the experimental evaluation of the experience-based approach. Section 6 summarizes our contribution, discusses our results, and compares our work to related work in the literature.

2 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, then 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 (for example, with a plain rating), then 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, she will not choose to interact with the service provider. Instead, if the latter consumer recognizes the scope of the rating, then she 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 [11]. This ontology covers the domain-level knowledge and fundamental concepts such as service demand, service provider, and rating.

2.1 Context Ontology

Fig. 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 her evaluation of what is received in 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 a Demand class in the ontology. The properties of the 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 by using these domain-level properties. For example, the hasPrice property is used to represent the money that a consumer is willing to pay for a service. The range of the hasShoppingItem property is the ShoppingItem class. This class has the properties hasQuantity, hasUnitPrice, and hasQuality. The range of the hasQuality property is the Quality class. This class describes
2.2 Selecting Service Providers

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

After collecting a 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 her current service demand. Similarity metrics and computation of similarity depends on the consumer. Then, the set of context-aware ratings is 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. The following shows the computation of the aggregated rating $AR_i$ for the provider $i$:

$$AR_i = \sum_j \frac{r_{ij} \times w_j \times R_j}{\sum_t r_{it} \times w_t},$$

where $j$ refers to a context-aware rating, $w_j$ is the weight of $j$, $R_j$ refers to the rating within $j$, and $r_{ij}$ has a value of 1 only if $i$ is the service provider within $j$; otherwise, its value is 0. After computing the aggregated ratings, the consumer chooses the provider with the highest aggregated rating.

3 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 is used as the OWL-DL reasoner [14] because in addition to OWL predicates, KAON2 supports some useful non-OWL predicates such as ifTrue and evaluate to test and evaluate logical and mathematical expressions. Simulations are run on an IBM server with a 2.8-GHz CPU and a 4.0-Gbyte RAM under a Windows 2003 server OS. Simulations are repeated 10 times in order to increase the reliability.

The main purpose of these simulations is 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 explained as follows:

- **Service provider selection using context-aware ratings ($SPS_{CAR}$).** This strategy is proposed in Section 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 who have had similar demands with respect to the context.
Given the service constraints, the simulation environment generates the demand of the service consumer. To do so, the demand space is constructed by removing the dimensions of the service space that do not belong to the Demand class. Then, a random region in this demand 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 the demand region, then the service consumer gets satisfied; otherwise, she 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 is classified as a similar demand by the consumer.

The simulation environment is set up with 20 service providers and 400 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, the agents gain experiences and collect context-aware ratings. There are several important factors in the simulations:

- **Variations in service demand.** As noted above, 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 percent of the consumers consume a service at a given epoch.

- **Variations in service satisfaction.** 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 misleading similarity factor \( (\beta) \) defines what ratio of the service consumers having similar service demands with respect to similarity criteria of \( X \) will have a satisfaction criteria conflicting with the satisfaction criteria of \( X \). Thus, the ratings of these consumers will probably mislead consumer \( X \) during service selection.

### 3.1 Simulation Environment Factors

In our simulations, service characteristics of a service provider are generated as the following. First, a service space is defined so that all possible services are represented within this service space. The dimensions of the service space and their ranges are tabulated in Table 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, then the service that the provider delivers will be constructed as follows: The properties that are specified (shopping item, quantity, and location) will be fixed. For the remaining attributes, the service provider will choose random values, making sure that the values stay in the range of its service region. Hence, for this example, the degree of freedom for generating the services will be reduced to 12.

