Grid Service Discovery with Rough Sets

Maozhen Li, *Member*, *IEEE*, Bin Yu, Omer Rana, and Zidong Wang, *Senior Member*, *IEEE*

Abstract—The computational grid is rapidly evolving into a service-oriented computing infrastructure that facilitates resource sharing and large-scale problem solving over the Internet. Service discovery becomes an issue of vital importance in utilizing grid facilities. This paper presents ROSSE, a Rough sets-based search engine for grid service discovery. Building on the Rough sets theory, ROSSE is novel in its capability to deal with the uncertainty of properties when matching services. In this way, ROSSE can discover the services that are most relevant to a service query from a functional point of view. Since functionally matched services may have distinct nonfunctional properties related to the quality of service (QoS), ROSSE introduces a QoS model to further filter matched services with their QoS values to maximize user satisfaction in service discovery. ROSSE is evaluated from the aspects of accuracy and efficiency in discovery of computing services.

Index Terms—Grid computing, Semantic Web, grid service discovery, QoS modeling, Rough sets.

1 Introduction

With the development of Web service technologies [1], the computational grid [2] is rapidly evolving into a service-oriented computing infrastructure that facilitates resource sharing and large-scale problem solving over the Internet [3]. The Open Grid Services Architecture (OGSA) [4], promoted by the Open Grid Forum (OGF, http://www.ogf.org) as a standard service-oriented architecture (SOA) for grid applications, has facilitated the evolution. It is expected that the Web Service Resource Framework (WSRF) [5] will be acting as an enabling technology to drive this evolution further. The promise of SOA is the enabling of loose coupling, robustness, scalability, extensibility, and interoperability for large-scale grid systems.

As shown in Fig. 1, various resources on the Internet including processors, disk storage, network links, instrumentation and visualization devices, domain applications, and software libraries can be exposed as OGSA/WSRF-based grid services, which are usually registered with a service registry. A service bus building on service-oriented grid middleware technologies such as Globus [6] enables the instantiation of grid services. A grid environment may host a large number of services. Therefore, service discovery becomes an issue of vital importance in utilizing grid facilities.

Grid services are implemented as software components, the interfaces of which are used to describe their functional and nonfunctional properties (attributes). Advertising services in a grid environment means that service-associated properties are registered with a service registry. Service discovery involves a matching process in which the properties of a service query are matched with that of a service advertisement.

In a grid environment, service publishers may advertise services independently using their predefined properties to describe services. Therefore, uncertainty of service properties exists when matching services. An uncertain property is defined as a service property that is explicitly used by one advertised service but does not appear in another service advertisement that belongs to the same service category. This can be further illustrated using Table 1. For example, property P_1, which is explicitly used by service S_1 in its advertisement, does not appear in the advertisement of service S_2. Similarly, property P_3, which is explicitly used by service S_2, does not appear in the advertisement of service S_1. When services S_1 and S_2 are matched with a service query using properties P_1, P_2, P_3, and P_4, property P_1 becomes an uncertain property in matching service S_2, and property P_3 becomes an uncertain property in matching service S_1. Consequently, both S_1 and S_2 may not be discovered because of the existence of uncertainty of properties even though the two services are relevant to the query.

It is worth noting that properties used in service advertisements may have dependencies, e.g., both P_1 and P_3 may be dependent properties of P_2 when describing services S_1 and S_2, respectively. Both S_1 and S_2 can be discovered if P_1 and P_3 (which are uncertain properties in terms of the user query) are dynamically identified and reduced in the matching process. To increase the accuracy of service discovery, a search engine should be able to deal with uncertainty of properties when matching services.

In this paper, we present ROSSE [21], [22], [23], [24]: a search engine for grid service discovery. Building on Rough sets theory [25], ROSSE is novel in its capability to deal with uncertainty of service properties when matching services. This is achieved by dynamically identifying and reducing dependent properties that may be uncertain properties

- *M. Li is with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK. E-mail: Maozhen.Li@brunel.ac.uk.*
- *B. Yu is with Level E Limited, ETTC, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK. E-mail: Bin.Yu@levelelimited.com.*
- *O. Rana is with the School of Computer Science, Cardiff University, Queen's Buildings, 5 The Parade, Roath, Cardiff, CF24 3 AA, UK. E-mail: O.F.Rana@cs.cardiff.ac.uk.*
- *Z. Wang is with the School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, UB8 3PH, UK. E-mail: Zidong.Wang@brunel.ac.uk.*

Manuscript received 10 Apr. 2007; revised 20 Sept. 2007; accepted 6 Dec. 2007; published online 19 Dec. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-04-0151.
Digital Object Identifier no. 10.1109/TKDE.2007.190744.

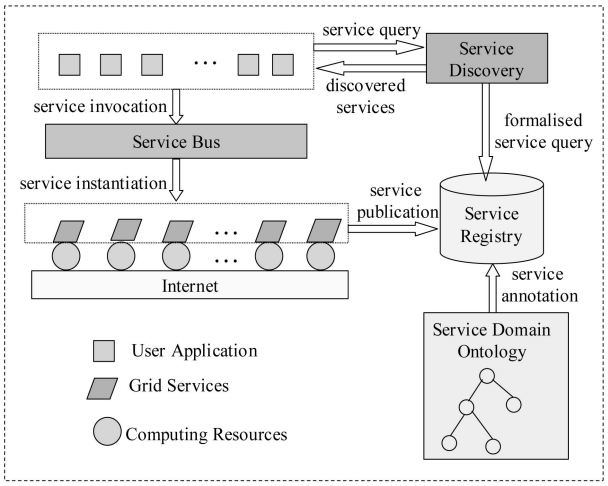

Fig. 1. A layered structure of service-oriented grid systems.

when matching a service query. In this way, ROSSE increases the accuracy in service discovery. In addition, functionally matched services may have distinct nonfunctional properties related to the quality of service (QoS). To maximize user satisfaction in service discovery, ROSSE introduces a QoS model to further filter matched services with their QoS values. Finally, ROSSE is evaluated from the aspects of accuracy and efficiency in discovery of computing services.

The remainder of the paper is organized as follows: Section 2 presents the design of ROSSE with a focus on dependent property reduction (DPR). Section 3 introduces a QoS model to filter matched services with their QoS values. Section 4 briefly describes the implementation of ROSSE and gives a case study to illustrate the application of ROSSE for discovery of computing services. Section 5 evaluates the accuracy and efficiency of ROSSE in service discovery. Section 6 discusses some related work, and Section 7 concludes the paper.

2 The Design of ROSSE

ROSSE considers input and output properties individually when matching services. For the simplicity of expression, input and output properties used in a service query are generally referred to as service properties. The same goes for service advertisements. Fig. 2 shows ROSSE components. The interactions between the components follow two processes—service publication and service discovery.

Service publication. Service publishers advertise their services to ROSSE through a Web user interface (step 1). Advertised services with WSDL interfaces or OWL-S [13] interfaces are then loaded into the ROSSE Service Repository, in which the elements of services such as the names and properties of services are registered with ROSSE (step 2). When advertising services, service publishers may also publish service ontologies that can be defined in OWL [14]. These OWL ontologies are then parsed by an OWL parser (step 3) and loaded into the ROSSE Ontology Repository (step 4). The ontology repository is used by an inference engine to infer the semantic relationships of properties when matching services.

TABLE 1
Two Service Advertisements with Uncertain Service Properties

advertised services	property	property	property	property
S_1	P_1	P_2		P_4
S_2		P_2	P_3	P_4

Service discovery. A user posts a service query to ROSSE via its Web user interface (step 5). The query includes a service category of interest and expected service properties. The query is then passed to the Irrelevant Property Identification component (step 6), which accesses the ROSSE Service Repository (step 7) to identify and mark the properties of advertised services that are irrelevant to the properties used in the service query based on the ontologies defined in the ROSSE Ontology Repository (step 8). The query is then passed to the DPR component (step 9), which accesses the ROSSE Service Repository to identify and mark dependent properties (step 10). Upon completion, the DPR component invokes the Service Similarity Computing (SSC) component (step 11), which accesses ROSSE Service Repository (step 12) to compute the match degrees of relevant properties of advertised services to the service query. An irrelevant property is given a match degree of zero. The SSC component further computes the similarity degrees of advertised services to the service query using the match degrees of their individual properties. It should be noted that dependent properties that may be uncertain properties are not involved in the similarity computing process. As a result, the similarity degrees of advertised services will not be affected by these uncertain properties. In this way, ROSSE can discover the services that are most relevant to the service query. Up to now, advertised services are matched with their functional properties. As functionally matched services may have distinct nonfunctional properties related to QoS, the SSC component invokes the QoS Modeling component (step 13), which in turn filters functionally matched services

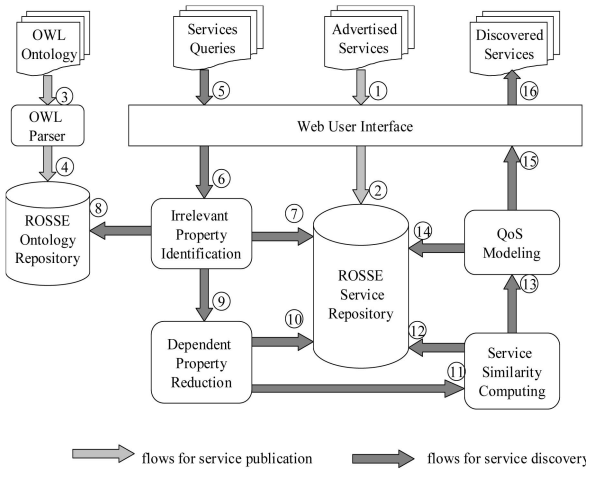

Fig. 2. ROSSE components.

with QoS values (step 14). Finally, a list of discovered services that are ranked with their functionally matched degrees is presented to a user (step 16) via the Web user interface of ROSSE (step 15). Each of the discovered services has a QoS value associated with it.

