An Efficient Service Discovery Method and its Application

Shuiguang Deng, Zhejiang University, China
Zhaohui Wu, Zhejiang University, China
Jian Wu, Zhejiang University, China

ABSTRACT

To discover services efficiently has been regarded as one of important issues in the area of Service Oriented Computing (SOC). This article carries out a survey on the issue and points out the problems for the current semantic-based service discovery approaches. After that, an information model for registered services is proposed. Based on the model, it brings forward a two-phase semantic-based service discovery method which supports both the operation matchmaking and operation-composition matchmaking. The authors import the bipartite graph matching to improve the efficiency of matchmaking. An implementation of the proposed method is presented. A series of experiments show that the method gains better performance on both discovery recall rate and precision than a traditional matchmaker and it also scales well with the number of services being accessed.

Keywords: Bipartite Graph Matching, Service Discovery, Service Matchmaking, Web Service

INTRODUCTION

Background

Nowadays, as the Internet has become the main platform on which enterprises carry out businesses globally, the environment of enterprise applications will be characterized by frequently changing market demands, time-to-market pressure and fierce competition. Therefore, it requires that the enterprise business systems should provide more flexibility than present-day systems can afford. The key to tackle this challenge completely is to utilize a kind of novel software system architecture which is required to be distributed, loose-coupled and reconfigurable. Fortunately, these requirements can be best addressed by Service-oriented Architecture (SOA).

SOA is an architectural style whose goal is to achieve loose coupling among interacting services and to build software systems by composing services (Papazoglou & Georgakopoulos, 2003). It provides greater flexibility and agility while allowing business systems to use heterogeneous resources efficiently and effectively. Web services technology has been regarded as the preferred implementation
vehicle for SOA. A Web service is a software entity that supports interoperable application-to-application interaction over Internet. At present, the accelerating creation and use of Web Services in enterprises informatics is a major trend (Kalogeras, Gialelis et al., 2006). Thus, more and more Web services are published in Internet by enterprises to accelerate the cooperation with their partners. For example, in the scenario of supply chain management, a manufacturer receives an order to deliver some merchandise to a retailer. In order to accomplish this business, the manufacture finds possible suppliers and selects the best available service provided by one supplier. However, due to the highly distributed and dynamic environment, Web services may be located at different enterprises and come and leave at any time without prior warning. In that context, no one is likely to have the detailed knowledge of all Web services in advance. As a result, one of great challenges is how to discover the suitable Web services accurately and quickly. Thus, service discovery, which aims at retrieving services advertised in a repository that match a user’s goal, has allured much attention both from industry and academy.

**Problem**

Currently, there is a good body of work on service discovery. Among the work, the effort of semantic Web service from the semantic Web community has been regarded as the most promising way to retrieve services in an accurate and automatic way. Based on related ontology languages and inference engines, semantic Web services provide machine understandable descriptions of what services do and how they achieve their goals (McIlraith, Son et al., 2001). Semantic Web service discovery utilizes semantic matchmaking to check whether an advertised service satisfies a user’s request by computing the similarity degree between the description for the service and the one for user’s request. If the similarity degree exceeds some threshold value specified by the user, the service is returned as a candidate for the user. Due to the accurate and unambiguous description of a service’s functionalities and a user’s request both enhanced by semantics, semantic Web service discovery tends to get good recall rate and precision. However, they can achieve even better performance with the following two factors taken into consideration.

(1) Not all inputs are compulsory for each output

According to the most frequently cited semantic Web service matchmaking algorithm proposed by Paolucci, Kawamura et al. (2002), an advertised service matches a request if the request provides all the inputs (possibly more) needed by the service while the service generates all the outputs (possibly more) needed by the requester. In other words, a successful matching demands that the request provides all the inputs of the advertised service to get any output of the service. This requirement has been widely accepted by most semantic Web service matchmaking methods.

However, the successful matching criteria are too strict and may lead to some unwanted situations. Consider an abstract scenario between an advertised service $S$ and a request $R$, where $S$ has two inputs ($a$ and $b$) and two outputs ($o$ and $p$), and $R$ specifies one input ($a$) and one output ($o$)$. According to the above successful matching criteria, $S$ does not match $R$ as $R$ cannot provide the input $b$. But for $S$, maybe the input $b$ is optional for the output $o$. In this case, $S$ should be a candidate service for $R$ but it is excluded. Consider a real weather report service (http://www.webservicex.com/global-weather.asmx?WSDL). It has an operation named GetWeather that returns WeatherResult on receiving a CityName and a CountryName. However, this operation also serves well if the inputs only include a CityName. Both the abstract and real cases show that to some extent, the recall rate can be improved with the interface dependency information (i.e., the dependency of outputs on inputs) considered.
(2) Operation compositions provide value-added functions

According to WSDL (Web Service Definition Language), the defacto standard service description language, a service is a collection of operations and each provides a function. For example, a stock service can provide both the stock query and exchange functions. At present, there are a large number of services described in WSDL and published in local or global repositories on Internet. We have implemented a program to query and download the WSDL files on Internet automatically by network programming with Google APIs. After collecting some statistics, we have two findings: One is that about 75% WSDL files contain more than two operations, about 30% contain more than five operations and about 5% contain more than ten operations; another significant finding is that there are more than 30% WSDL files, in which some outputs of an operation are the same as some inputs of others. For example, the service (http://www.ripedev.com/webservices/ZipCode.asmx?WSDL) has two different operations named CityToZipCode and ZipCodeToTimeZone, respectively. The former translates from a CityName to its ZipCode and the latter from a ZipCode to its corresponding TimeZone. As the output ZipCode of the first operation is the same as the input of the second one, it implies that the service has the function to translate indirectly from a CityName to its TimeZone through the concatenation of the two operations. This example shows that the operations within a service may be concatenated to provide value-added functions.

However, for the most current service discovery methods, they all regard a service as an operation or several isolated ones. While matchmaking between an advertised service and a request, they simply check whether there is such an operation that offers the requested function. If such an operation exists, the service will be selected as a candidate one; otherwise, it will be filtered. Accordingly, some deserved services that actually match the request by a composition of several operations are excluded. Thus, taking the composition of operations within a service into matchmaking can improve the recall rate.

