An Efficient Algorithm of Service Discovery Based on Service Clusters

JunJing Gai and Yu Yue Du
Shandong University of Science and Technology, 266590, Qingdao, China

Corresponding Author: Yu Yue Du, Shandong University of Science and Technology, 266590, Qingdao, China

ABSTRACT

A service discovery algorithm based on service clusters is proposed in this study. By using ontology and Petri nets, the formal definition of services is given and service clusters are formally represented as service cluster net units. The usefulness of the algorithm is demonstrated by an experiment.

Key words: Petri net, service discovery, service cluster, algorithm

INTRODUCTION

With the development of Service Oriented Architecture (SOA), the number of services is becoming larger and larger. Due to the growth of Web services, service discovery faces a huge challenge.

The semantic Web Service can describe the function of services. The service matching based on semantic description can understand the semantic information of services and service requests well and improve the accuracy of semantics.

Mass services in a library are clustered by using the clustering technology. The functions of the services in a cluster are similar. However, the functions of the services in several different clusters are highly dissimilar. The service clustering can improve the accuracy of the service classification and reduce the size of service searching. Therefore, it can improve the efficiency of service discovery.

Petri nets are a tool for modeling and analysis of distributed systems and have the strict mathematical definition and the ability of graphic expression. It can effectively describe the information processing systems with concurrency and uncertainty. Service clusters have the nature of parameter uncertainty. Petri nets will be used to describe and analyze service clusters in the study.

With the wide application of the semantic Web, the service discovery methods is researched extensively based on semantic description. In Pan and Zhang (2009), a service discovery method is proposed and it improves the performance of services by combining the syntax and semantics. In Wu et al. (2005), a service discovery method is proposed based on ontology and semantic similarity. The semantic similarity of services is discussed by constructing service ontology. They can improve the accuracy of the service discovery from the perspective of semantics. But the efficiency of service discovery is not increased. And the service compositions satisfied a service request can not be found by previous methods efficiently.

The efficiency of the traditional service discovery is enhanced in this study. Services are described based on semantics and they are clustered by semantic similarity (Rajagopal and Selvi, 2009). Through clustering services, the searching scale can be reduced greatly. Service clusters are described formally and service cluster net units are introduced. By using service cluster net units
proposed in this study, the services can be quickly found. And service compositions can be built efficiently. The efficiency and the accuracy of service discovery are greatly improved.

In this study, $R$ is a matrix and $R^p$ represents the $p$'th row of $R$. $R^p(q)$ is the $q$'th value in the $p$'th row of $R$. When $p = 1$, the $q$'th value in the first row of $R$ can be expressed by $R(q)$.

**BASIC CONCEPTS**

Ontology and Petri nets are briefly reviewed in this section. In this study, $N$ is a natural number set, i.e., $N = \{0, 1, 2, \ldots\}$; $N_k = \{0, 1, 2, \ldots, k\}$; $N^* = N \setminus \{0\}$; and $N_k^* = N_k \setminus \{0\}$.

Ontology can be represented as an oriented tree, an ontology tree. The nodes of the ontology tree represent concepts. Connections between the nodes correspond to the semantic concepts.

**Definition 1 (Semantic similarity):** Concepts $X$ and $Y$ are similar in the semantics if and only if $X$ and $Y$ are synonymous, or the similarity of $X$ and $Y$ is higher than a certain value.

**Definition 2 (Ontology concept set):** An ontology concept set is a set of some ontology concepts, represented by $O = \{c_1, c_2, \ldots, c_k\}$, where $c_i$ is an ontology concept, $n \in N^*_k$, $i \in N^*_k$.

Petri nets can not only describe the structure of a system but also simulate the operation of the system (Du et al., 2008). Petri nets (Murata, 1989) are defined as follow.

**Definition 3 (Petri net):** $N = (S, T; F)$ is a net where:

- $S \cup T \neq \emptyset$
- $S \cap T \neq \emptyset$
- $F \subseteq (S \times T) \cup (T \times S)$
- $\text{Dom}(F) \cup \text{cod}(F) = S \cup T$, where $\text{dom}(F) = \{x \in S \cup T \exists y \in S \cup T: (x, y) \in F\}$ and $\text{cod}(F) = \{x \in S \cup T \exists y \in S \cup T: (y, x) \in F\}$

Petri nets are used to describe services and service clusters in this study.