In the following sections, we describe in depth the processes involved in service discovery in ROSSE. First, we introduce Rough sets for service discovery.

2.1 Rough Sets for Service Discovery

The Rough sets theory can be considered as a mathematical technique to deal with uncertainty in knowledge discovery [35]. A fundamental principle of a Rough sets-based learning system is to discover redundancies and dependencies between the given features of a problem to be classified. The Rough sets theory approaches a given concept using lower and upper approximations.

Let

- Ω be a domain ontology,
- U be a set of N advertised services whose properties are defined in Ω, $U = \{s_1, s_2, \ldots, s_N\}$, $N \geq 1$,
- P be a set of K properties that describe the N advertised services of the set U, $P = \{p_1, p_2, \ldots, p_K\}$, $K \geq 1$,
- P_A be a set of M properties that are relevant to the properties used in a service query Q in terms of Ω, $P_A = \{p_{A1}, p_{A2}, \ldots, p_{AM}\}$, $P_A \subseteq P$, $M \geq 1$,
- X be a set of advertised services that are relevant to the service query Q in terms of Ω, $X \subseteq U$,
- $\underline{X}$ be a lower approximation of the set X,
- $\overline{X}$ be an upper approximation of the set X, and
- $[x]_{P_A}$ be a set of advertised services that are exclusively defined by the properties of the set P_A, $x \in U$.

According to the Rough sets theory, we have

$$\underline{X} = \{x \in U : [x]_{P_A} \subseteq X\}, \tag{1}$$

$$\overline{X} = \{x \in U : [x]_{P_A} \cap X \neq \emptyset\}. \tag{2}$$

For a service property $p \in P_A$, we have the following:

- $\forall x \in \underline{X}$, x definitely has property p.
- $\forall x \in \overline{X}$, x possibly has property p.
- $\forall x \in U - \overline{X}$, x absolutely does not have property p.

For a service query, there could be a large number of matched services. Using the size of the set $\underline{X}$, a user can dynamically determine the size of the set $\overline{X}$ that would maximize user satisfaction in service discovery. The selection of services based on lower and upper approximations will be further discussed in Section 2.5.

2.2 Irrelevant Property Identification

The properties used in a service advertisement may have semantic relationships with the properties used in a service query based on the definition of a domain ontology.

Let

- p_Q be a property used in a service query and
- p_A be a property used in a service advertisement.

We define the following relationships between p_Q and p_A based on the work proposed by Paolucci et al. [15]:

- *Exact* match. p_Q and p_A are equivalent, or p_Q is a subclass of p_A.
- *Plug-in* match. p_A subsumes p_Q.
- *Subsume* match. p_Q subsumes p_A.
- *Nomatch.* There is no subsumption between p_Q and p_A.

If p_A has a *nomatch* relationship with each p_Q used in a service query, then p_A will be marked as an irrelevant property when matching the service query.

As introduced in Section 1, uncertainty of properties may exist in advertised services when matching a service query. An uncertain property is a service property that is explicitly used in one advertised service but does not appear in another service advertisement that belongs to the same service category. As advertised services are structured as service records in ROSSE using a database, we define an *uncertain* relationship between p_Q and p_A as follows:

- *Uncertain.* There is no subsumption between p_Q and p_A, and p_A has an empty value, which is NULL.

2.3 Dependent Property Reduction

The properties of advertised services may have dependencies by which ROSSE deals with uncertainty of properties. Dependent properties are indecisive (redundant) properties that can be reduced when matching services. A reduct is a set of decisive properties that are sufficient enough to describe those advertised services that are relevant to a service query. Based on the Rough sets theory, ROSSE identifies indecisive properties in the following way:

Let

- Ω, X, P, and P_A be defined as in Section 2.1,
- P_A^D be a set of L_D decisive properties when matching a service query Q in terms of Ω, $P_A^D = \{p_{A1}^D, p_{A2}^D, \ldots, p_{AL_D}^D\}$, $P_A^D \subseteq P_A$, $L_D \geq 1$,
- P_A^{IND} be a set of L_{IND} indecisive (dependent) properties when matching the service query Q in terms of Ω,

$$P_A^{IND} = \{p_{A1}^{IND}, p_{A2}^{IND}, \ldots, p_{AL_{IND}}^{IND}\},$$
$$P_A^{IND} \subseteq P_A, L_{IND} \geq 1,$$

- $IND()$ be an indiscernibility relation,
- f be a mapping function from a property to an advertised service,
- $S(Q, s)$ be the similarity degree of an advertised service s to the service query Q in terms of Ω, $s \in X$ ($S(Q, s)$ can be computed using (6) to be presented in Section 2.4),
- Y be a set of objective services that are relevant to the service query Q, $Y \subseteq X$ (we define $Y = \{\forall(x, y) \in Y, S(Q, x) = S(Q, y) : \forall s \in (X - Y), S(Q, s) < S(Q, x)\}$),
- $[Y]_{P_A}$ be a set of objective services that are defined by the properties of the set P_A,
- $[Y]_{P_A^D}$ be a set of objective services that are defined by the decisive properties of the set P_A^D, and

- $[Y]_{(P_A - P_A^{IND})}$ be a set of objective services that are defined by the properties of the set $P_A - P_A^{IND}$.

Then, we have

$$IND(P_A^{IND}) = \{(x,y) \in X : \forall p_{Ai}^{IND} \in P_A^{IND}, f(x, p_{Ai}^{IND}) = f(y, p_{Ai}^{IND})\}, \quad (3)$$

$$[Y]_{P_A} = [Y]_{P_A^D} = [Y]_{(P_A - P_A^{IND})}. \quad (4)$$

Expression (3) shows that indecisive properties are dispensable in differentiating advertised services, and (4) further indicates that the set of objective services can always be identified regardless of the existence of indecisive properties. Algorithm 1 shows the discovery of decisive properties in which lines 1-7 are used to identify individual indecisive properties, and lines 8-15 are used to check all possible combinations of these individual indecisive properties with an aim to compute a maximal set of indecisive properties. It should be noted that some uncertain properties of advertised services may be indecisive properties. As a result, these uncertain properties will be reduced when computing the similarity degrees of services. In other words, for a service query, some uncertain properties may not affect the similarity degree of an advertised service. Accordingly, ROSSE increases the accuracy of service discovery.

Algorithm 1: Algorithm for Dependent Property Reduction
Input: P_A, which is a set of service properties that are relevant to a service query;
Output: P_A^D, which is a set of decisive service properties that are relevant to a service query;
1: for each property p∈ P_A
2: if p is an indecisive property based on expression (4), then
3: add p into P_A^{IND};
4: $P_A^{IND\_Core} = \emptyset$;
5: add p into $P_A^{IND\_Core}$;
6: end if
7: end for
8: for i= sizeof(P_A^{IND})-1 to 2
9: compute all possible i combinations of the properties of P_A^{IND};
10: if any combined i properties are indecisive properties based on expression (4), then
11: $P_A^{IND\_Core} = \emptyset$;
12: add the i properties into $P_A^{IND\_Core}$;
13: break;
14: end if
15: end for
16: $P_A^D = P_A - P_A^{IND\_Core}$;
17: return P_A^D;

2.4 Computing Similarity Degrees

As described in Section 2.2, a match between p_Q and p_A can be *exact*, *plug-in*, *subsume*, *uncertain*, or *nomatch.* For each match, a numerical degree should be assigned so that the relationship between p_Q and p_A can be quantified. Tsetsos et al. [37] use fuzzy set theory to evaluate service matchmaking. The relationships between the properties of advertised services and the properties of service queries are mapped to fuzzy linguistic variables, e.g., *exact* is mapped to *very relevant*, and *plug-in* is mapped to *relevant.* In this work, linear trapezoidal membership functions are assumed for capturing the vagueness of the various linguistic terms. The preliminary results show the effectiveness of the fuzzy linguistic approach in quantifying match degrees for service matchmaking. However, one major concern with the approach is that the mapping from a semantic relationship to a fuzzy variable does not consider the semantic distances of the properties involved. For example, two *plug-in* matches with different semantic distances are mapped to the same fuzzy variable that is *relevant.* To increase the accuracy in assigning matching degrees between p_Q and p_A, semantic distances should be taken into account.