<table>
<thead>
<tr>
<th>Dimension Name</th>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasShoppingItem</td>
<td>Integer</td>
<td>1 - 1000</td>
</tr>
<tr>
<td>toLocation</td>
<td>Integer</td>
<td>1 - 100</td>
</tr>
<tr>
<td>hasDeliveryType</td>
<td>Integer</td>
<td>1 - 6</td>
</tr>
<tr>
<td>hasDeliveryDuration</td>
<td>Integer</td>
<td>1 - 60</td>
</tr>
<tr>
<td>hasShipmentCost</td>
<td>Double</td>
<td>0 - 250</td>
</tr>
<tr>
<td>hasPrice</td>
<td>Double</td>
<td>10 - 11000</td>
</tr>
<tr>
<td>hasUnitPrice</td>
<td>Double</td>
<td>1 - 100</td>
</tr>
<tr>
<td>hasQuantity</td>
<td>Integer</td>
<td>1 - 100</td>
</tr>
<tr>
<td>hasQuality</td>
<td>Integer</td>
<td>1 - 10</td>
</tr>
<tr>
<td>isRefundable</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
<tr>
<td>hasConsumerSupport</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
<tr>
<td>didReceiveMerchandise</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
<tr>
<td>hasStockInconsistency</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
<tr>
<td>isAsDescribed</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
<tr>
<td>isDamaged</td>
<td>Boolean</td>
<td>0 - 1</td>
</tr>
</tbody>
</table>

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 the real world, service consumers periodically change their service demands. This is done to mimic variations on context.

### 3.2 Simulation Results

This section summarizes the simulation results for the \( SPSCAR \). Fig. 3 shows the simulation results for the parameters \( P_{CD} = 0.2 \) and \( \beta = 0 \). This means that a consumer will change its service demand with a probability of 0.2 after receiving a service. Fig. 3 indicates that the performance of \( SPSSR \) decreases sharply with time, whereas the performance of \( SPSCAR \) is constant around 100 percent of satisfaction because the 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 will be more misleading than before. However, SPS\textsubscript{CAR} can differentiate ratings for different contexts. Hence, as seen in Fig. 3, SPS\textsubscript{CAR} leads to decisions with a 100 percent satisfaction, but the satisfaction ratio of SPS\textsubscript{SR} decreases and approaches to that of SPS\textsubscript{RR} as consumers change their demands. The satisfaction ratio is low and constant around 0.2 for SPS\textsubscript{RR}. This means that only 20 percent of the service decisions result in satisfaction for consumers using this strategy.

Fig. 4 shows the simulation results for the settings: 20 service providers, 400 service consumers, \( P_{CD} = 0 \), and \( \beta = 0.5 \). This is the case when the consumers do not change their service demands, and approximately half of the consumers owning similar contexts with respect to the similarity criteria of a consumer \( X \) will have satisfaction criteria conflicting with the satisfaction criteria of \( X \). For this setting, SPS\textsubscript{CAR} is slightly better than SPS\textsubscript{SR}. Similarly, the performance of SPS\textsubscript{SS} does not decrease over time because consumers using SPS\textsubscript{SS} do not change their service demands over time (\( P_{CD} = 0 \)). As expected, approximately 50 percent of the selected services lead to the satisfaction of service consumers both for SPS\textsubscript{SR} and SPS\textsubscript{CAR} because approximately half of the ratings in the contexts will reflect the taste of the consumer making the service decision (\( \beta = 0.5 \)). Therefore, the consumer will make a wrong decision half of the time. For these settings, the satisfaction ratio of SPS\textsubscript{SR} is also halved and constant around 0.1. Because \( \beta \) is equal to 0.5, the probability of finding a consumer with the same taste decreases to half in our simulations.

Unlike SPS\textsubscript{SS}, SPS\textsubscript{CAR} is robust to \( P_{CD} \). That is, its performance is insensitive to changes in \( P_{CD} \). On the other hand, both SPS\textsubscript{SR} and SPS\textsubscript{CAR} are seriously affected by \( \beta \). Fig. 5 shows the simulation results for the parameters \( P_{CD} = 0.2 \) and \( \beta = 0.5 \). The performance of SPS\textsubscript{CAR} is similar in both Figs. 4 and 5 because in both cases, \( \beta \) is set to 0.5. However, the performance of SPS\textsubscript{SR} is lower than that of SPS\textsubscript{CAR} and continuously decreases over time because \( P_{CD} \) is set to 0.2.

For the case where \( \beta = 0 \) and \( P_{CD} = 0 \), the satisfaction ratio of SPS\textsubscript{CAR} and SPS\textsubscript{SS} approaches 1. On the other hand, the ratio of decisions that resulted in satisfaction is constant around 0.2 for SPS\textsubscript{SS}. In this setting, consumers never change their service demands, and if a consumer classifies a context retrieved from another consumer as a similar context, then any provider satisfying the latter will also satisfy the former. Hence, ratings taken from consumers having a similar context will always guide to the satisfaction.