**SERVICES AND SERVICE CLUSTERS**

The formal description of Web services can be given from Definition 1.

**Definition 4 (Web service):** A service can be represented by $WS = (BI, SF, I, O, QoS)$ where:

- $BI$ is the basic information of a service
- $SF$ is the functions of a service
- $I$ is the input set of a service and $\forall i \in I, i = (i.\text{concept}, i.\text{value})$ where:
  - $i.\text{concept}$ is the ontology concept corresponding to the input parameter
  - $i.\text{value} = \{0, 1\}$, i. value = 0 represents the input parameter can not be given and i. value = 1 represents the input parameter can be given
- $O$ is the output set of a service and $\forall o \in O, o = (o.\text{concept}, o.\text{value})$ where:
  - $o.\text{concept}$ is the ontology concept corresponding to the output parameter
  - $o.\text{value} = \{0, 1\}$, o. value = 0 represents the output parameter can not be obtained and o. value = 1 represents the output parameter can be obtained and
- $QoS$ is the quality of the service
In Definition 4, BI includes the service name, the service provider, etc. QoS includes the response time, the price, etc. Similarly, a service request can be defined as follow.

**Definition 5 (Service request):** SR = \{SF_r, I_r, O_r, QoS_r\} is a service request where:

- SF_r is the functions needed to be obtained
- I_r is the set of the input request. \( \forall ir \in I_r, ir = (ir\text{.concept}, ir\text{.value}) \)
- O_r is the output set of the request and \( \forall or \in O_r, or = (or\text{.concept}, or\text{.value}) \) and
- QoS_r is the set of the qualities needed to be obtained.

To improve the efficiency of service discovery, service clusters are proposed. Services are clustered (Skoutas et al., 2010) according to their functions. The services with similar functions are in a same cluster. Petri nets can describe the uncertainty and concurrency of a service cluster. By Definition 3, the formal description of service clusters can be defined as follow.

**Definition 6 (Service cluster net unit):** CNU = \{CF, tu, Iu, Ou, Fu\} is a service net unit where:

- CF is the function of the service cluster
- tu is the transition of a service cluster net unit
- Iu is the input parameter set of the services in the service cluster and \( \forall iu \in Iu, iu = (iu\text{.concept}, iu\text{.label}) \) where:
  - iu\text{.concept} is the ontology concept corresponding to the input parameter
  - iu\text{.label} is an \( n \times 1 \) matrix where \( n \) is the number of the services in the service cluster and
  - iu\text{.label} is the value corresponding to iu\text{.concept} for a service
- Ou is the output parameter set of the services in the service cluster and \( \forall ou \in Ou, ou = (ou\text{.concept}, ou\text{.value}) \) where:
  - ou\text{.concept} is the ontology concept corresponding to the output parameter
  - ou\text{.label} is an \( n \times 1 \) matrix where \( n \) is the number of the services in the service cluster
  - and ou\text{.label} is the value corresponding to ou\text{.concept} for a service; and
- Fu = (Iu\times tu)U(tu\times Ou) is a set of directed arcs.

**Example 1:** There exists three services: \( ws_1, ws_2 \) and \( ws_3 \) where \( ws_1 = \{(i_1, departure, 1), (i_2, date, 1), (i_3, arrival, 1), (i_4, ID, 1), (o_1, time, 1), (o_2, price, 1), (o_3, flightno, 1)\} \), \( ws_2 = \{(i_1, departure, 1), (i_2, date, 1), (i_3, arrival, 1), (i_4, ID, 1), (o_1, time, 1), (o_2, price, 1)\} \) and \( ws_3 = \{(i_1, departure, 1), (i_2, date, 1), (i_3, arrival, 1), (i_4, ID, 1), (o_1, time, 1), (o_2, price, 1), (o_3, flightno, 1), (o_4, flignttype, 1)\} \). The service cluster net unit can be shown in Fig. 1.

In Fig. 1, Iu and Ou are two matrices and they are called the input matrix and the output matrix. In an input matrix, the input parameters of services in a service cluster are numbered and each row is the values corresponding to the input parameters.