Research Objective and Methodology

The above two factors has been discussed in our two pre-published papers (Deng, Wu et al., 2006a & 2006b), respectively. However, the service matchmaking algorithms proposed in these two papers are with high computing complexity. And moreover, these two factors are not taken into consideration together before. Thus, in this article we consider the above two factors in one discovery method and propose an efficient two-phase semantic service discovery mechanism (TSSD). Given an advertised service and a request, TSSD checks whether there is a single operation matching the request at the first phase, where the semantic matchmaking is carried out between each operation and the request, and the interface dependency of an operation is also considered. If no single operation matches the request, it performs operation-composition matchmaking to check whether there is such a composition of operations matching the request. Compared with our pre-published work, we import the concept of bipartite graph matching and proposed a new matchmaking algorithm based on it to improve the matchmaking efficiency. A series of evaluations in our implementation suggest that considering these two factors in TSSD offers better performance and TSSD scales well with the number of services being accessed.

The article is organized as follows. In the second section of this article, we give a literature survey on service discovery. Then, an information model for registered services is introduced in the third section. In the fourth section, we present the two-phase service discovery method. The implementation of TSSD and A series of evaluations are given in the fifth section. Finally, we conclude this article and give the future direction in the sixth section.
LITERATURE SURVEY

Service discovery is an active research area. It originates from the issue of software component reuse and discovery in software engineering, distributed computing and information retrieval in knowledge and data engineering. During the past few years, much effort was placed in the area of service discovery. In this section, we discuss some representative semantic-based approaches and classify them into the following categories according to the semantic service models used in discovery. Notice that besides OWL-S/DAML-S (Ontology Web Language for Service/ DARPA Agent Markup Language for Service) and WSMO (Web Service Modeling Ontology), WSDL-S and SWSO (Semantic Web Service Ontology) are other two semantic service models. However, there is little literature on service discovery based on these two models at present.

OWL-S/DAML-S based Semantic Service Discovery

OWL-S (formerly DAML-S) is an OWL-based Web service ontology. As the first effort towards semantic Web service, it provides a core set of concepts for describing the properties and capabilities of a Web service in an unambiguous and computer-interpretable way. It describes a service through three main parts: the service profile for advertising and discovering services; the process model, which gives details on how the service works; and the grounding, which provides the description on how to access the service.

Based on the service profile of DAML-S, Paolucci et al. (2002) firstly proposed an approach for semantic matchmaking of Web service capability. It considered the matching of input/output concepts defined in an ontology. After that, they augmented the UDDI (Universal Description, Discovery, and Integration) registry with a new module (Semantic Services Matchmaker), a search engine for Web services that enhanced the discovery facilities of UDDI to make use of semantic information described in a service profile (Kawamura, Blasio et al. 2003; Srinivasan, Paolucci et al., 2004). Luo et al. (2006) proposed a method to enable the syntax-based UDDI search engine behavior like a semantic matchmaker without requiring modification in the UDDI registries. During the service publication, they transformed OWL-S service profiles into a UDDI data model, resolved ontology concepts and indexed them. So UDDI registries performed semantic matchmaking like querying on UDDI data.

Due to the process model of OWL-S providing a far richer description of a service than the service profile, some researchers also utilized the process model in service matchmaking. Bansal and Vidal (2003) stored an OWL-S process model as a tree, in which the root of the model was the root of the tree; the composite processes were intermediary nodes; and atomic processes were leaves. They proposed a service matchmaking algorithm based on tree matching. As an improvement on Bansal and Vidal’s work, Brogi and Corfini (2007) proposed a method named SAM (Service Aggregation Matchmaking) that determined whether a query could be satisfied by a service by the analysis of a dependency graph constructed according to the process model of the service. Unlike Bansal and Vidal’s algorithm performing matchmaking at the level of entire services, SAM supported the matching both at the level of atomic processes and at the level of composition of their composition.

Besides the service profile and process model used in service discovery and matchmaking, the underlying logic basis of OWL-S—Description Logic (DL) had also attracted researchers’ attention. Benatallah et al. (2005) proposed an approach in the context of description logics to automate the Web service discovery. They transformed a service discovery as the best covering problem and then formalized the problem in the framework of DL-based ontologies. Then, they presented a service matchmaking algorithm using a hypergraph-based algorithm to compute the best cover.
WSMO Based Semantic Service Discovery

WSMO is a conceptual model for semantically describing Web services and their specific properties. It depicts a semantic Web service through four elements: ontologies which provide the terminology for other elements; Web services which define a semantic description of services; goals which specify the users’ requirements for a Web service; and mediators which resolve the heterogeneity problem (Roman, Lausen et al., 2005).

Keller et al. (2004) gave an in-depth analysis of the major conceptual issues involved in Web service discovery and proposed a conceptual model for the discovery in WSMO. They differentiated between service and Web service discovery and performed Web service discovery based on matching abstracted goal descriptions with semantic annotations of Web services. They utilized the set-based modeling approach to model Web services and goals; and based on set-theoretic criteria, proposed two conceptual semantic-based discovery approaches, namely discovery based on simple semantic descriptions of services and discovery based on rich semantic descriptions of services. Both of them differentiated between different types of matches such as exact-match, subsumes-match, plug-in-match and intersection match. Kifer et al. (2004) presented a logical framework to dynamically discover Web services in two stages, namely discovery and contracting, in the processes of searching for an appropriate service. They defined proof obligations that formalized the concepts of a match in these two stages based on the WSMO conceptual model. Stollberg et al. (2005) presented a partner and service discovery approach in their system for automated collaboration on the semantic Web by integrating agents, ontologies and Web services. They utilized WSML and the logic expression to model the action knowledge and object knowledge of a Web service, respectively; and then performed service discovery by the set-based object matchmaking using a theorem prover.

Miscellaneous Semantic Service Discovery Approaches

Besides the above two categories of semantic-based service discovery methods, there are many others. For example, Klein and Bernstein (2004) used process models to capture service semantics and proposed a pattern-matching algorithm to retrieve target services. Syeda-Mahmood et al. (2005) used domain-independent and domain-specific ontologies to find matching service descriptions. They combined the two cues to determine an overall semantic similarity score. In addition, they integrated semantic and ontological matching with attribute hashing to accelerate service retrieval. Osman et al. (2006) utilized the case based reasoning (CBR) methodology for modeling dynamic Web service discovery and matchmaking. In order to improve the accuracy of service discovery and matchmaking, they took into account the past matchmaking experiences and used ontology to describe both services and rules of CBR reasoning engine.