Let

- $dom(p_Q, p_A)$ be the degree of a match between p_Q and p_A and
- $\|P_Q, P_A\|$ be the semantic distance between p_Q and p_A in terms of a domain ontology Ω.

Following the work proposed in [36] for assigning match degrees, we further define $dom(p_Q, p_A)$ as follows:

$$dom(P_Q, P_A) = \begin{cases} 1 & exact \text{ match}, \\ \frac{1}{2} + \frac{1}{e^{(\|P_Q, P_A\| - 1)}} & plugin \text{ match}, \|P_Q, P_A\| \geq 2, \\ \frac{1}{2 \times e^{(\|P_Q, P_A\| - 1)}} & subsume \text{ match}, \|P_Q, P_A\| \geq 1, \\ 0.5 & uncertain \text{ match}, \\ 0 & nomatch. \end{cases} \quad (5)$$

According to (5), for a *plug-in* match between p_Q and p_A, $dom(p_Q, p_A) \in (0.5, 1)$. For a *subsume* match between p_Q and p_A, $dom(p_Q, p_A) \in (0, 0.5]$.

Let

- P_A^D and $S(Q, s)$ be defined as in Section 2.3,
- P_Q be a set of M properties used in a service query Q, $P_Q = \{p_{Q1}, p_{Q2}, \ldots, p_{QM}\}$, $M \geq 1$, and
- $dom(p_{Qi}, p_{Aj})$ be a match degree between P_{Qi} and P_{Aj} in terms of a domain ontology Ω, $p_{Qi} \in P_Q$, $1 \leq i \leq M$, $p_{Aj} \in P_A^D$, $1 \leq j \leq L_D$.

As every decisive property of an advertised service s has a maximal match degree when matching all the properties used in a service query, $S(Q, s)$ can be computed using the following:

$$S(Q,s) = \sum_{j=1}^{L_D} \sum_{i=1}^{M} \max(dom(p_{Qi}, p_{Aj})) \Big/ L_D. \quad (6)$$

Therefore, each advertised service has a similarity degree to a service query.

2.5 Lower and Upper Approximations of Matched Services

For a service query, the number of matched services could be large. To facilitate users in choosing the services that would maximally satisfy their queries, the set of discovered services need to be dynamically determined. We apply the concept of lower and upper approximations of Rough sets for this purpose.

Let

- U, X, $\underline{X}$, and $\overline{X}$ be defined as in Section 2.1,
- $S(Q, s)$ be defined as in Section 2.3,
- $|\underline{X}|$ be the cardinality of the set $\underline{X}$,
- $|\overline{X}|$ be the cardinality of the set $\overline{X}$,
- ρ be an approximation degree, $\rho = \frac{|\underline{X}|}{|\overline{X}|}$, $0 < \rho \leq 1$, $|\overline{X}| \neq 0$,

- X_1 be a set of advertised services, $\forall s \in X_1$, $S(Q,s) = 100$ percent,
- $|X_1|$ be the cardinality of the set X_1,
- X_2 be a set of advertised services, $\forall s \in X_2$, $0 < S(Q,s) < 100$ percent, and
- $|X_2|$ be the cardinality of the set X_2.

Then, according to (1) and (2), we have

$$\underline{X} = \begin{cases} X_1 & |X_1| > 0, \\ \underline{X} \subseteq X_2 : |\underline{X}| = |X_2| \times \rho & |X_1| = 0, \end{cases} \quad (7)$$

$$\overline{X} = \begin{cases} X_1 & |X_1| > 0, \\ X_2 & |X_1| = 0. \end{cases} \quad (8)$$

If the set X_1 exists, then the set X_1 will be presented to a user as both the upper and the lower approximation sets of matched services for the service query.

If the set X_1 does not exist, then a user can apply an approximation degree ρ to dynamically determine the lower approximation set of matched services. In this way, advertised services with low similarity degrees may not be presented to the user. The set X_2 will be used as the upper approximation set of matched services for the service query.

3 QoS Modeling

As described in Section 2, ROSSE uses the functional properties of services to match services. However, services may have distinct nonfunctional properties related to QoS. To maximize user satisfaction in a service query, functionally matched services should be further filtered with their QoS properties. Zeng et al. [26] propose a set of QoS properties for Web service composition. In this section, we revisit these QoS properties in terms of grid computing environments. We classify QoS properties into two classes—system-related properties and non-system-related properties.

3.1 System-Related QoS Properties

The performance of a service is largely affected by the capacity of a computing environment that hosts the service. We discuss three QoS properties in this category, i.e., reliability, execution efficiency, and availability.

3.1.1 Reliability

The reliability $q_{reliability}(s)$ of a service s represents the ability of the service to perform its required functions under stated conditions for a specified period of time [40]. In ROSSE, the $q_{reliability}(s)$ of service s is measured with its successful execution rate, which can be computed using the statistical approach proposed in [26].

Let

- S be a set of services that are functionally matched to a service query, $S = \{s_1, s_2, \ldots, s_n\}$, $n \geq 1$,
- E_i be the successful execution rate of service s_i ($s_i \in S$) for a given period of time, $E_i = \frac{n_{exe}(s_i)}{N_{invoke}(s_i)}$, where $n_{exe}(s_i)$ is the number of times that service s_i has been successfully completed within the maximum expected time frame as specified in the service description, and $N_{invoke}(s_i)$ is the total number of invocations of service s_i,
- $E_{\max}$ be the maximal successful execution rate among the list of

$$\{E_1, E_2, \ldots, E_n\}, E_{\max} = \max(E_1, E_2, \ldots, E_n),$$

and
- $E_{\min}$ be the minimal successful execution rate among the list of

$$\{E_1, E_2, \ldots, E_n\}, E_{\min} = \min(E_1, E_2, \ldots, E_n).$$

Then, we define

$$q_{reliability}(s_i) = \begin{cases} \frac{E_i - E_{\min}}{E_{\max} - E_{\min}} & E_{\max} \neq E_{\min}, \\ 1 & E_{\max} = E_{\min}. \end{cases} \quad (9)$$

It should be noted that running a service reliably demands a certain amount of CPU processing power and memory space. The reliability of a service is largely affected by the capacity of the grid that hosts the service. A grid is a dynamic computing environment in nature. For example, computing nodes may join or leave the environment dynamically. Services can be dynamically deployed to a certain node in the environment upon request. The workload of a node may change frequently. Considering the dynamic nature of grid environments, we further tune the computed $q_{reliability}(s_i)$ of service s_i in such a way that $q_{reliability}(s_i) = K \times q_{reliability}(s_i)$. K is a variable that can be computed using $K = \frac{n_{node}(s_i)}{N_{node}}$, where $n_{node}(s_i)$ is the number of computing nodes in a grid environment that meet both the CPU and memory requirements of service s_i at the time when the service is requested, and N_{node} is the total number of computing nodes in the grid at that time. The resource information on the usage of CPU and memory of computing nodes in a grid can be collected using a monitor system such as Ganglia.[1]

3.1.2 Execution Efficiency

The execution efficiency $q_{efficiency}(s)$ of a service s refers to how fast the service can be executed. The $q_{efficiency}(s)$ of service s is measured in terms of its execution duration, which could be computed using the approach proposed in [26]. The execution duration of a service is the sum of the processing time and the transmission time. It is worth noting that computing the processing time of a service in a dynamic grid environment is a challenging issue itself [41], [42]. The work presented in [26] does not provide an approach to compute the processing time of a service. In ROSSE, we measure the execution duration of a service and compute its execution efficiency in the following way:

Let

- S be a set of services that are functionally matched to a service query, $S = \{s_1, s_2, \ldots, s_n\}$, $n \geq 1$,
- T_i be a measured execution duration of service s_i ($s_i \in S$) running on a dedicated computing node

1. http://ganglia.sourceforge.net.