By service cluster net units, the scale of service discovery is greatly reduced and the efficiency of service searching can be improved. While by the formal description of service clusters, a novel service discovery algorithm can be proposed to further improve the efficiency.
SERVICE DISCOVERY ALGORITHM

A service discovery algorithm based on service clusters is proposed in this section. Firstly, a service cluster discovery algorithm is presented to reduce the searching scale based on service clusters. Then, a service matching algorithm is given to reduce the difficulty of the service discovery and a service discovery algorithm in the service cluster is shown to further improve the efficiency of the searching services. A service composition algorithm is proposed to search the service compositions satisfying a service request. In the follow, the service satisfying a service request is called an atomic service.

In the study, the number of services is n, the number of the service clusters is w and the number of the services in the service cluster is m, n ∈ N, w, m ∈ N *. Let the services be W = (B, S, I, O, QoS) and the number of input and output parameters are n, and n, respectively. CNU = {CF, tu, Iu, Ou, Fu} is a service cluster net unit, i ∈ N *, k ∈ N *. A service request is s = {f, i, o, qos}. In the next algorithms, Iu, Ou, i, and o are represented as matrices. And Iu is an m×n matrix. Ou is an m×n matrix. i is an 1×n matrix. o is an 1×n matrix. Iu(q) is the p * row of the matrix Iu. Iu(q) is the q'th value in the p'th row of the matrix Iu.

A matching algorithm is proposed first in this subsection.

Algorithm 1: A service cluster discovery algorithm

```
Input: WS=(B, S, I, O, QoS), CNU={CF, tu, Iu, Ou, Fu} and s=[f, i, o, qos];
Output: the service cluster net unit cnu of the service cluster that satisfy s;
Step 1: u=1;
Step 2: if u=w, then
        Step 2.1: if CF is not similar with sf, u=u+1;
        Step 2.2: else output cnu=CNU;
Step 3: else cnu is not found
```

From Algorithm 1, the searching scale can be reduced a lot. The searching number without service clusters is n. The searching number of Algorithm 1 is w. And w<n, the efficiency of the service discovery can be improved.
The process of the matching services is very important in the process of the service discovery. Services can be found by a determination as long as the services can satisfy a certain matching conditions. A novel service matching algorithm is proposed as follow.

**Algorithm 2: A service matching algorithm**

1. **Input:** $CN_{U_2}=\{ CF_{i_2}, tu_{i_2}, Iu_{i_2}, Ou_{i_2}, Fu_{i_2} \}$ and $s_{i_2}=\{ s_{i_2}, i_{i_2}, q_{i_2}, q_{os_2} \}$
2. **Output:** $A$ and $B$, where $A$ is an $m \times n$ matrix and $B$ is an $m \times n$ matrix.
3. **Step 1:** $p_1=1$, $q_1=1$, $q_2=1$;
4. **Step 2:** if $p_1 < m$:
   1. if $Iu_{i_2}(q_2)=1$ and $i_{i_2}(q_2)=0$, $A^{i_{i_2}(q_2)}=0$;
   2. if $A^{i_{i_2}(q_2)}=1$;
5. **Step 3:** if $p_2 < m$:
6. **Step 4:** output matrix $A$ and matrix $B$.

Algorithm 2 is a service matching algorithm based on service cluster net units. The time complexity of Algorithm 2 is $m \times n_1^2 + m \times n_2^2$.

Matrix $A$ is called the input matching matrix and matrix $B$ is called the output matching matrix. $A^{i_{i_2}(q_2)}=1$ means the $q_2$’th parameter can be satisfied by the service $ws_{i_{i_2}}$ and $B^{i_{i_2}(q_2)}=1$ means the $q_2$’th parameter can be satisfied by the service $ws_{i_{i_2}}$. Then if $A^{i_{i_2}}=1$, the service can satisfy $i_{i_2}$, and if $B^{i_{i_2}}=1$, the service can satisfy $i_{i_2}$. The services can be discovered by a certain determination. A service discovery algorithm is needed.

A service discovery algorithm is represented first in this subsection.