**4 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, the ratings will reflect the subjective opinion of the raters. In order to minimize the subjectiveness of the rating-based approaches, we propose to use an objective experience-based approach. In this approach, the experience of a consumer with a provider is represented using ontologies. This representation contains the requested service and the supplied service in detail.
Hence, by using the experiences, other consumers can evaluate the supplied services with respect to their own evaluation criteria.

In order to represent the experiences, the ontology in Fig. 1 is extended properly. 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 those concepts and properties, a service consumer can express the details of its transactions with different service providers.

4.1 Base-Level Ontology

The base-level ontology (Fig. 6) 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 the service consumers in the system. As in 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 in the end. To conceptualize the service demand and the received service of the consumer, the 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 the Description class can be used to describe the service demands, supplied services, responsibilities, and fulfillments of sides during transactions. These properties are shown in the domain-level ontology.

Each Description class contains an owner and a date field that are themselves represented as classes. 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 the demanded service or the provided service.

An owner may have commitments toward others to carry out responsibilities [15]. 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 1 demonstrates a simple responsibility instance. The Commitment and Responsibility classes are used to express commitments and responsibilities, respectively, in the experience ontology. Fulfillments are the accomplishments of responsibilities and are denoted with the Fulfillment class. Owners of responsibilities or fulfillments can be the service consumers or the providers, depending on the context.

Example 1. Consider a service provider who 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 the toLocation property referring to New York City and has the 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, the conditional commitments have preconditions. For example, a conditional commitment $CC(X, Y; P; Q)$ denotes that if the precondition $P$ is carried out by $Y$, then $X$ will be committed to carry out responsibility $Q$. In this definition, $Y$ is the owner of the precondition, and $X$ is the owner of the responsibility. The ConditionalCommitment and Precondition classes are used in the ontology to specify the conditional commitments and preconditions. The conditional commitments can be used to represent contracts and offers made by the service consumers and providers. An example case is demonstrated in Example 2.

Example 2. A service consumer can offer to pay an additional $100 for a one-week early delivery. If the provider makes the shipment one week early, then the consumer is committed to pay $1,100 for a product whose actual value is only $1,000. Service providers can also make offers by using conditional commitments.

Fig. 6. Base-level ontology.
4.2 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 the domain-level ontology is the Description class. The domain-specific properties of the Description class are used to describe the service demands, supplied services, responsibilities, and fulfillments of parties during transactions. A domain-level ontology for online shopping is shown in Fig. 7.

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 2.1. However, the properties of the Description class have slightly different meanings for different subclasses of the Description class. In addition to the properties of the Description class, concepts in the ontology may also have domain-specific properties that other concepts do not have. For example, for the consumer goods domain, Boolean data type properties such as didReceiveMerchandise, hasStockInconsistency, isAsDescribed, and isDamaged are included as the properties of the Service class in the domain-level ontology.

4.3 Exchanging Experiences

A consumer society emerges as a result of the consumers’ need to retrieve the experiences of other consumers. Initially, each service consumer knows only a subset of all the consumers in the society and lists these consumers in an acquaintance list. An acquaintance list is a dynamic list of service consumers having service demands classified as a 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 in Algorithm 1.

Algorithm 1

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

In this algorithm, when a service consumer decides to receive a service, it checks its 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 percent confidence interval [16] (line 3). If the number of experiences in the repository is not enough to perform this computation, then the service consumer collects new experiences (lines 6-25). However, in order to collect new experiences, the consumer should have a 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 enough to perform this computation, then the service consumer collects new experiences (lines 6-25). However, in order to collect new experiences, the consumer should have a 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 a sufficient number of acquaintances, then it should increase the number of its acquaintances (lines 10-19).

For this purpose, we 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). Both PDM and RAM contain an SWRL rule that expresses the similar demand criteria of the originator of the message (lines 6, 10, and 15). When a consumer Y receives a PDM, it checks if its service demands are similar to that of the originator X. If so, then it notifies X, and X adds Y as a new acquaintance entry in its acquaintance list. This entry contains the 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 the consumers having demands similar to the demand of the originator to forward the request. If there is no such consumer, then 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 agents that receive the request act the same way Y does. When Y receives a RAM 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. Thus, X can add these entries to its acquaintance list.