(each service runs exclusively on the node during its execution),

- $T_{\max}$ be the maximal execution duration among the list of $\{T_1, T_2, \ldots, T_n\}$, $T_{\max} = \max(T_1, T_2, \ldots, T_n)$, and
- $T_{\min}$ be the minimal execution duration among the list of $\{T_1, T_2, \ldots, T_n\}$, $T_{\min} = \min(T_1, T_2, \ldots, T_n)$.

Then, we define

$$q_{efficiency}(s_i) = \begin{cases} \frac{T_{\max} - T_i}{T_{\max} - T_{\min}} & T_{\max} \neq T_{\min}, \\ 1 & T_{\max} = T_{\min}. \end{cases} \quad (10)$$

3.1.3 Availability

The availability $q_{availability}(s)$ of a service s is defined as the probability that service s can be accessed at a particular time [26]. In ROSSE, $q_{availability}(s)$ is computed using $q_{availability}(s) = \frac{n_{avail}(s)}{N_{invoke}(s)}$. For a given period of time, $n_{avail}(s)$ is the number of successful invocations of service s within the maximum expected time frame as specified in service description, and $N_{invoke}(s)$ is the total number of invocations of service s.

3.2 Non-System-Related QoS Properties

We classify non-system-related QoS properties into cost-effectiveness and reputation.

3.2.1 Cost-Effectiveness

The cost-effectiveness $q_{\cos t\_effectiveness}(s)$ of a service s refers to the effectiveness of cost in using the service. In ROSSE, cost-effectiveness is computed in the following way:

Let

- S be a set of services that are functionally matched to a service query, $S = \{s_1, s_2, \ldots, s_n\}$, $n \geq 1$,
- C_i be the cost of using service s_i, $s_i \in S$, $1 \leq i \leq n$,
- $C_{\max}$ be the maximal cost among the list of $\{C_1, C_2, \ldots, C_n\}$, $C_{\max} = \max(C_1, C_2, \ldots, C_n)$, and
- $C_{\min}$ be the minimal cost among the list of $\{C_1, C_2, \ldots, C_n\}$, $C_{\min} = \min(C_1, C_2, \ldots, C_n)$.

Then, we define

$$q_{\cos t\_effectiveness}(s_i) = \begin{cases} \frac{C_{\max} - C_i}{C_{\max} - C_{\min}} & C_{\max} \neq C_{\min}, \\ 1 & C_{\max} = C_{\min}. \end{cases} \quad (11)$$

3.2.2 Reputation

The reputation $q_{reputation}(s)$ of a service s is a ranking degree of a user's experience in using the service. In ROSSE, $q_{reputation}(s)$ is computed using the following:

$$q_{reputation}(s) = \frac{\sum_{j=1}^{k} RD_j(s)}{k},$$

where $RD_j(s)$ is a ranking degree of user j on using service s, $0 \leq RD_j(s) \leq 1$, and k is the total number of users involved in the evaluation process.

3.3 Overall QoS Values of Functionally Matched Services

Based on the computed values of the aforementioned QoS properties, the overall QoS value $Q(s)$ of service s can be computed using the following:

$$Q(s) = w_1 q_{availability} + w_2 q_{reliability} + w_3 q_{efficiency} + w_4 q_{\cos t\_effectiveness} + w_5 q_{reputation}, \quad (12)$$

where w_1, w_2, w_3, w_4, and w_5 are the weights of availability, reliability, execution efficiency, cost-effectiveness, and reputation of service s, respectively, $w_i \in [0, 1]$, $\sum_{i=1}^{5} w_i = 1$. The weights are assigned based on user preferences.

4 ROSSE Case Study

In this section, we briefly describe the implementation of ROSSE. Then, we give a case study to illustrate the application of ROSSE to discover computing services.

4.1 ROSSE Implementation

ROSSE is implemented as a Web system using Java and Web technologies. ROSSE has a Web user interface, as shown in Fig. 3, for service publication and discovery. ROSSE provides users with graphical user interfaces for publishing services with WSDL interfaces. ROSSE uses the OWL-S API to directly register services with OWL-S interfaces. The ROSSE Service Repository consists of a UDDI registry for WSDL services and a service repository for OWL-S services. jUDDI and mySQL are used to build the UDDI registry. The OWL-S service repository is also structured as a database that records service elements such as service names and service properties.

To facilitate the reduction of dependent properties, service records are structured in such a way that each column has only one property associated with it. Ontologies defined in OWL documents are loaded into ROSSE using the Protégé OWL API, which in turn invokes RACER [27] to infer a semantic relationship between two properties defined in an OWL ontology. However, we implemented a light-weighted reasoner in ROSSE to replace RACER. The main reason for this is that RACER has a high overhead when parsing multiple OWL documents. Each time RACER parses an OWL document, it needs an initialization process that is time consuming. It should be pointed out that ROSSE services that belong to the same service category may be defined with distinct ontologies. For example, two ontologies used by ROSSE may have the same set of service properties but with different topologies. ROSSE manages multiple ontologies in its ontology repository. When requested, ROSSE only loads the ontology repository once, and it maintains multiple ontologies in memory. The relationships of the properties defined in these OWL ontologies are kept in a database. The ROSSE reasoner accesses these database records to make inferences between multiple ontologies. We compare the overhead of RACER with that of the ROSSE reasoner in Section 5.

4.2 Discovery of Computing Services in ROSSE

Fig. 4 shows the ontologies used in this case study defining the classifications of grid entities, security, virtual organization (VO) management, computing resource properties, CPU properties, and hard-disk properties, respectively.

When matching properties related to VO management, two ontologies (strictly two ontology topologies) are used to classify VOs, which are represented respectively by *c1-c4*

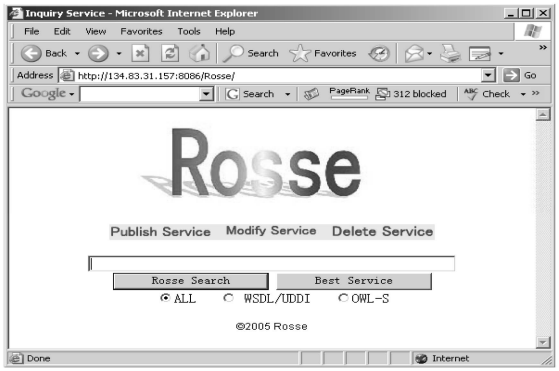

Fig. 3. The Web user interface of ROSSE.

and *g1-g3*. Computing services are registered with the ROSSE Service Repository. In the following parts, we describe how services are matched and discovered using the following query:

Query: gridentity, ID, highcost, vomanage, 2.6 GHz, highspeedHD, high reliability

4.2.1 Building a Decision Table

A service decision table is used to identify dependent properties among services. As the number of services registered with ROSSE is large, the decision table is constructed by sampling registered services. For a specific query, ROSSE randomly selects a certain number of service records. An advertised service record is selected if one of its properties has a valid relationship with a property used in a service query. The relationship can be *exact*, *plug-in*, or *subsume*, as described in Section 2.2.

Table 2a shows a segment of the decision table with 13 advertised service records. As can be seen in Table 2a, the properties of advertised services that are relevant to the service query are *d3*, *b4*, *e4*, *f3*, *f5*, *d8*, *d6*, *f2*, *c4/g3*, *e1*, and *b3*. If a property in a service record is marked with *1*, this means that the property is explicitly used by the service in its advertisement. For example, the service S_1 has properties of *d3*, *b4*, *e4*, *f3*, *e1*, and *b3* in its advertisement. A property marked with X in a service record means that the service does not explicitly have the corresponding property in its advertisement. It should be noted that a property marked with X in a service record does not necessarily mean that this property is not relevant to the service. Such a property could be dependent on other properties used by the service. ROSSE considers properties marked with X as uncertain properties when matching services.

4.2.2 Reducing Dependent Properties

Once a service decision table is constructed, the next step is to identify dependent properties. Using Algorithm 1, presented in Section 2.3, we identify that properties *b4*, *e4*, and *f3* are dependent (indecisive) properties that can be reduced from the decision table when computing the similarity degrees of services. Table 2b shows the dependent properties identified, and Table 2c shows the segment of the decision table without dependent properties.