After a deep insight into the above representative methods, we find that they did not consider the interface dependency relation in a Web service while doing matchmaking. This is due to the absence of some information to describe interface dependency relations in a Web service registered in a registry. Moreover, most of them did not support the matchmaking at the operation composition level. Although there exist a few composition-oriented discovery approaches composing services on the fly for users (Aversano, Canfora et al., 2004; Liang, Chakarapani et al., 2004), they belong to the coarse-grain composition rather than fine-grain, namely operation-oriented composition as in our approach. Furthermore, compared with those approaches supporting service-oriented composition, our approach achieves efficiency by pre-computing off-line all the compositions for the later online matchmaking.
INFORMATION MODEL FOR SERVICE REGISTRATION

To enable Web services provided by different enterprises to be findable, accessible and reusable, they must be published and registered in a service repository. In fact, service discovery is carried out within such repositories by performing service matchmaking between a user’s request and the registration information of services. As a de-facto standard registry for Web services, UDDI make use of a data structure tModel to represent the registered information of a service. However, Due to the fact that UDDI preserves the syntax information of services only, such as names, comments and descriptions, service discovery is key-word based and does not yield a satisfactory performance. To address this issue completely and also to make the aforementioned ignored factors be considered in service discovery, it is a key step to enrich the information for registered services in a repository. Thus, we extend the standard registered information of a service in UDDI in the following three aspects.

(1) Annotate inputs/outputs with semantic information

Semantic information can help machine understand the capability of services accurately and explicitly. This information can be gained by requesting a service provider to annotate each input/output with a concept (also called term or word) from a public domain-independent ontology such as WordNet (Miller, 1995) and HowNet (Dong & Dong, 2006) when he/she registers a service. For example, when the service provider advertises the aforementioned order-management service, he/she is required to specify a concept from a public ontology for each input message and output message. This extension enhances operations with semantics and enables matchmaking at the semantic level.

Notice that we do not use a domain-ontology for service annotation here. This is because of the following three reasons: 1) different services in a repository come from different domains. For many domains, no public ontology is available at present. It is a difficult and time-consuming project to establish a new and widely accepted one; 2) Multiple domain-ontology used by services brings much complexity to semantic service discovery, due to many open issues in the ontology research area such as ontology integration, ontology mapping and semantic similarity computing (Rodriguez & Egenhofer, 2003); 3) the design of public ontologies such as WordNet and HowNet is inspired by current psycholinguistic and computational theories of human lexical memory. They have enough concepts to annotate various services from different domains.

(2) Add interface dependencies between inputs and outputs

For each output of an operation, its interface dependency is used to find out inputs, on which the output depends on. Interface dependency can be declared by service providers who can specify the optional inputs for each output in an operation. For example, “CountryName” is required to be declared as optional for “WeatherResult” by the provider of the service mentioned before. The declaration indicates that WeatherResult depends on CityName only. This extension makes it possible to consider interface dependency in matchmaking.

(3) Add assignment relation into an advertised service

An assignment relation between an output of an operation and an input of another one indicates that the output can be fed into the input both at the syntactic and semantic level. As the inputs/outputs have been annotated with ontology concepts, assignment relations can be automatically generated by the semantic inference among concepts. In order to reduce the semantic differentia in an assignment, one ontology concept can be assigned to another only if both of them are the same or the former
is a subclass of the latter. For an input, we can get outputs which can be fed to it directly using assignment relations. Thus, adding assignment relations into a service in advance can accelerate the process of composing operations to fulfill a request.

We present the formal definition for an operation and service as follows after the above three extensions are added. Notice that, we do not aim to propose a new service model different from existing models such as WSDL, OWL-S and WSMO. In fact, the information model for service registration can be regarded as a common abstraction from those service models which represents the necessary information in a repository used by service discovery.

**Definition (Operation)** An operation $p$ is a 4-tuple: $p = \{n_p, I, O, f_p\}$ where:

1. $n_p$ is the operation name.
2. $I = \{i_1, i_2, ..., i_n\}$ is the set of input names.
3. $O = \{o_1, o_2, ..., o_m\}$ is the set of output names.
4. $f_p : O \rightarrow P(I)$ is a mapping from the set $O$ to the power set of $I$.

Notice that any element in $I$ or $O$ corresponds to a concept in a public domain-independent ontology. For an output $o \in O$, $f_p(o) = I$ means that $o$ depends on the input set $I$ (denoted as $o \propto I$), i.e., all the inputs in $I$ must be provided for an invocation of this operation to get $o$.

**Definition (Fully-dependent/Partially-dependent Output)** Given an operation $p = \{n_p, I, O, f_p\}$ and $o \in O$, if $o \propto I$, i.e., $f_p(o) = I$, $o$ is a fully-dependent output; otherwise, i.e., $f_p(o) \subset I$, $o$ is a partially-dependent output.

This definition indicates that a fully-dependent output depends on all the inputs whereas a partially-dependent output depends on a part of the inputs.

**Definition (Service)** A service $s$ is a 3-tuple: $s = (n_s, P, f_s)$ where:

1. $n_s$ is the service name.
2. $P = \{p_1, p_2, ..., p_s\}$ is the set of operations.
3. $f_s : \bigcup_{i \in s} p_i.O \rightarrow P(\bigcup_{i \in s} p_i.I)$ is the mapping from all outputs to the power set of all inputs.

Notice that the service definition only includes the necessary information to be used for service matchmaking but ignores others, such as the information about the service provider and the service binding. It shows that a service is a collection of operations. For any output $o$ of an operation, the function $f_s$ returns those inputs of others that can be assigned with $o$. For example, three operations $p_i (i = 1, 2, 3)$ of a service, $p_1.O(o_1, o_2)$, $p_2.J(i_2, i_3)$ and $p_3.J(i_3, i_5, i_4)$, the two assignment relations $f_s(o_1) = \{i_2, i_3\}$ and $f_s(o_2) = \{i_1, i_4\}$ mean that $o_1$ can be assigned to $i_2$ and $i_3$, while $o_2$ can be assigned to $i_1$ and $i_4$.

Consider the Geography Information Service shown in Figure 1. There are three operations named GetTimeInfo, GetWeather and GetCapital, respectively. Within the operations, a dashed line directed from an output to an input represents an o-i Dependency Relation, i.e., the input is compulsory for the invocation of the operation to get the output. A double dots line directed form an output to an input between two operations represents an o-i Assignment Relation, i.e., the output can be fed into the input.

According to the definitions of an operation and a service, we formalize the above service as follows.
Figure 1. An geography information service extended with interface dependency and assignment relations

TWO-PHASE SERVICE DISCOVERY

For a service request and a registered service, a semantic service discovery method carries out service matchmakings to evaluate to what extent the service’s description satisfies the request’s description by calculating the Semantic-Similarity Degree (SSD) between them. If the degree exceeds the threshold value specified by the requester, for example 80%, the service is returned. The higher the degree, the more the service matches the request. In general, a similarity degree ranges from “0” to “1”. The value “1” means that the service matches the request completely whereas “0” means that the service doesn’t match the request at all. A value between “0” and “1” means the service matches the request partially. While computing the SSD, a semantic matchmaking method utilizes the semantic information in the descriptions of a service and a request, for example, the ontology concepts annotated with inputs and outputs.

