Appl. Math. Inf. Sci. **8**, No. 4, 2025-2032 (2014) 2025

Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080461

Real-Time Service Composition Algorithm based on Semantic Web

Hu Jingjing[1,*], *Ma Siying*[1], *Zhao Xing*[2] *and Cao Yinyin*[1]

[1] School of Software, Beijing Institute of Technology, Beijing 100081, P. R. China
[2] School of Mathematics, Capital Normal University, Beijing 100037, P. R. China

Received: 30 Aug. 2013, Revised: 2 Dec. 2013, Accepted: 3 Dec. 2013
Published online: 1 Jul. 2014

Abstract: Real-time service composition needs to meet the time constraints precisely and enhanced the speed of combination computing quickly. To improve the accuracy of service selection, we establish a time ontology model and a set of inference rules, the search strategies through logic reasoning combined with the extended UDDI improve the search efficiency by 129% and accuracy by 161%. In the process of service composition, the improved simulated annealing algorithm (WSC-ISA) is proposed to implement the NP-hard problem of web service composition with QoS properties, which optimizes the solution by time priority and accepting deterioration solution in probability. Experiment results show that the efficiency of WSC-ISA is improved by more than 6% and the time is reduced by more than 12.3%, which shows the algorithm reduces the computation time while maintains the high efficiency.

Keywords: Service composition, Semantic Web, QoS, real-time

1 Introduction

With the widely used of web services and the technology becoming increasingly mature, a growing number of web services have been published on the Internet, and it arose the web service composition (WSC) [1,2]. Semantic web has been developed to make WSC not only limited to the data type and keyword matching, but also raised to the ontologies' relations and kinship degree reasoning, which makes WSC more accurate and reliable than normal way [3,4]. The semantic extension of web service is mainly implemented in non-functional property of web service – QoS, which involves time, cost, reputation and others. The temporal information is an important factor for a user to invoke web service [5]. But now the vast majority of the methods are focusing on the functions of semantic web services or on automatic QoS discovery, searching and matching, real-time demand for services is ignored [6]. Therefore, a real-time web service composition planning is presented in this paper. In the framework, a semantic searching strategy is used to complete the automatic discovery of services, and in the service composition process, the improved simulated annealing algorithm is used to optimize the service composition based on QoS properties. The algorithm achieves higher timeliness and efficiency in WSC.

The paper is organized as follows: The next section proposes the search strategies based on semantic reasoning to improve the accuracy of candidate services selection. The third section puts forward to the WSC-ISA algorithm with QoS-aware to improve the efficiency of WSC. The fourth section explains the performance evaluation for experiments, and the last section presents our conclusions.

2 Search strategies

Before searching for the temporal condition of semantic web services, a temporal hierarchical ontology should be established firstly, which is used to divide time periods into different types. When a web service is registered, the publisher will record the time information to an extended 'tModel' according to its position in the temporal hierarchy. In the searching process, we can first make sure the position of the time type of the current service in the hierarchy, and then get all the keywords of the types which contain the current type. The inferred keywords will be sent to UDDI for searching, and the returning

* Corresponding author e-mail: hujingjing@bit.edu.cn

services are further referred by Jena engine and the time ontology to make sure their availability.

2.1 Temporal hierarchy and UDDI extension

Ontology is an official, accurate and formalized norm for describing the knowledge of a specific field, and also a concept model and modeling language in the semantic knowledge level [7,8]. Our system uses the standard web ontology language (OWL-S) to establish the domain ontology repository, especially build the concept of time as well as the relationship between the time units.

The major part of searching strategy is inference of searching candidate services. The inference promotes the keywords matching to semantic level, in order to make sure the information is not only literally matched but also has the same meaning. During the inference, the time structure is used is in the time hierarchy, which divides time into different types depending on the time unit and creates a tree structure. The temporal hierarchy is depicted as Figure 1.