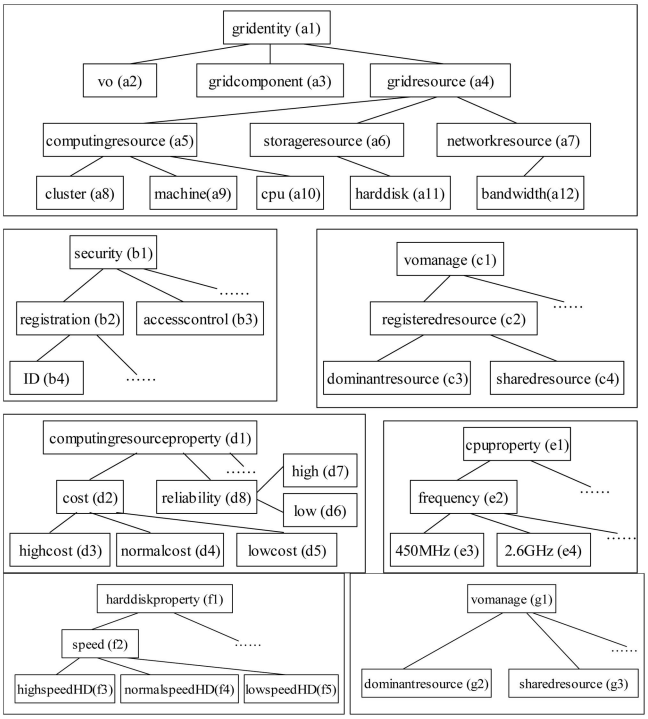

Fig. 4. Ontologies used in searching for computing services.

4.2.3 Computing Similarity Degrees

Decisive properties are used for computing the similarity degrees of advertised services to a service query. A match degree can be computed between each decisive property used in a service advertisement and a property used in the service query using the ontologies defined in Fig. 4 and (5),

TABLE 2
(a) A Segment of the Decision Table; (b) Identified Dependent Properties; (c) the Decision Table without Dependent Properties; (d) Match Degrees to the Service Query

properties / services	d3	b4	e4	f3	d7	f2	c4/g3	e1	b3
S_1	1	1	1	1	X	X	X	1	1
S_2	X	1	X	1	X	X	X	1	X
S_3	X	1	X	1	X	1	1	X	X
S_4	X	1	X	X	1	1	1	X	X
S_5	X	1	1	X	X	X	1	X	X
S_6	1	1	1	1	X	X	X	X	X
S_7	X	1	X	X	X	1	X	X	X
S_8	1	1	1	1	X	X	1	X	X
S_9	X	1	X	1	X	1	X	1	X
S_{10}	X	1	X	X	X	X	X	1	X
S_{11}	X	1	X	X	X	1	X	X	X
S_{12}	X	1	X	1	X	X	X	1	1
S_{13}	1	1	1	1	1	X	1	X	X

(a)

properties / services	d3	b4	e4	f3	d7	f2	c4/g3	e1	b3
S_1	1	1	1	1	X	X	X	1	1
S_2	X	1	X	1	X	X	X	1	X
S_3	X	1	X	1	X	1	1	X	X
S_4	X	1	X	X	1	1	1	X	X
S_5	X	1	1	X	X	X	1	X	X
S_6	1	1	1	1	X	X	X	X	X
S_7	X	1	X	X	X	1	X	X	X
S_8	1	1	1	1	X	X	1	X	X
S_9	X	1	X	1	X	1	X	1	X
S_{10}	X	1	X	X	X	X	X	1	X
S_{11}	X	1	X	X	X	1	X	X	X
S_{12}	X	1	X	1	X	X	X	1	1
S_{13}	1	1	1	1	1	X	1	X	X

(b)

properties / services	d3	d7	f2	c4/g3	e1	b3
S_1	1	X	X	X	1	1
S_2	X	X	X	X	1	X
S_3	X	X	1	1	X	X
S_4	X	1	1	1	X	X
S_5	X	X	X	1	X	X
S_6	1	1	X	X	X	X
S_7	X	X	1	X	X	X
S_8	1	X	X	1	X	X
S_9	X	X	1	X	1	X
S_{10}	X	X	X	X	1	X
S_{11}	X	X	1	X	X	X
S_{12}	X	X	X	X	1	1
S_{13}	1	1	X	1	X	X

(c)

matching degrees	100%	100%	100%	34%	87%	64%
properties / services	d3	d7	f2	c4/g3	e1	b3
S_1	1	X	X	X	1	1
S_2	X	X	X	X	1	X
S_3	X	X	1	1	X	X
S_4	X	1	1	1	X	X
S_5	X	X	X	1	X	X
S_6	1	1	X	X	X	X
S_7	X	X	1	X	X	X
S_8	1	X	X	1	X	X
S_9	X	X	1	X	1	X
S_{10}	X	X	X	X	1	X
S_{11}	X	X	1	X	X	X
S_{12}	X	X	X	X	1	1
S_{13}	1	1	X	1	X	X

(d)

TABLE 3
Some Results of ROSSE in Matching Services

Advertised Services	Properties					Similarity Degrees		
	P1	P2	P3	P4	P5	UDDI	OWL-S	ROSSE
S1	S	E	P	P	E	40%	85%	96%
S2	U	E	P	U	E	67%	0	96%
S3	S	U	P	U	E	33%	0	79%
S4	U	P	P	P	E	25%	0	91%
S5	S	U	P	P	U	0	0	62%

E: Exact (100%) P: Plugin (87%) S: Subsume (50%) U: Uncertain (50%)
P1,P4: Dependent Properties

presented in Section 2.4. Table 2d shows match degrees of the decisive properties used in the 13 service records. It should be noted that both *c1* and *g1* refers to the same property *vomanage*, but they are defined with different ontology topologies. The match degree of the *vomanage* property used in the query to the *sharedresource* property used in advertised services is computed in such a way that a mean of two match degrees using the two ontology definitions (i.e., 50 percent and 18 percent) is computed, which is 34 percent.

It is worth noting that for an uncertain property that is marked with X in Table 2d, a match degree of 50 percent is given using (5), presented in Section 2.4. The similarity degree of an advertised service to a service query can be computed using (6), presented in Section 2.4. For the service query, for example, service S_1 has a similarity degree of 67 percent, and service S_{13} has a similarity degree of 64 percent.

5 ROSSE Evaluation

To evaluate ROSSE, we conducted a set of experiments. The evaluation was focused on the accuracy and efficiency of ROSSE in service discovery. In this section, we present the evaluation results.

5.1 Accuracy of ROSSE in Service Discovery

ROSSE can discover WSDL/UDDI and OWL-S services. We compare ROSSE with UDDI keyword matching and the OWL-S matching [15], respectively, from the aspect of accuracy in service discovery. First, we show how ROSSE increases the similarity degrees of relevant services to a service query.

5.1.1 Increased Similarity Degrees of ROSSE

Table 3 shows five services (S1-S5) that are relevant to a service query. Each service has five properties (P1-P5), of which P1 and P4 are dependent properties. Except for service S1, all the other four services S2-S5 have uncertain properties in their service records. The match degrees of *exact*, *plug-in*, *subsume*, and *uncertain* are assigned 100 percent, 87 percent, 50 percent, and 50 percent, respectively.

We observe that UDDI keyword matching, OWL-S matching, and ROSSE produce different similarity degrees when matching the five services. In the case of service S1, UDDI has a match degree of 40 percent, OWL-S has 85 percent, and ROSSE has 96 percent. UDDI produces the lowest similarity degree because it only supports an *exact* match. ROSSE performs best because of its reduction of dependent properties (i.e., P1 and P4). OWL-S matching cannot deal with uncertain properties. As a result, OWL-S matching produces a similarity degree of zero when matching services S2-S5 with uncertain properties. In the case of service S5, UDDI produces a match degree of zero because there is no *exact* match in the service advertisement. ROSSE shows its effectiveness in dealing with uncertain properties when matching services S2-S5 with a match degree of 96 percent, 79 percent, 91 percent, and 62 percent, respectively.

5.1.2 Measuring Precision and Recall

Precision and recall are standard measures that have been used in information retrieval for measuring the accuracy of a search method or a search engine [38], [39]. To evaluate the precision and recall of ROSSE in service discovery, we selected a set of 30 services, of which 10 services were relevant to a service query. Each service had five properties, of which two were dependent properties. We performed two groups of tests, of which each group involved 10 tests. For each test, we produced a list in which the 30 services were listed in a random order. In the tests of group 1, we enforced two constraints on the selected services. First, no service had an uncertain property. Second, at least one property of a service was assigned an *exact* match. This ensured that all the relevant services were returned by UDDI matching, OWL-S matching, and ROSSE, respectively. In the tests of group 2, we removed the two constraints. We allowed a few services in each of the 10 lists to have properties that did not have an *exact* match. We also assigned some services with uncertain properties. Upon the completion of matching, the services in each list were returned with their match degrees in a descending order. It should be noted that services with a match degree of zero were not returned.

Let $\mathrm{Re}l$ be the set of relevant services, $\mathrm{Re}t$ be the set of returned services, $\mathrm{Re}trel$ be the set of returned relevant services, R_C represent recall, and PRE_C represent precision. We define

$$R_C = \frac{|\mathrm{Re}\ trel|}{|\mathrm{Re}\ l|},\ PRE_C = \frac{|\mathrm{Re}\ trel|}{|\mathrm{Re}\ t|}.$$

Figs. 5 and 6 show the averaged results of the 10 tests in group 1 and group 2, respectively.