Semantic Similarity Computing Between Concepts

The study of semantic similarity between ontology concepts has been a generic issue in the areas of natural language processing and information retrieval for many years. At present, a number of semantic similarity computing methods have been proposed and proven to be practicable in some specific applications. These methods can be classified into two categories: edge counting-based (or dictionary/thesaurus-based) methods and information theory-based (or corpus-based) methods (Li, Bandar et al., 2003).

The methods in the first category calculate the similarity between two concepts using the
shortest path between concepts in the hierarchical structure of the shared ontology. In general, the longer the shortest path between two concepts is, the smaller the similarity between them. Besides shortest path, some methods also consider the depth of the two concepts in the hierarchical structure and the density of the sub-hierarchies of two concepts in the hierarchical structure. Methods in this category are intuitive, intelligible and easy to be implemented.

The methods in the second category calculate the similarity of two concepts by the amount of information shared by the two concepts. In general, the more information shared, the greater the similarity between them. The shared information can be defined as the maximum of the information content of the concept that subsumes the two concepts in the taxonomy hierarchy. And the information content of a concept can be evaluated by the probability of encountering an instance of the concept in a corpus (Resnik, 1999). As the information content is calculated from an application-dependent corpus, the methods in this category are not general and do not adapt to a domain-independent ontology well. Moreover, due to its reliance on the statistic on the occurrence frequency of a concept and its sub-concept in a generous corpus, the methods in this category, compared to those in the first category, are more complex and difficult to be implemented.

Thus, we prefer the methods of the first category. Since the method proposed by Li et al. (2003) considers both the shortest path and the depth and outperforms other methods significantly, we select it for the semantic similarity computing between two ontology concepts. The method calculates the similarity between two concepts $C_1$ and $C_2$ (denoted as $\text{SimCC}(C_1, C_2)$) according to formula (1).

$$\text{SimCC}(C_1, C_2) = \begin{cases} e^{-\alpha l} \cdot e^{\beta h} - e^{-\beta h}, & \text{if } (C_1 \neq C_2) \\ e^{\beta h} + e^{-\beta h}, & \text{if } (C_1 = C_2) \end{cases}$$

where $l$ is the length of the shortest path between them; $h$ is the depth of the closest ancestor of both $C_1$ and $C_2$ in the hierarchical structure; and $\alpha, \beta \geq 0$ are parameters scaling the contribution of $l$ and $h$, respectively. It shows that the similarity between two concepts is monotonically decreasing with respect to $l$ and monotonically increasing with respect to $h$. According to the experiments proposed by Li et al., $\alpha = 0.2$ and $\beta = 0.6$ bring the optimal effect and they are used in this work.

Consider the similarity between two concepts (StationWagon and Train) in the fragment of the semantic hierarchy of WordNet shown in Figure 2, the shortest path between them is StationWagon-Car-Transportation-Train and the closest ancestor for both StationWagon and Train is Transportation. Thus $l = 3$, $h = 2$ and $\text{SimCC}(\text{StationWagon}, \text{Train}) = e^{-0.2 \times 3} \cdot e^{0.6 \times 2} - e^{-0.6 \times 2} = 0.64$.

First Phase: Discovery Based on Operation Matchmaking

As different functions of a service are provided by its operations indeed, in TSSD, whether a service satisfies a request is whether there is such an operation or a composition of operations matching the request. In this section, we introduce how TSSD discovers services based on operation matchmaking.

Definition (Service Request) A service request $r$ is a 3-tuple: $r = (F, O', \omega)$ where:

1. $F$ is the set of inputs that a requester can provide to invoke a target service.
2. $O'$ is the set of outputs that the requester desires to get from the invocation of the service.
3. $0 < \omega \leq 1$ is the threshold value specified by the requester to show to what extent the target service should match the request.

Notice that, to simplify the problem at this stage so as to focus on the key points of
the article, here we only concern the input/output requirements in a request description but ignore non-functional requirements such as QoS at present. All the inputs and outputs in a request are also annotated by a requester with ontology concepts in order to ensure the request to be also semantic-enhanced.

**Definition (Matched-Operation)** For an operation \( p \) and a request \( r = (I_r, O_r, \omega) \), \( p \) is a Matched-Operation for \( r \) if \( \text{SimPR}(p, r) \geq \omega \), where \( \text{SimPR}(P, R) \) is the SSD between \( p \) and \( r \).

**Definition (Best-Matched-Operation)** For a service \( s \), one of its operation \( p \) and a request \( r = (I_r, O_r, \omega) \), \( p \) is a Best-Matched-Operation for \( r \) if \( p \) is a Matched-Operation and \( \forall p_i \in P, \text{SimPR}(p_i, r) \geq \text{SimPR}(p, r) \).

For a service request \( r = (I_r, O_r, \omega) \) and an operation \( p = \{n_p, I, O, f_p\} \), the goal of TSSD in the first phase is to find the Best-Matched-Operation. The algorithm DBMO shown below accepts a service and a request as inputs and returns the Best-Matched-Operations for the request. It computes the SSD represented by \( \text{Sim}(o, r) \) between each operation and the request and chooses the operations with the highest SSD. Then it judges whether the SSD of the selected operations is higher than the threshold value \( \omega \). If it is, the selected operations are returned as the Best-Matched-Operations; otherwise there is no operations matching the request (see Table 1).

For an operation \( p = \{n_p, I, O, f_p\} \) and a request \( r = (I_r, O_r, \omega) \), \( \text{Sim}(p, r) \) is calculated through the algorithm \( \text{calSim} \) shown in below.