**Algorithm 3: A service discovery algorithm**

1. **Input:** input matching matrix $A$ and output matching matrix $B$;
2. **Output:** $S$: a set of services;
3. **Step 1:** $p=1$, $q_1=1$, $q_2=1$;
4. **Step 2:** if $p=m$ and $q_1 < n$:
   1. if $A^{i_{i_2}(q_2)}=1$, $q_1=q_1+1$;
   2. if $A^{i_{i_2}(q_2)}=1$;
5. **Step 3:** if $q_2 < n$:
6. **Step 4:** output $S$. 
The required services can be discovered according to Algorithm 3. In Algorithm 3, the output services in set $S$ are that their corresponding rows of $A$ are:

$$\begin{bmatrix} 1, 1, \ldots, 1 \end{bmatrix}_{n_i}$$

and corresponding rows of $B$ are:

$$\begin{bmatrix} 1, 1, \ldots, 1 \end{bmatrix}_{n_o}$$

That means the service request can be satisfied by the services in the set $S$. The searching number of Algorithm 3 is $m \times n_i + m \times n_o$.

In a service cluster, one service may not satisfy a service request. However, a composition of several services may satisfy the service request. Service compositions conclude the parallel composition and the serial composition. In this study it is assumed that the granularity of service clusters is small enough and the situations of serial compositions do not exist.

From Algorithm 2, $A^v(q_i) = 0$ means the input parameter $q_i$ is not satisfied in the service request. In other words, the service can not be fired in the service request. It is assumed that compositions with $v$ services need to be found, $v \in N_m^+$. A service composition discovery algorithm is proposed as follow.

**Algorithm 4:** A service composition discovery algorithm

```
Input: a $v \times n_i$ matrix $C$, the input matching matrix $A$ and the output matching matrix $B$
Output: $S$: a set of services and service compositions:
Step 1: $p = 0$
Step 2: if $p < m$, $p = p + 1$ and $q_i = 1$
    Step 2.1: if $q_i < n_i$ and $A^v(q_i) = 1$, $q_i = q_i + 1$
    Step 2.2: else if $A^v(q_i) = 0$, update all elements of $A^p$ and $B^p$ are 0 and go to Step 2;
    Step 2.3: else go to Step 2;
Step 3: else select $v$ rows from $E_p$ in matrix $C$
    Step 4: if each column of $C$ exists a value of 1, then $S = S_1$ (the composition of the $v$ services), else select another $v$ rows from $E_p';$
Step 5: if all possible combinations are selected, output $S$.
```

By Algorithm 4, the service compositions can be discovered. And when $v = 1$, Algorithm 4 are equivalent to Algorithm 3.

**COMPARATIVE ANALYSIS**

This section describes the experiments to verify the feasibility and effectiveness of the proposed algorithms. 500 services about buying books online are selected from UDDI server. And the services are registered on the local server as experimental subjects. To avoid the chanceiness of results, this entire experiment is repeated 100 times.
Fig. 2: Efficiency of 5 service numbers

Two experiments are done. The first experiment is atomic service discovery and the second experiment is to search service compositions of two services. The two experiments are done by a traditional algorithm (Elgazzar et al., 2010) and the algorithm in this study.

Figure 2 shows the results of searching services in both the traditional algorithm and in the algorithm proposed in this study. The horizontal axis is the number of services and the vertical axis is the respond time. Two lines represent the results of the first experiment and another two lines represent the results of the second experiment. The traditional algorithm is represented as “traditional” and the algorithm proposed in this study is represented as “new”. In the follow analysis, the respond time is contrary to the searching efficiency.

Several expected results can be found by observing Fig. 1. First, the efficiency of the algorithm proposed in this study is clearly higher than that of the traditional algorithm. Second, the efficiency is decline with the increase in the number of services. Third, the efficiency of searching service compositions is slower than that of the atomic service discovery.

CONCLUSION

A service discovery algorithm is proposed in this study. Both atomic services and service compositions can be found by the algorithm proposed in this study. And the validity of the algorithm is demonstrated by an experiment. And by the comparative analysis, the efficiency of service discovery can be improved a lot. The method of searching service composition can be used in the research of service composition. And service cluster net units proposed in this study can be used to analyze and study service composition and service substitution.

ACKNOWLEDGMENTS

This study is supported by the National Natural Science Foundation of China under Grants 61170078 and 61173042; the National Basic Research Program of China under Grant 2010CB328101; the Doctoral Program of Higher Education of the Specialized
REFERENCES