We observe that ROSSE achieves the best performance in the tests of group 1, whereas UDDI has the worst performance because of its keyword matching. For example, when the recall is 30 percent, the precisions of ROSSE, OWL-S, and UDDI are 95 percent, 84 percent, and 65 percent, respectively. This is mainly because of the capability of ROSSE in the reduction of dependent properties and its use of semantic inference techniques, as outlined in Sections 2.2 and 2.3, respectively.

We observe that in most cases, ROSSE shows the best performance in the tests of group 2. However, OWL-S performs better than ROSSE in two cases when recall is 10 percent and 30 percent, respectively. The reason is that OWL-S does not deal with uncertain properties. As a result,

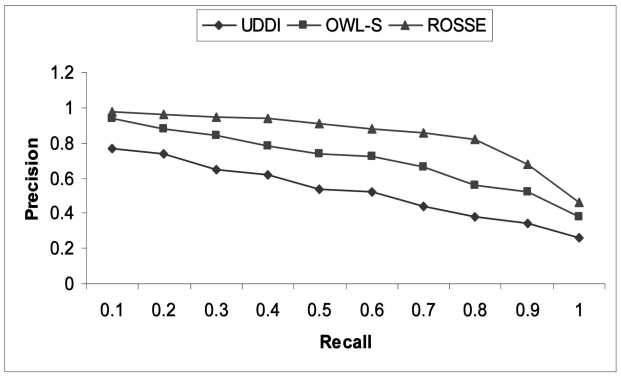

Fig. 5. The performance of ROSSE, OWL-S, and UDDI in the tests of group 1.

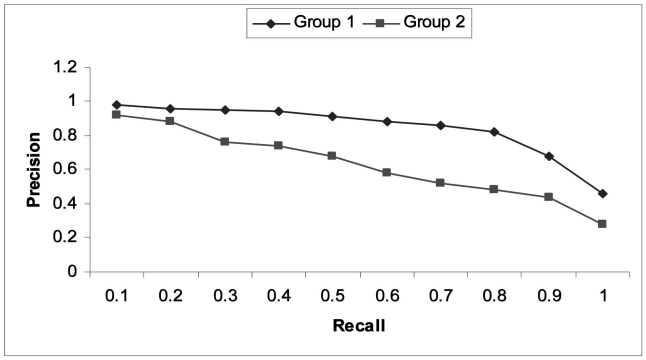

Fig. 7. The performance of ROSSE in group 1 and group 2.

some irrelevant services are not returned because of the existence of uncertain properties. This leads to a high precision for some instances. We also observe that in the tests of group 2, UDDI performs better than OWL-S in some cases. The reason is that some relevant services that have uncertain properties are not matched by OWL-S. However, these relevant services are matched by UDDI because of the existence of an *exact* match in their properties.

It should be noted that both OWL-S and UDDI do not reach a recall of 100 percent in the tests of group 2 because of their limitations in service matching. For example, OWL-S only matches six of the 10 relevant services, and UDDI matches eight relevant services.

We also compared the performance of ROSSE in the tests of group 1 and group 2, respectively. The results are plotted in Fig. 7 showing that ROSSE performs better in group 1 than in group 2. This can be explained by the existence of uncertain properties of services in the tests of group 2 in which ROSSE loses some useful information on relevant services when matching the query. It is worth noting that ROSSE matches all the relevant services in the tests of both group 1 and group 2, and it performs reasonably well in terms of precision and recall.

5.2 Efficiency of ROSSE in Service Discovery

ROSSE was evaluated on a Pentium IV 2.6-GHz machine with 512 Mbytes of RAM running Red Hat Fedora Linux 3. We compared the efficiency of ROSSE with that of UDDI and the OWL-S algorithm [15], respectively. RACER was used by the OWL-S algorithm to reason the relationships of properties. As described in Section 4.1, RACER has a time-intensive initialization process for parsing every OWL document. We implemented a light-weighted reasoner in ROSSE to overcome the high overhead incurred by RACER when parsing multiple OWL documents. We compared the overhead of the ROSSE reasoner with that of RACER using two ontologies defined in two OWL documents, and the results are shown in Fig. 8.

We observe that the difference in overhead between RACER and the ROSSE reasoner gets larger with an increase in the number of queries. This is because for each query received, RACER performs two initialization processes, which consumes roughly 12 seconds. It takes the ROSSE reasoner about 3.4 seconds to load the two ontologies. However, once loaded by the ROSSE reasoner, the two ontologies are maintained in memory for access. The overhead to parse a query is just a few hundred microseconds. As a result, the overhead of the ROSSE reasoner does not change much with an increase in the number of queries.

To evaluate the efficiency of ROSSE in service discovery, we registered 10,000 computing services with ROSSE. Each service had five properties, of which two were dependent properties. We posted a service query with two properties. Therefore, 10 ontology queries were parsed by the ROSSE reasoner and RACER, respectively, when matching the query. Service properties were defined in one ontology. We

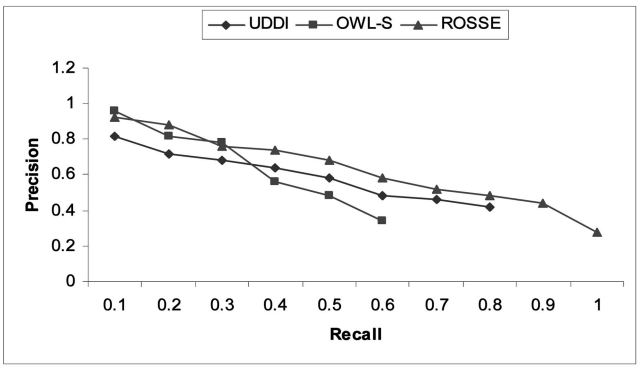

Fig. 6. The performance of ROSSE, OWL-S, and UDDI in the tests of group 2.

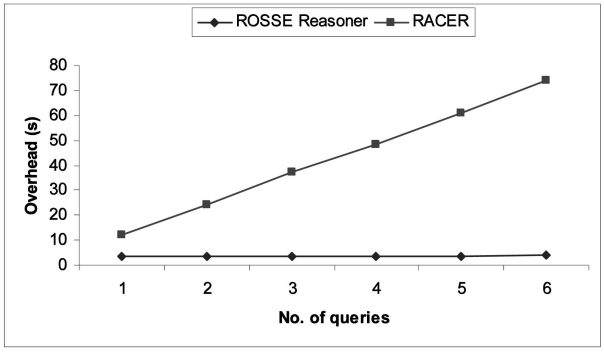

Fig. 8. The overhead of the ROSSE reasoner in the access of two ontologies.

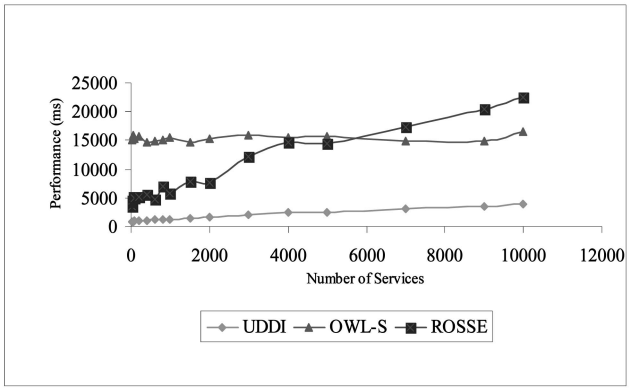

Fig. 9. The overhead of ROSSE in matching services.

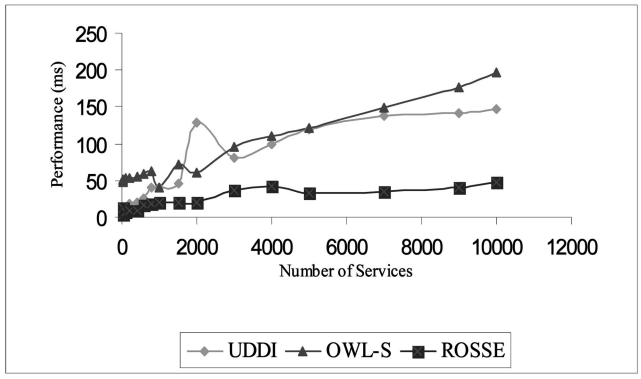

Fig. 10. Efficiency of ROSSE in accessing service records.

performed two groups of tests. In group 1, we compared the overhead of ROSSE with that of UDDI and OWL-S, respectively, in matching services. In group 2, we evaluated the efficiency of ROSSE when accessing service records. Figs. 9 and 10 show the evaluation results of the two groups, respectively.