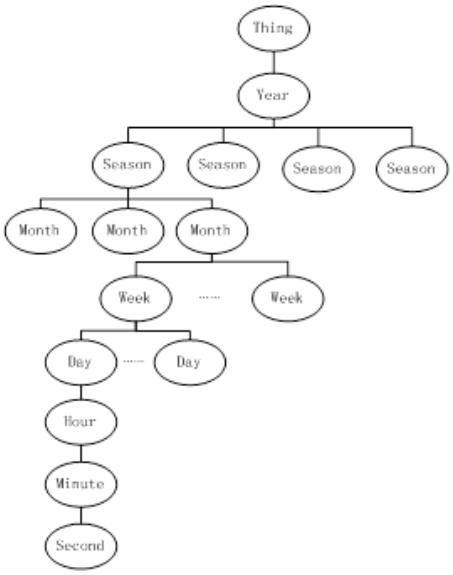


Fig. 1: Temporal hierarchy

In order to coordinate the semantic inference with the search strategy, we extend the UDDI by creating a tModel for time property. When the publisher registers a web service, a tModel needs to create additionally. The tModel includes the URL of time description, and the temporal hierarchy keyword of the service. The tModel completes the reflection of time property from OWL-S to UDDI. Its structure is depicted as Table 1.

For the example of 'Hour', when using the search strategy, we firstly look for the position of 'Hour' in the temporal hierarchy, and find out that the type of 'Day', 'Week', 'Month', 'Season' and 'Year' also includes hour. So during the search for tModel, we can find type of 'Hour' and the types include it, and ignore the types which are included by it, such as the type of 'Second' and 'Minute'. The returning services are mostly available when searching by this way. Experimental results show that the search efficiency is kept higher than 70% when using the strategy in searching for 100, 400, 700, 1000, 1400 and 1700 services. The rate is slightly decreased when the number of services rises. Compared with it, the search efficiency by using random strategy is more depending on the number of the searching information. If the searching information has a large amount in UDDI, the efficiency is higher. If there is less, the efficiency will become very low, which is usually lower than 50% in average. The comparison of the two strategies is shown as Figure 2.

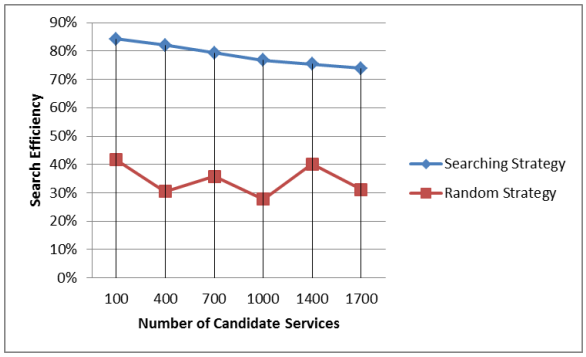


Fig. 2: Search efficiency of two strategies

The experiment results show that the efficiency is improved by an average of 129%.

2.2 Ontology Reasoning

The search strategy combines the ontology structure with the application of the inference engine. In the returned service lists from UDDI, we should verify each service of them whether its conditions match the requests of users'.

Table 1: tModel for time

tModel for Time	
Category Bag	
KeyName	Time
KeyValue	Second/Minute/Hour/Day/.
tModelKey	UUID of OWL-S Time tModel
overviewDoc	
description:	Time Description
overviewURL	Time Ontology URI

To implement it, the time ontology is built as the inference schema in the engine, the service information as binding data and the matching inference rules.

2.2.1 Time Ontology

To verify whether the details of the current service meet the user's needs about real-time conditions, it establishes the structure of time ontology-time.owl in this section, and the matching inference rules are also constructed. This structure of time ontology is shown as Figure 3.