1. Judge whether \( |O_r| \leq |O| \) holds, i.e., check whether \( p \) can provide all the requested outputs. If it can, \( \text{calSim} \) goes on to the next step; otherwise, it terminates and returns \( \text{SimPR}(p, r) = 0 \).
2. Find an injection \( f \) from \( O_r \) to \( O \), i.e., \( f : O_r \rightarrow O \), that makes \( \sum_{o' \in O'} \text{SimCC}(o', f(o')) \) reach its max. In fact, the goal of this step is to find a different matched output in \( O \) for each one in \( O_r \) and to ensure the sum of pairwise similarity to be the max. We denote the range of \( f \) as \( O' \), i.e., \( O' = \bigcup f(o') \).
3. Determine the input set that must be provided to invoke \( p \) to get all the outputs in \( O' \). The result set is \( I' = \bigcup_{o \in O'} f_p(o) \) according to the semantic hierarchy of WordNet.
Table 1. Algorithm: Discovery of best-matched-operations (DBMO)

| Input: A service \( s = (n, P, f) \) and a service request \( r = (F, O', \omega) \) |
| Output: Best-Matched-Operations or null |

Initiate \( MaxSim = 0 \) and \( SOP = \emptyset \);
for each operation \( op \in P \} \)
\[
\text{if}(\text{sim}(op, r) = \text{MaxSim}) \{
\text{sop}.\text{insert}(op);
\}
\text{else if}(\text{sim}(op, r) > \text{MaxSim}) \{
\text{sop}.\text{clear}();
\text{sop}.\text{insert}(op);
\text{MaxSim} = \text{sim}(op, r);
\}
\}
\text{return } (\text{MaxSim} \geq \omega) ? \text{sop}: \text{null};

...to the interface dependency. Go on to the next step.

(4) Judge whether \( |F| \leq |F'| \) holds, i.e., check whether \( F \) can provide all the necessary inputs. If \( F \) can, \( \text{calSim} \) goes on to the next step; otherwise, it terminates and returns \( \text{SimPR}(p, r) \).

(5) Find an injection \( g \) from \( F' \) to \( F \), i.e., \( g: F' \rightarrow F \), that makes \( \sum_{i=1}^{n} \text{SimCC}(x_i, y_i) \) reach its max. In fact, the goal of this step is to find a different matched input in \( F \) for each one in \( F' \) and to ensure the sum of pairwise similarity to be the max.

(6) Calculate \( \text{sim}(p, r) \) through the following formula.

\[
\text{sim}(p, r) = \sum_{x \in X} \left( \sum_{y \in Y} \text{SimCC}(x, f(x)) \right) \sum_{\text{sim}(i, g(i)) \neq 0} \text{SimCC}(i, g(i)) / |Y|
\]

In fact, we can model the given conditions as a weighted bipartite graph \( G = (X, Y, E) \) where \( X \) and \( Y \) are two sets of vertices corresponding to the given two sets of ontology concepts, respectively, and \( E \) is the set of weighted edges constructed according to the following rule. For \( \forall x_i(1 \leq i \leq n), y_j(1 \leq j \leq m) \) if \( \text{SimCC}(x_i, y_j) > 0 \), we draw an edge \( <x_i, y_j> \) with a weight \( W_{x_i, y_j} = \text{SimCC}(x_i, y_j) \). After that, the problem is transformed to find an optimal matching from \( X \) to \( Y \) in \( G \) and ensure that the matching covers all the vertices in \( X \). Thus, the optimal matching is the target injection \( f: X \rightarrow Y \). We design a method named \( \text{IOMatch} \) based on the Kuhn-Munkres algorithm (Lovasz and Plummer, 1986) to find such a matching. Due to the constraint \( n \leq m \), i.e., \( |X| \leq |Y| \), specified in the problem, \( \text{IOMatch} \) considers the following two cases.

1) \( |X| = |Y| \). In this case, we use the Kuhn-Munkres algorithm directly to find an optimal matching \( M \) for \( G \). According to the algorithm, \( M \) is a perfect matching in \( G \), i.e., \( M \) covers all the vertices in \( X \) and \( Y \). Thus \( M \) is the target matching.

2) \( |X| < |Y| \). In this case, we change \( G \) to \( G' \) by adding \( |Y| - |X| \) vertices into \( X \). Then, between each new added vertex and each vertex in \( Y \), add a new edge with a weight 0. If the new added vertices set is denoted as \( V' = \{v_1, v_2, ..., v_k\}, k = |Y| - |X| \), we get...
The algorithm is shown in Figure 4. It uses the Kuhn-Munkres algorithm to find an optimal matching \( M \) for \( G \). Then we change \( M \) to \( M' \) by eliminating those edges which cover vertices in \( F \). Because the sum of weight of the eliminated edges equals 0, \( M' \) is an optimal matching for \( G' \) and \( G \). Due to \( M' \) covering all vertices in \( X, M' \) is the target matching.

As the complexity of the Kuhn-Munkres algorithm is \( O(n^3) \), where \( n = |Y| = |X| \), the complexity of \( calSim \) is \( O(2m^3) \) where \( m = |Y| \) and that of algorithm \( DBMO \) is \( O(p \times 2q^3) \), also a polynomial complexity, where \( p \) is the number of the operations in a service and \( q \) is the biggest number of outputs of an operation. According to the aforementioned statistic result in Introduction, only 5% services contain more than ten operations. So the value of \( q \) is always less than 10. Moreover, \( p \) and \( m \) are always less than 5. Thus \( calSim \) and \( DBMO \) have an acceptable time complexity.

To introduce the bipartite graph matching and the Kuhn-Munkres algorithm is beyond the scope of this article. Here we show how \( calSim \) calculates \( sim(p, r) \) using the example illustrated in Figure 3, where the operation \( p \) has three outputs and two inputs, and the request \( r \) specifies two inputs and two outputs. The value on each connection between \( O \) and \( O' \) \((1 \leq i \leq 3, 1 \leq j \leq 2) \), \( I \) and \( I' \) \((1 \leq i, j \leq 2) \) denotes the semantic similarity between the two concepts connected. Below is the process for the calculation of \( sim(p, r) \) according to \( calSim \).

(1) As \( (|O'| = 2 \land |O| = 3) \Rightarrow |O'| < |O|, calSim \) continues.

(2) Use \( IOMatch \) to find an injection \( f: O' \rightarrow O \). We model the problem into the optimal matching problem from \( O' \) to \( O \) in \( G = (X, Y, E) \) shown in the left of Figure 4 where \( X = \{o'_1, o'_2, o'_3\}, Y = \{o_1, o_2, o_3\} \) and \( E = \{o'_1 \land o_1, o'_1 \land o_2, o'_1 \land o_3, o'_2 \land o_1, o'_2 \land o_2, o'_2 \land o_3, o'_3 \land o_1, o'_3 \land o_2, o'_3 \land o_3\} \). Since \( |X| < |Y| \), \( IOMatch \) changes \( G \) to \( G' \) by adding one vertex named \( v \) into \( X \) and three 0-weighted edges \( <v, o_1>, <v, o_2>, <v, o_3> \) and \( <v, o_1> \) into \( E \) as shown in the right of Figure 4. Then it uses the Kuhn-Munkres algorithm to find an optimal matching \( M \) for \( G' \). Thus, we get \( M = \{<o'_1, o_1>, <o'_2, o_2>, <o'_3, o_3>, <v, o_1>, <v, o_2>, <v, o_3>\} \) and the sum of the weight is \( 0.8 + 0.7 + 0 = 1.5 \). Then \( IOMatch \) changes \( M \) to \( M' = \{<o'_1, o_1>, <o'_2, o_2>, <o'_3, o_3>\} \) by eliminating the edge \( <v, o_1> \). Accordingly, we get \( f(o'_1) = o_2, f(o'_2) = o_1 \) and \( O' = \bigcup f(o'_i) = \{o_0, o_1, o_2\} \).