In Fig. 9, we observe that UDDI has the least overhead when matching services. This is because UDDI only supports keyword matching. It does not incur a reasoning process, which is usually time consuming. The overhead of ROSSE in matching services is mainly caused by the process of reducing dependent properties. The reduction process gets slower with an increase in the number of services. We observe that the overhead of OWL-S matching does not change much with an increase in the number of services. This is because the overhead of OWL-S matching is mainly caused by the initialization process of RACER. Other overhead involved in OWL-S matching is small, e.g., the overhead of RACER in parsing an ontology query is just a few hundred microseconds. As a result, the number of services involved does not make much change to the overall overhead of OWL-S matching.

In Fig. 10, we observe that ROSSE performs best when accessing service records due to its reduction of dependent properties. The OWL-S matching has a similar performance to UDDI in this process.

6 Related Work

As the computational grid is evolving toward a service-oriented computing infrastructure, service discovery has been a research focus in the grid community. Grid information services such as Globus MDS [31] and R-GMA [32] facilitate discovery of resources and services in a grid environment. However, they are restricted to keyword-based queries. UDDI is an industry initiative for discovery of Web services. UDDI has been utilized by the grid community for discovery of grid services [7], [8], [11]. Similar to Globus MDS, UDDI only supports keyword matching when searching for services. Various UDDI extensions have been proposed to enhance service discovery [9], [10], [11]. Among them, UDDI-M [11] is flexible in attaching metadata defined in RDF triples to various entities associated with a service. Building on UDDI, the Grimoires service registry [33] supports multiple service description models, and it takes into account robustness, efficiency, and security issues.

Semantic Web technologies [12] can be used to further enhance service discovery. As shown in Fig. 1, services can be annotated with metadata whose relationships are typically defined with a domain ontology. One key technology to facilitate service discovery with semantic annotations is OWL-S, an OWL-based ontology for encoding properties of Web services. OWL-S ontology defines a service profile for encoding a service description, a service model for specifying the behavior of a service, and a service grounding for invoking the service. Typically, a service discovery process involves a matching between the profile of a service advertisement and the profile of a service request using domain ontologies described in OWL. The service profile describes not only the functional properties of a service such as its inputs, outputs, preconditions, and effects (IOPEs) but also nonfunctional features such as name, category, and QoS-related aspects of a service. Srinivasan et al. [34] enhanced UDDI for service discovery by embedding OWL-S in a UDDI registry. Paolucci et al. [15] present a matchmaking algorithm for discovery of services with OWL-S interfaces. Building on this algorithm, a number of extensions are available. For example, Jaeger et al. [16] introduce "contravariance" in matching inputs and outputs between service advertisements and service requests using OWL-S, Li and Horrocks [17] introduce an "intersection" relationship between a service advertisement and a service request, and Majithia et al. [18] introduce reputation metrics in matching services.

Besides OWL-S, another prominent effort for semantic annotations of services is WSMO [19], which is built on four key concepts—ontologies, standard Web services with WSDL interfaces, goals, and mediators. WSMO stresses the role of a mediator in order to support interoperation between Web services. A mechanism is also proposed for discovery of WSMO services [20].

Although the aforementioned approaches and algorithms are available to facilitate the discovery of grid services, these efforts have never considered uncertainty of properties when matching services. As a result, they may potentially produce a low accuracy when matching services. Building on the Rough sets theory, ROSSE is capable of reducing uncertain properties. In this way, ROSSE increases the accuracy of service discovery. ROSSE represents an initial but significant advance toward solving

uncertainty of properties when matching services for high accuracy in service discovery.

7 Conclusion and Future Work

In this paper, we have presented ROSSE, a search engine for discovery of grid services. ROSSE builds on the Rough sets theory to dynamically reduce uncertain properties when matching services. In this way, ROSSE increases the accuracy of service discovery. The evaluation results have shown that ROSSE significantly improves the precision and recall compared with UDDI keyword matching and OWL-S matching, respectively. We have also introduced a QoS model to filter functionally matched services with their QoS-related nonfunctional performance. To maximize user satisfaction in service discovery, ROSSE dynamically determines the set of services that will be presented to users based on the lower and upper approximations of relevant services. We expect to carry out the following work to improve ROSSE in the future:

- *Efficiency.* It has been shown that finding a minimal reduct in Rough sets is an NP-hard problem when the number of properties gets large [28]. Heuristic methods need to be investigated to speed up the process in service property reduction.
- *Scalability.* The number of services that are registered with ROSSE could be large. Scalability is another issue that needs to be addressed. UDDI version 3 provides supports for multiple registries, but the specification does not specify how these registries should be structured to enhance scalability in service registration. Distributed Hash Table (DHT)-based Peer-to-Peer (P2P) systems such as Chord [29] and Pastry [30] have shown their efficiency and scalability in content distribution and lookup. We expect that ROSSE's scalability in service discovery can be improved with DHT-structured P2P systems.
- *Deployment.* Services, once discovered by ROSSE and selected by the user, should be dynamically deployed and invoked [43]. Such deployment could be made part of an existing workflow engine such as Triana [47], which could utilize such a discovery technique like ROSSE as part of the enactment process.
- *Ontology alignment* [44]. Services may be advertised with properties that follow distinct ontologies. To further increase the accuracy of service discovery, ROSSE needs to be enhanced with the existing efforts on ontology alignment such as [45] and [46].

References

[1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing,* vol. 6, no. 2, pp. 86-93, 2002.

[2] I. Foster and C. Kesselman, *The Grid, Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998.

[3] M.P. Atkinson, D. De Roure, A.N. Dunlop, G. Fox, P. Henderson, A.J.G. Hey, N.W. Paton, S. Newhouse, S. Parastatidis, A.E. Trefethen, P. Watson, and J. Webber, "Web Service Grids: An Evolutionary Approach," *Concurrency—Practice and Experience,* vol. 17, no. 2-4, pp. 377-389, 2005.

[4] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *Computer,* vol. 35, no. 6, pp. 37-46, 2002.

[5] K. Czajkowski, D.F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework," http://www.globus.org/wsrf/specs/ws-wsrf.pdf, Mar. 2004.

[6] B. Sotomayor and L. Childers, *Globus Toolkit 4: Programming Java Services.* Morgan Kaufmann, 2005.

[7] S. Banerjee, S. Basu, S. Garg, S. Garg, S.J. Lee, P. Mullan, and P. Sharma, "Scalable Grid Service Discovery Based on UDDI," *Proc. Third Int'l Workshop Middleware for Grid Computing (MGC '05),* pp. 1-6, Dec. 2005.

[8] B. Sinclair, A. Goscinski, and R. Dew, "Enhancing UDDI for Grid Service Discovery by Using Dynamic Parameters," *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA '05),* pp. 49-59, May 2005.

[9] A. ShaikhAli, O.F. Rana, R.J. Al-Ali, and D.W. Walker, "UDDIe: An Extended Registry for Web Service," *Proc. Symp. Applications and the Internet Workshops (SAINT '03),* pp. 85-89, Jan. 2003.

[10] A. Powles and S. Krishnaswamy, "Extending UDDI with Recommendations: An Association Analysis Approach," *Proc. Joint Workshop Web Services and Model-Driven Enterprise Information Services (WSMDEIS '05),* pp. 45-54, May 2005.

[11] S. Miles, J. Papay, V. Dialani, M. Luck, K. Decker, T. Payne, and L. Moreau, "Personalised Grid Service Discovery," *IEE Proc. Software,* special issue on performance eng., vol. 150, no. 4, pp. 252-256, 2003.

[12] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.,* vol. 284, no. 4, pp. 34-43, 2001.

[13] D.L. Martin, M. Paolucci, S.A. McIlraith, M.H. Burstein, D.V. McDermott, D.L. McGuinness, B. Parsia, T.R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K.P. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach," *Proc. First Int'l Workshop Semantic Web Services and Web Process Composition (SWSWPC '04),* pp. 26-42, July 2004.

[14] D.L. McGuinness and F. van Harmelen, *OWL Web Ontology Language Overview,* World Wide Web Consortium (W3C) recommendation, http://www.w3.org/TR/owl-features, Feb. 2004.

[15] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic Matching of Web Service Capabilities," *Proc. First Int'l Semantic Web Conf. (ISWC '02),* pp. 333-347, June 2002.

[16] M.C. Jaeger, G. Rojec-Goldmann, G. Mühl, C. Liebetruth, and K. Geihs, "Ranked Matching for Service Descriptions Using OWL-S," *Proc. Comm. in Distributed Systems (KiVS '05),* pp. 91-102, Feb. 2005.