The service consumer populates it acquaintance list through PDMs and RAMs (lines 10-12 and lines 15-17). After having a sufficient number of acquaintances, the consumer uses REM to collect new experiences (lines 21-24). A REM also contains a rule for expressing the similar demand criteria of the sender. When service consumer Y gets a REM from service consumer X, it evaluates its demands in its experiences by using the similarity criteria in the REM. Later, it can send its experiences to X if the experiences have similar demands with respect to the 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). The next section describes this procedure.

The protocol that is summarized above is also used by our rating-based approach in Section 2 to gather context-aware ratings. The only difference is that REMs are used to request context-aware ratings instead of experiences.

4.4 Service Selection Using Experiences

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 nonparametric 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.

4.4.1 Service Selection 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 GM for each service provider. There are two classes for each model: satisfied and dissatisfied.

The demand and service specifications within the experiences are received in the form of ontologies, but then, they are converted into an internal representation of the service consumer. The demand and commitment information in each experience is represented as a vector. Each field in this vector is extracted from the experience ontology such as service price. Then, the supplied service for this demand is classified as satisfied or dissatisfied with respect to the satisfaction criteria of the consumer. After that, the (vector, class) pairs are used as the training set.

For each class, the covariance and the mean are extracted from the training set. Then, for each of the classes, a discriminant function is defined to compute the probability of satisfaction [17]. 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, the mean and the covariance are represented by $\mu_i$ and $\Sigma_i$, respectively. Equation (2) formulates the class likelihood $p(X|C_i)$, and the probability that the demand $X$ is observed in class $C_i$. Equation (3) formulates the posterior probability $p(C_i|X)$, and the probability that the demand $X$ is observed in class $C_i$. In (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. Last, the discriminant function for the $i$th class $g_i(X)$ is formulated as in (4):

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

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

$$g_i(X) = \log[p(C_i|X)] + \log[p(C_i)].$$
4.4.2 Service Selection 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 [18]. The intuition is that if two problems are similar, then 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 ill-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 by using an OWL ontology and SWRL rules. Using these well-defined similarity metrics, the consumer queries the consumer society for similar experiences by using the procedure explained in Section 4.3. After retrieving the similar experiences, the consumer computes a score for each retrieved experience. The computation depends on the following factors:

- **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.
- **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.
- **Satisfaction.** This is an important factor that measures how satisfied the current consumer agent would be if 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 by using

$$ S_i = recency_i \times sim_i \times sat_i, $$

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 and distributes the case base such that it can be searched with individually defined similarity metrics.

![Fig. 8. Simulation results for the settings: 20 service providers, 400 service consumers, $P_{CD} = 0$, and $\beta = 0.5$.](image)

5 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:

- **Service provider selection using GM ($SPS_{GM}$).** This strategy is proposed in Section 4.4.1.
- **Service provider selection using CBR ($SPS_{CBR}$).** This strategy is proposed in Section 4.4.2.

By varying the aforementioned factors (that is, 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 the services. The simulation environment is set up as explained in Section 3. Furthermore, the experiences are set to expire after 20 epochs to keep the experience repositories fresh and small.

5.1 Experience-Based Approach: GM

This section summarizes the results of the simulations for $SPS_{GM}$. Fig. 8 shows the simulation results for $P_{CD} = 0$ and $\beta = 0.5$. This value of $\beta$ implies that 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. As expected, only 50 percent of the services will lead to the satisfaction of the service consumers if these service consumers use $SPS_{SR}$. On the other hand, service consumers using $SPS_{GM}$ will be almost always satisfied with the supplied service.

Fig. 9 shows the simulation results for the parameters $P_{CD} = 0.2$ and $\beta = 0$. 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 a 100 percent satisfaction, but the satisfaction ratio of $SPS_{SR}$ decreases over time as consumers change their service demands. The simulation results of $SPS_{CBR}$ for the same settings (see Fig. 3) is similar.
to the results in Fig. 9. This means that performances of SPCSAR and SPSGM are similar for the settings $PCD = 0.2$ and $\beta = 0$.