(3) Get \( I = \bigcup f(o) = f(o_1) \cup f(o_2) = \{i_1, i_2\} \cup \{i_3, i_4, i_5\} \).

(4) As \( (|I'| = 2 \land |I| = |I'|) \Rightarrow |I'| = |I|, calSim \) continues.

(5) Use \( IOMatch \) to find an injection \( g \) from \( I \) to \( I' \), i.e., \( g: I \rightarrow I' \). Since \( |X| = |Y| \), \( IOMatch \) uses the Kuhn-Munkres algorithm directly to find the optimal matching \( M = \{<i_1, i'_1>, <i_2, i'_2>, <i_3, i'_3>\} \). Thus, we get \( g(i_1) = i'_1 \) and \( g(i_2) = i'_2 \).

(6) Calculate \( sim(p, r) \) according to formula (2).

\[
sim(p, r) = \frac{\text{SimCC}(o'_1, o_1) \times \text{SimCC}(i'_1, i_1) + \text{SimCC}(i'_2, i_2)) / 2}{\text{SimCC}(o'_1, o_1) \times \text{SimCC}(i'_1, i_1) + \text{SimCC}(i'_2, i_2)) / 2} = 0.6375
\]

The value of \( sim(p, r) \) is less than the threshold value \( \alpha = 0.8 \) specified in the request, it denotes that the operation doesn’t satisfy the request. If none of the operations within a service can satisfy the request alone, it doesn’t mean that the service should be ignored. In fact, a composition of operations within a service can bring value-added functions. Thus, in this case, TSSD goes on to its second phase.

**Second Phase: Discovery Based on Composition Matching**

Operations can be connected by feeding one’s output to the other’s input in an orchestration way similar to an assembly line in a factory. Notice that there are such cases where the inputs of a service cannot be fed by all the outputs from only one service, but can be fed by the outputs from more than one service. At present, we only support the linear composition in
which two services can be concatenated together only if all the inputs of the successive service are fed by the outputs of the preceding one. In this section, we introduce how TSSD performs service discovery based on the matchmaking of operation compositions.

**Definition (Operation-Concatenation)** Given a service \( s = (n_s, P, f_s) \) and its two operations \( p_1 = \{ n_{p_1}, I_{p_1}, O_{p_1}, f_{p_1} \} \) and \( p_2 = \{ n_{p_2}, I_{p_2}, O_{p_2}, f_{p_2} \} \), \( p_2 \) can be concatenated to \( p_1 \) (denoted as \( P_1 \cdot P_2 \)) if and only if:

1. \( |O_{p_1}| \geq |I_{p_2}| \), and
2. For \( \forall i_1, i_2 \in I_{p_2} \) and \( i_1 \neq i_2 \), \( \exists o_1, o_2 \in O_{p_1} \), \( o_1 \neq o_2 \), \( i_1 \in f_{p_1}(o_1) \), and \( i_2 \in f_{p_1}(o_2) \)

If an output \( o \) of \( p_1 \) is fed into an input \( i \) of \( p_2 \), they construct a Concatenation Point (denoted as a 2-tupe \( cp = \langle o, i \rangle \)). \( P_1 \cdot P_2 \) is called an Operation-Concatenation and denoted as a 3-tupe: \( oc = (p_1, p_2, CP) \) where \( CP = \{ cp_1, cp_2, ..., cp_k \} (k = |O_{p_2}|) \) is the set of connection points.

This definition indicates the conditions under which two operations can be concatenated together. The first condition ensures that the input number of \( p_2 \) is not larger than the output number of \( p_1 \); and the second condition ensures that for any input of \( p_2 \) there is one different output of \( p_1 \) that can be assigned to the input. According to the assignment relations defined in a service, one output can be assigned to an input if both of them are the same concept or the output is a sub-class of the input. For an concatenation point \( cp = \langle o, i \rangle \) if the output and the input are of the sub-class-of relation, the semantic difference between them bring a semantic distance in an operation-concatenation.

For an operation concatenation \( oc = (p_1, p_2, CP) \), the semantic distance of the concatenation (denoted as \( \zeta(oc) \)) is computed according to the formula (3).

\[
\zeta(oc) = 1 - \frac{\sum_{cp \in CP} SimCC(cp.o, cp.i)}{|CP|}
\] (3)

**Definition (Operation-Sequence)** For a service \( s = (n_s, P, f_s) \) and an ordered operation set \( os = \langle p_i, p_j, ..., p_k \rangle \) \( (1 \leq i, j, ..., k \leq n) \), for any two neighboring elements \( p_i \) and \( p_j \) in \( os \), if they construct an operation-concatenation, i.e., the relation \( p_i \cdot p_j \) holds, \( os \) is called an operation-sequence for \( S \). The number of the operations in \( s \) is called the length of the operation-sequence.
If two operations construct an operation-concatenation, they also construct an operation-sequence with the length of 2. In fact, every two neighboring operations in an operation-sequence form an operation-concatenation.

**Definition (Sequence-Distance)** For an operation-sequence $os = <p_1, p_2, \ldots, p_n>$, its Sequence Distance (denoted as $\zeta(os)$) is the weighted-sum of a Physical Distance (denoted as $\ell(os)$) and a Semantic Distance (denoted as $h(os)$), where the parameter $\lambda$ is a preference parameter specified by a service requester according to his/her preference.

\[
\zeta(os) = \lambda (1 - \frac{1}{\ell(os)}) + (1 - \lambda) h(os), \text{ where } 0 \leq \lambda \leq 1 
\]  

(4)

The Physical Distance of $os$ is defined as the length of $os$, i.e., $\ell(os) = |os|$. The Semantic Distance of an operation-sequence is computed according to the formula (5), where $oc$ is an operation-concatenation constructed by two neighboring operations $p_i$ and $p_{i+1}$.

\[
h(os) = \frac{1}{|os| - 1} \sum_{i=2}^{|os|-1} \zeta(oc_i)
\]  

(5)

For example, consider the operation-sequence composed by three operations named \( p_1, p_2 \), and \( p_3 \), respectively, shown in Figure 5. The value of $\ell(os)$ is 3 and the value of $h(os)$ is calculated as:

\[
h(os) = \frac{1}{3} \sum_{i=2}^{2} \zeta(oc_i) = \frac{1}{3} (0.85 + 1.0) = 0.92.
\]

Thus, the Sequence Distance is $\zeta(os) = \lambda (1 - \frac{1}{3}) + (1 - \lambda) \times 0.92 = 0.775 + 0.589\lambda$.