[17] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology," *Int'l J. Electronic Commerce,* vol. 8, no. 4, pp. 39-60, 2004.

[18] S. Majithia, A.S. Ali, O.F. Rana, and D.W. Walker, "Reputation-Based Semantic Service Discovery," *Proc. 13th IEEE Int'l Workshops Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE '04),* pp. 297-302, 2004.

[19] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web Service Modeling Ontology," *Applied Ontology,* vol. 1, no. 1, pp. 77-106, 2005.

[20] U. Keller, R. Lara, A. Polleres, I. Toma, M. Kifer, and D. Fensel, "WSMO Discovery," *Working Draft D5.1v0.1,* WSMO, http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112/, 2004.

[21] M. Li and M.A. Baker, *The Grid: Core Technologies.* John Wiley & Sons, 2005.

[22] M. Li, P. van Santen, D.W. Walker, O.F. Rana, and M.A. Baker, "SGrid: A Service-Oriented Model for the Semantic Grid," *Future Generation Computer Systems,* vol. 20, no. 1, pp. 7-18, 2004.

[23] M. Li, B. Yu, C. Huang, and Y.H. Song, "Service Matchmaking with Rough Sets," *Proc. Sixth IEEE Int'l Symp. Cluster Computing and the Grid (CCGrid '06),* pp. 23-30, May 2006.

[24] B. Yu, W. Guo, M. Li, Y.H. Song, P. Hobson, and M. Qi, "Service Matchmaking and Discovery with Rough Sets," *Proc. Second Int'l Conf. Semantics, Knowledge and Grid (SKG '06),* p. 80, Nov. 2006.

[25] Z. Pawlak, "Rough Sets," *Int'l J. Computer and Information Science,* vol. 11, no. 5, pp. 341-356, 1982.

[26] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.,* vol. 30, no. 5, pp. 311-327, May 2004.

[27] V. Haarslev and R. Möller, "Description of the RACER System and Its Applications," *Proc. Int'l Workshop Description Logics (DL '01),* Aug. 2001.

[28] A. Skowron and C. Rauszer, "The Discernibility Matrices and Functions in Information Systems," *Decision Support by Experience—Application of the Rough Sets Theory*, R. Slowinski, ed. Kluwer Academic Publishers, pp. 331-362, 1992.
[29] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE Trans. Networks*, vol. 11, no. 1, pp. 17-32, 2003.
[30] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. 18th IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01)*, pp. 329-350, Nov. 2001.
[31] J.M. Schopf, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D'Arcy, and A. Chervenak, "Monitoring the Grid with the Globus Toolkit MDS4," *J. Physics: Conf. Series*, vol. 46, pp. 521-525, 2006.
[32] A.W. Cooke et al., "The Relational Grid Monitoring Architecture: Mediating Information about the Grid," *J. Grid Computing*, vol. 2, no. 4, pp. 323-339, 2004.
[33] W. Fang, S. Miles, and L. Moreau, "Performance Analysis of a Semantics-Enabled Service Registry," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 3, pp. 207-223, 2008.
[34] N. Srinivasan, M. Paolucci, and K.P. Sycara, "An Efficient Algorithm for OWL-S Based Semantic Search in UDDI," *Proc. First Int'l Workshop Semantic Web Services and Web Process Composition (SWSWPC '04)*, pp. 96-110, July 2004.
[35] N. Zhong and A. Skowron, "A Rough Set Based Knowledge Discovery Process," *Int'l J. Applied Math. and Computer Science*, vol. 11, no. 3, pp. 603-619, 2001.
[36] C. Caceres, A. Fernandez, S. Ossowski, and M. Vasirani, "Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach," *IEEE Intelligent Systems*, vol. 21, no. 6, pp. 11-20, 2006.
[37] V. Tsetsos, C. Anagnostopoulos, and S. Hadjiefthymiades, "On the Evaluation of Semantic Web Service Matchmaking Systems," *Proc. Fourth IEEE European Conf. Web Services (ECOWS '06)*, pp. 255-264, Dec. 2006.
[38] D.A. Buell and D.H. Kraft, "Performance Measurement in a Fuzzy Retrieval Environment," *Proc. ACM SIGIR '81*, pp. 56-62, 1981.
[39] C. van Rijsbergen, *Information Retrieval*. Butterworths, 1979.
[40] K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park, *QoS for Web Services: Requirements and Possible Approaches*, World Wide Web Consortium (W3C) Working Group Note 25, 2003.
[41] W. Smith, I. Foster, and V. Taylor, "Predicting Application Run Times Using Historical Information," *Proc. Fourth Workshop Job Scheduling Strategies for Parallel Processing*, pp. 122-142, 1998.
[42] L. Gong, X. Sun, and E.F. Watson, "Performance Modeling and Prediction of Nondedicated Network Computing," *IEEE Trans. Computers*, vol. 51, no. 9, pp. 1041-1055, Sept. 2002.
[43] L. Qi, H. Jin, I.T. Foster, and J. Gawor, "HAND: Highly Available Dynamic Deployment Infrastructure for Globus Toolkit 4," *Proc. 15th Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing (PDP '07)*, pp. 155-162, 2007.
[44] N.F. Noy, "Semantic Integration: A Survey of Ontology-Based Approaches," *ACM SIGMOD Record*, vol. 33, no. 4, pp. 65-70, 2004.
[45] B. Kryza, R. Slota, M. Majewska, J. Pieczykolan, and J. Kitowski, "Grid Organizational Memory—Provision of a High-Level Grid Abstraction Layer Supported by Ontology Alignment," *Future Generation Computer Systems*, vol. 23, no. 3, pp. 348-358, 2007.
[46] R. Pan, Z. Ding, Y. Yu, and Y. Peng, "A Bayesian Network Approach to Ontology Mapping," *Proc. Int'l Semantic Web Conf. (ISWC '05)*, pp. 563-577, 2005.
[47] I. Taylor, M. Shields, I. Wang, and A. Harrison, "Visual Grid Workflow in Triana," *J. Grid Computing*, vol. 3, nos. 3-4, pp. 153-169, 2005.

Maozhen Li received the PhD degree from the Institute of Software, Chinese Academy of Sciences, Beijing, in 1997. He is a lecturer in the School of Engineering and Design, Brunel University. His research interests are in the areas of grid computing, distributed problem-solving environments for large-scale simulations, intelligent systems, service-oriented computing, and semantic Web. He has more than 50 publications in these areas. He coauthored *The Grid: Core Technologies*, a research-level textbook on grid computing published by John Wiley & Sons in 2005. He is on the editorial boards of the *Encyclopedia of Grid Computing Technologies and Applications* and the *International Journal of Grid and High Performance Computing*. He has been serving as a TPC member for various conferences in the area of grid computing, e.g., IEEE CCGrid 2005, CCGrid 2006, CCGrid 2007, CCGrid 2008, IEEE SKG 2005, SKG 2006, SKG 2007, and IEEE CSE 2008. He is a member of the IEEE.

Bin Yu received the PhD degree from the School of Engineering and Design, Brunel University, in April 2007. He is currently a system analyst at Levele Ltd., Edinburgh. His research interests are in the areas of service-oriented computing, grid computing and applications, service discovery, and composition optimization.

Omer Rana received the PhD degree in neural computing and parallel architectures from the Imperial College of Science, Technology, and Medicine, London University. He is a professor of performance engineering in the School of Computer Science and the deputy director of the Welsh eScience Center at Cardiff University, United Kingdom. His research interests include high-performance distributed computing, multiagent systems, and data mining.

Zidong Wang received the PhD degree in electrical and computer engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1994. He is a professor of dynamical systems and computing at Brunel University. His research interests include dynamical systems, signal processing, bioinformatics, and control theory and applications. He has published more than 80 papers in refereed international journals. He is currently serving as an associate editor for the *IEEE Transactions on Automatic Control*, the *IEEE Transactions on Signal Processing*, the *IEEE Transactions on Systems, Man, and Cybernetics—Part C*, the *IEEE Transactions on Control Systems Technology*, and *Circuits, Systems & Signal Processing*, an action editor for *Neural Networks*, an editorial board member for the *International Journal of Systems Science*, *Neurocomputing*, the *International Journal of Computer Mathematics*, and the *International Journal of General Systems*, and an associate editor on the conference editorial board for the IEEE Control Systems Society. He is a senior member of the IEEE, a fellow of the Royal Statistical Society, a member of the program committee for many international conferences, and a very active reviewer for many international journals. He was nominated as an appreciated reviewer for *IEEE Transactions on Signal Processing* in 2006 and an outstanding reviewer for *IEEE Transactions on Automatic Control* in 2004 and for the journal *Automatica* in 2000.