Fig. 10 shows the simulation results for the parameters $PCD = 0.2$ and $\beta = 0.5$. The performance of SPSSR decreases further for these settings. However, for SPSGM, the satisfaction ratio is around 1 after the fifth epoch (before the fifth epoch, there are not enough experiences accumulated in the environment for the modeling of service providers).

In order to see the influences of $\beta$ and $PCD$ on the satisfaction ratios achieved by the SPSGM and SPSSR strategies, simulations are repeated for different $\beta$ and $PCD$ values. The average ratios of satisfactions for these simulations are shown in Tables 2 and 3. The performance of SPSSR decreases considerably with an increase in the value of $\beta$ or $PCD$. These parameters are independent of each other. Thus, the combinatorial effect of these parameters on the performance of SPSSR is the multiplication of influences of each parameter. Simulations show that the proposed method SPSGM is robust to changes in the $\beta$ and $PCD$ parameters. Unlike SPSSR, the performance of SPSGM does not change with changing $\beta$ and $PCD$ values, and the achieved satisfaction is 100 percent if the service consumers use SPSGM for decision making.

5.2 Experience-Based Approach: CBR

This section summarizes the results of the simulations for SPSCBR and compares its performance with that of SPSGM. Initially, there are two primary variables in the simulations: $PCD$ and $\beta$. We measure the average satisfaction ratio of the strategies when $\beta$ equals 0 and 0.5 and when $PCD$ varies from 0 to 1. In Table 4, we immediately note that for all values of $PCD$, both SPSGM and SPSCBR achieve a high average satisfaction ratio, whereas SPSSR achieves a fluctuating value (Table 3). Unlike the rating-based approaches such as SPSSR, both SPSGM and SPSCBR are robust to $\beta$ and $PCD$. Moreover, the performance of these two strategies is impressive and almost equivalent.

Although the performance of SPSGM and SPSCBR are almost the same, the time that they use to select the providers are different. Fig. 11 shows the time consumed by each approach for different numbers of experiences. For a small number of experiences, the time required by these approaches are similar. However, as the number of experiences increases, the time consumed by SPSGM exceeds the time consumed by SPSCBR dramatically. For the modeling of the consumers using GM, the size of the data set, namely, the number of experiences, is important. The size of the data set should be large enough to remove the bias [17]. However, increasing the number of experiences will increase the time required for the computations.

5.3 Additional Simulation Factors

So far, we have deliberately assumed that providers always provide the same quality of service consistently. However, some providers may have a 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 indeterminism on the provider side. This probability denotes how much providers deviate

<table>
<thead>
<tr>
<th>$P_{CD}$</th>
<th>$SPS_{GM}$</th>
<th>$SPS_{CBR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>0.1</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1.0</td>
<td>0.98</td>
<td>0.95</td>
</tr>
</tbody>
</table>

and the achieved satisfaction is 100 percent if the service consumers use SPSGM for decision making.
from their expected behavior. Consider a provider who 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, then this kind of indeterminism 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 indeterminism 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 the behaviors of the providers are predictable and free of indeterminism, $SPS_{CBR}$ 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, then the provider will probably satisfy the consumer’s current demand too. When $PI$ is set to 0, this assumption always holds. The 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, each provider deviates from its usual service offering in favor of the consumers with these probabilities.

Figs. 12 and 13 show the simulation results for $PI = 0.001$ and $PI = 0.01$, respectively. The first thing to notice 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 percent of satisfaction. On the other hand, the performance of $SPS_{CBR}$ further decreases with time because each time, new misleading experiences are added to the environment, since the number of misleading experiences increases with time.

6 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

<table>
<thead>
<tr>
<th>$P_{CD}$</th>
<th>$SPS_{GM}$</th>
<th>$SPS_{CBR}$</th>
<th>$SPS_{GM}$</th>
<th>$SPS_{CBR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = 0$</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>0.1</td>
<td>0.97</td>
<td>0.96</td>
<td>0.94</td>
<td>0.96</td>
</tr>
<tr>
<td>0.2</td>
<td>0.97</td>
<td>0.97</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>0.4</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>0.6</td>
<td>0.98</td>
<td>0.97</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>0.8</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>1.0</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
</tbody>
</table>

TABLE 4: Average Ratio of Satisfaction with Respect to Different $\beta$ and $P_{CD}$ Values
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 paper 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.