An operation-sequence can provide value-add functions for users. However, to compose operations together on-line for each new incoming request is time-consuming, especially when the number of operations within a service is large. In order to avoid the time consumption in the service discovery, we can transfer the composition process from the service discovery to the service registration. That means when a service is registered, we can find out all the possible operation-sequences in it and also compute their sequence-distances using a background program running on the service registry. After all the operation-sequences are constructed, they can be used for the service discovery based on composition matchmaking.

From the perspective of a user, an operation-sequence can be regarded as a pipe-line that accepts inputs from the beginning operation, generates outputs at the end and hides its inner details. Thus, an operation-sequence can be transformed into an operation extended with a sequence-distance attribute. In this operation, the input set is the input set of its first operation and the output set is the output set of its last operation. Thus, for an operation-sequence $os$
= <p_1, p_2, ..., p_n> and a service request \( r = (P, O_r, \omega) \), we can follow the steps of the algorithm \( calSim \) to compute the similarity \( Sim(\text{os}, r) \) between them except that in step (5) we calculate the similarity using the new formula (6) instead of (2).

\[
Sim(\text{os}, r) = \sum_{o'} \left( \sum_{i \in I_{o'}} SimCC(i, g(i)) \right) \left( \frac{1}{O_r(f(o'))} \right) \times (1 - \xi(o))
\]

If the value of \( SimSR(\text{os}, r) \) is larger than the threshold value \( \omega \), the sequence is returned; Otherwise, it is excluded. Notice that a user can adjust the value of \( Sim(\text{os}, r) \) by giving the parameter \( \lambda \) a different value.

### IMPLEMENTATION AND EVALUATION

The aim of this section is to introduce the implementation of the proposed service discovery mechanism; and also to show the efficacy of the discovery method through a set of experiments.

**Implementation of Service Discovery in DartFlow**

The proposed service discovery mechanism have been implemented in DartFlow (Deng, Li et al. 2006c) which is a business process management platform for e-commerce. Its goal is to provide a convenient and efficient way to model and execute collaborative processes based on Web services across enterprises. The architecture of the service registration and discovery sub-system in DartFlow is shown in Figure 6. It can be divided into two sides: the client and server.

For the client, we have implemented a service registration and discovery portal based on Eclipse GEF as shown in Figure 7. With this portal, users can register and discover services. During the registration, a user provides a service’s URL. Then the portal parses the WSDL files and displays operations and interfaces using graphic elements as shown in the right-bottom of Figure 7. Then the user is required to annotate the service with ontology concepts shown in the top-right of Figure 7.

For the server, we have built a service registration center using Apache JUDDI (v0.9) and Oracle database (Oracle 9i). In order to support the extended-WSDL model in JUDDI, we have added additional tables in database to save semantic information, interface dependency, assignment relation and operation sequences for a registered service. Two programs named ARG (Assignment Relation Generator) and OSG (Operation Sequence Generator) are running on the background of registration center to generate assignment relations and operation sequences for a new registered service. In order to deal with both Chinese and English concepts, we use HowNet as the ontology resource—an common-sense knowledge base unveiling inter-

---

*Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.*
conceptual relations and inter-attribute relations of concepts as connoting in lexicons of the Chinese and their English equivalents (Dong and Dong, 2006). The version used in the system is HowNet 2005 which has more than 80000 Chinese concepts and 70000 English concepts. We have implemented the algorithm proposed by Li et al. (2003) using JAVA to compute the semantic similarity between concepts from HowNet. For the two-phase semantic service discovery, we have implemented it through the combination of two algorithms named OM (Operation Matchmaker) and CM (Composition Matchmaker). Moreover, we also provide a simple version of Operation Matchmaker named SOM (Simple Operation Matchmaker) which doesn’t consider the interface dependency. For CM, we use a backward-chaining method to retrieve all the operation sequences for a new registered service in advance.

Experiments and Evaluations

We evaluate the performance of the service discovery mechanism by using three well-recognized metrics, namely service recall rate, precision rate, and scalability. The recall rate is the proportion of relevant services that are retrieved. It can be calculated according to the formula (7). The precision rate is the proportion of retrieved services that are accurately matched which can be calculated according to the formula (8). Scalability refers to the computing complexity with respect to the growing number of services being accessed.

\[
\text{Recall Rate} = \frac{|\text{relevant services} \cap \text{retrieved services}|}{|\text{relevant services}|}
\]

(7)

\[
\text{Precision} = \frac{|\text{relevant services} \cap \text{retrieved services}|}{|\text{retrieved services}|}
\]

(8)

EXPERIMENTAL SETUP

In order to prepare the test set for the discovery experiments, we developed a tool based on the IBM XML Generator that enables one to generate random XML files based on schemas. With this tool, we generate 5 groups with 100 services for each as Table 2 shows. During the generation of each group, we ensure that the number of operations in this group is approximately in accordance with the statistics aforementioned in Introduction, i.e., 25% services has only 1 operation, 45% has 2-4 operations, 25% has 5-9 operations and 5% has 10-15 operations. Moreover, we make the proportion of partially-dependent outputs (PPDO) for the 5 groups is 0%, 20%, 60%, 80% and 100%, respectively. We select a sub-concept-tree with 200 concept-nodes from HowNet to be used for the annotation for inputs and outputs. When generating an operation of a service, the number of inputs or outputs is randomly selected from 1 to 5. And each input/output is annotated with a concept randomly selected from the sub-concept-tree.

We carry out a series of service discoveries on each group. The discovery process is carried out automatically and the criteria for judging whether a service satisfies a request is to determine whether there is a matched operation or matched operation composition. Notice that we use a same service request which is randomly generated within the sub-concept-tree for all the service discoveries conducted in the experiments. We run the experiments on an IBM x260 server with a 2.0-GHz Intel Xeon MP processor and 1-GB of RAM, running a RedHat Linux operating system.

RECALL RATE AND PRECISION OF SERVICE DISCOVERY

**Experiment A:** How the Partially-Dependent Outputs Influence the Recall Rate and Precision?
In this experiment, we carry out twice service discovery using OM and SOM on each group. We specify the threshold value in the request as $\omega = 0.8$. The experiment result is illustrated in Figure 8. For each group in the following two bar charts, the first bar represents the recall rate or precision from SOM whereas the second represents that from OM.

The experiment result illustrates that 1) both the recall rate and precision from SOM maintain almost stable with respect to the proportion of partially-dependent outputs. However, the recall rate from OM is increasing dramatically with respect to PPDO; the precision from OM is increasing slightly with respect to the proportion of partially-dependent outputs; 2) for each group besides G-1, the recall rate and precision from OM gain a distinct improvement on those from SOM; 3) the improvement from OM is increasing with respect to PPDO. The average improvement for recall rate and precision from G-2 to G-5 is about 24% and 15%, respectively; 4) for G-1, the result from OM and SOM are the same because there are no partially-dependent outputs in it.

The above four findings indicate that 1) taking interface dependency information into consideration can bring a better recall rate and precision; especially that it can improve the recall rate to a great extent; 2) the more the partially-dependent outputs in a set of services, the more significant the improvement of the recall rate and precision.

**Experiment B: How the Operation Composition Influence the Recall Rate and Precision?**

In this experiment, we carry out a service discovery using the combination of OM and CM on each group. We specify the threshold value in the request that $\omega = 0.8$ and the parameter $\lambda = 0.6$ for composition matchmaking. The comparison between the results of this experiment with those from Experiment A is illustrated in Figure 9.

The comparison illustration shows that 1) for each group, both the recall rate and the precision from OM&CM are better than those from SOM and OM; 2) Compared to SOM (or OM), OM&CM gain an average improvement of the

![Figure 6. The architecture of service registration and discovery sub-system in DartFlow](image_url)
recall rate and precision that are 37% (or 15%) and 21% (or 8%), respectively; the improvement of the recall rate from OM&CM is larger than that of the precision; 3) the improvement of the recall rate and the precision from OM&CM compared to OM maintains almost stable although the proportion of partially-dependent outputs is increasing from G-1 to G-5.

The above three findings indicate that taking the composition of operations into matchmaking can bring a better recall rate and precision. However, due to the ignorance of interface dependency information in composition matchmaking, the improvement is not affected by the proportion of partially-dependent outputs.

**Experiment C: How the Preference Parameter λ Influence the Composition Matchmaking?**

In this experiment, we carry out a series of service discovery using CM with the parameter λ from 0 to 1 with a 0.1 interval on each group. We specify the threshold value ω = 0.8. The experiment is shown in Figure 10.

The experiment result shows that, for each group, the recall rate reaches the max when the parameter λ = 0.6 and the precision reaches the max value when the parameter λ = 0.8.

**Table 2. Test set preparation**

<table>
<thead>
<tr>
<th>Group</th>
<th>Service Number</th>
<th>Proportion of Partially-Dependent Outputs (PPDO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-1</td>
<td>100</td>
<td>0%</td>
</tr>
<tr>
<td>G-2</td>
<td>100</td>
<td>20%</td>
</tr>
<tr>
<td>G-3</td>
<td>100</td>
<td>60%</td>
</tr>
<tr>
<td>G-4</td>
<td>100</td>
<td>80%</td>
</tr>
<tr>
<td>G-5</td>
<td>100</td>
<td>100%</td>
</tr>
</tbody>
</table>

Figure 7. Service registration and discovery portal in DartFlow
SCALABILITY OF SERVICE DISCOVERY

Experiment D: How the scale of a service repository influence the response time?

A good service discovery method should scale well as the number of services being accessed is increasing. After accepting a service discovery request, it must return the target services quickly even for a repository with a large number of services. We evaluate the discovery for different scale of service repositories. We generate 6 test sets with 100, 200, 400, 600, 800 and 1000 services and make the proportion of partially-dependent outputs in each set 20%. On each set, we conduct three discoveries using SOM, OM and OM&CM, respectively with the parameter $\lambda = 0.7$. The response time of each discovery is shown in Figure 11.

The experiment result illustrates that 1) the response times for SOM, OM and OM&CM are almost linearly increasing with respect to the number of services; 2) the response time of OM is almost the same to that of SOM; 3) the response time of OM&CM is about twice as much as that of SOM and OM. The findings indicate that the proposed discovery method scales well with the number of services in a repository. That is to say the proposed approach can return the result in a short time even the service repository has a large number of services.

CONCLUSION

This article proposes a two-phase semantic-based service discovery mechanism. The main contributions of this work are (1) it is the first time to point out the two shortcomings in the most current service matchmaking methods and find out that both of them lower the recall rate and the precision for discovery; (2) it proposes an information model for registered services in a repository with feasible and convenient mechanisms to describe operation relations and interface dependencies implied in a service; (3) it proposes a two-phase semantic-based service discovery method. Compared to other approaches, the new method has two salient characteristics: (a) it takes into account the interface dependencies and imports the bipartite graph matching to improve the efficiency of matchmaking; b) it supports two-level matchmaking, namely operation matchmaking and operation-composition matchmaking. A series of experiments demonstrate that the proposed mechanism has both a good recall rate and precision and also scales well with the number of services accessed.

However, the two-phase semantic-based service discovery mechanism, in the second phase, only supports matchmaking on the linear composition of operations. So, how to enhance its ability to support matchmaking on more complex composition is one of the future directions. Moreover, to simplify the problem at this stage...
so as to focus on the key points of the article, this work has not taken into account the QoS requirements for a service request at present, which is also one of our future direction.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 60803004, 60873224 and 60873045; the National High-Tech Research and Development Plan of China under Grant 2007AA01Z124, 2008AA01Z141 and 2009AA01Z121.

A preliminary and shorter version of this article was published as a 2-page poster paper in the proceeding of 17th International Conference on World Wide Web (WWW2008), Beijing, China, April 21-25, pp1189-1190.

REFERENCES


Figure 9. The influence of operation composition on recall rate and precision

Figure 10. The influence of parameter $\lambda$ on composition matchmaking
Figure 11. Scalability of service discovery with the increasing number of services accessed

![Graph showing scalability of service discovery](image-url)


ENDNOTE

Notice that, for OM&CM, the time consumption doesn’t include the time for composing operations on-the-fly. In fact, the process of composing is carried out during the service registration. All the possible compositions are prepared before discovery. Thus, the time for composing operations is excluded from the total discovery time. So, the time consumption of OM&CM increases linearly.