6.1 Real-Life Applications

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 the Web Services Description Language (WSDL) can be used [19].

In the protocol that we have used in Section 4.3, 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 [20]. The agents may initially look up the service providers from these registries and have interactions with these service providers. When enough experiences have been accumulated, 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, the service consumers may sign in to another directory called the consumer directory. They may provide their identity and Internet Protocol (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 peer-to-peer (P2P) systems. After locating some acquaintances, the system will continue functioning, as explained in Section 4.3.

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.

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

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 $SPS_{SR}$ and $SPS_{SSR}$. However, it is vulnerable to changes in the parameter $\beta$ because the ratings reflect the satisfaction criteria and the 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 the achieved ratio of satisfaction is the experience-based approach. When the service providers do not change the quality of their services, both $SPS_{GM}$ and $SPS_{CBR}$ perform equally well in finding the service providers (they achieve about 100 percent of 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 with a small percentage, then $SPS_{CBR}$ performs much worse than $SPS_{GM}$.

Although the experience-based approach is the best service selection approach in this paper, it requires the 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 the service consumers over the experience-based approach even though $SPS_{CAR}$ has some disadvantages with respect to the experience-based approach in terms of the achieved ratio of satisfaction.

6.3 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 (for example, in response to a 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 [9] is a trust and reputation model consisting of four components: interaction trust, role-based trust, witness reputation, and certified reputation. The 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, the 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 [21] 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 her needs.

Yolum and Singh study the properties of referral networks for service selection, where referrals are used among the service consumers to locate the service providers [7]. Current applications of referral networks also rely on exchanging ratings. Hence, they 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 a distributed information retrieval task [22]. In their work, each agent has a view of its environment called agent view. The agent-view structure of an agent contains information about the 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 the acquaintance list structure in our work. Zhang et al.’s protocol does not use ontologies or DL reasoners during information retrieval. However, if their protocol is modified to accommodate the experience ontology and DL reasoners, then their protocol can be used for retrieving experiences instead of the protocol that we have used in Section 4.3.

Soh and Chen propose a multiagent approach to improve distributed information retrieval performance by balancing ontological and operational factors [23]. 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, 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 quality-of-service (QoS) ontology to represent the quality levels of service agents and the preferences of the consumers [24]. 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 consumer preferences. Aguzzoli et al. propose a collaborative case-based recommendation system for the music market [25]. 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 [26]. This system is hosted by a Web site, 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 [27]. 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 that it includes and the relationships between these services. When a user has a new request for a composite service, the similarity measure is used to find the closest cases in the case base. As the 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.

In this work, we have assumed that agents exchange their experiences or context-aware ratings honestly and 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 [28]. 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.
ACKNOWLEDGMENTS
This research has been partially supported by Boğaziçi University Research Fund under grant BAP07A102 and the Scientific Research Council of Turkey by a CAREER Award under grant 105E073. This paper significantly extends a paper that appears in the Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems [1] and a paper that appears in the Proceedings of the Fifth International Workshop on Agents and Peer-to-Peer Computing [2].

REFERENCES

Murat Şensoy received the MSc degree in computer engineering from Boğaziçi University, Istanbul, in 2004, where he is a research assistant. During his MSc studies, he worked on the online reconfiguration of dynamic multicast trees for real-time communication. He is currently pursuing the PhD degree in computer engineering at Boğaziçi University under the guidance of Dr. Pınar Yolum. His current research is on context-aware service selection in agent-mediated e-commerce. His research interests include multiagent systems, semantic Web services, and peer-to-peer networks.

Pınar Yolum received the BSc degree in computer engineering from Marmara University, Istanbul, in 1998 and the MS and PhD degrees in computer science from North Carolina State University in 2003 and 2000, respectively. She is an assistant professor at Boğaziçi University, Istanbul. She worked as a postdoctoral researcher at the Free University of Amsterdam. Her research interests include multiagent systems, peer-to-peer networks, and service-oriented computing.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.