Fig. 3: Structure of time ontology

There are two subclasses of Temporal Entity: 'Instant' and 'Interval'. Instants are point-like concept in which have no interior points while Intervals are things with extent. It is generally safe to think of an instant as the interval with zero length, where the beginning and end are the same. 'ProperIntervals' are those time intervals whose beginnings and endings are different, and 'ProperInterval' is the subclass of Interval. It is disjoint with 'Instant'.

An interval can be described by CalendarYear in different time interval, like 2010-09-20, or described by the length of time, like 2 hours and 34 minutes. CalendarYear type is described by 'DateTimeDescription' class, and the length of time is described by 'DurationDescription' class.

DateTimeDescription with DurationDescription has different temporal unit, so they cannot be compared directly. Here we create a 'TemporalUnit' class including Year, Month, Day, Hour, Minute and Second. The upper unit can contain the units which are lower than it. For example, if a service is available in Year 2011, it must be available in June of 2011.

2.2.2 Inference rules

When using the inference engine, appropriate inference rules need to be constructed after the modeling of time ontology. The file of time.owl only defines different temporal classes and their relations, while inference rules are needed to make it work in the inference engine.

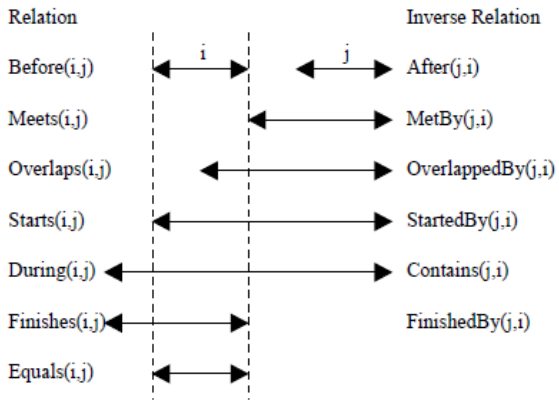


Fig. 4: Property relations analysis

Before constructing the rules, we can do formal analysis to the logic of these relations, which is shown as Figure 4.

According to the analysis, we can simplify the relations between intervals, and express them with formula expression. Intervals can be described by their beginning time and end time, so their relations can also be described by relations between their beginning time and ending time, namely the relation '$=, >, <$' between these instants. In the rules, 'i' represents interval, 'e' represents end and 'b' represents beginning in formula (1)–(7).

$$i_1 \text{ before } i_2 \equiv e_1 < b_2 \tag{1}$$

$$i_1 \text{ meets } i_2 \equiv e_1 = b_2 \tag{2}$$

$$i_1 \text{ overlaps } i_2 \equiv (b_1 < b_2) \wedge (e_1 < e_2) \wedge (e_1 > b_2) \tag{3}$$

$$i_1 \text{ starts } i_2 \equiv (b_1 = b_2) \wedge (e_1 < e_2) \tag{4}$$

$$i_1 \text{ during } i_2 \equiv (b_1 > b_2) \wedge (e_1 < e_2) \tag{5}$$

$$i_1 \text{ finishes } i_2 \equiv (b_1 > b_2) \wedge (e_1 < e_2) \tag{6}$$

$$i_1 \text{ equals } i_2 \equiv (b_1 = b_2) \wedge (e_1 = e_2) \tag{7}$$

Inference rules are constructed according to the formulas, and they have specific format which is provided by Jena [9]. For example, formula 1 can be translated into a rule like this : [rule_iconcept_intervalBefore: (?x time: hasEnd ? b) (?y time: hasBeginning ?e) (?b time: before ?e)− >(?x time: intervalBefore ?y)].

2.2.3 The process of reasoning

In order to implement the querying and reasoning process, a reasoner with ModelFactory needs to create.

The resoner will read in the file of time.owl as the basic model schema, bind the inference rules, and read in the OWL-S of services returning from UDDI as data. Then it calls the createInfModel method of ModelFactory, and acquires the inference result InfModel. The InfModel is a basic model, so we can call the general method of ModelFactory to read the inference result from InfModel. Figure 5 depicts the flow of reasoning.

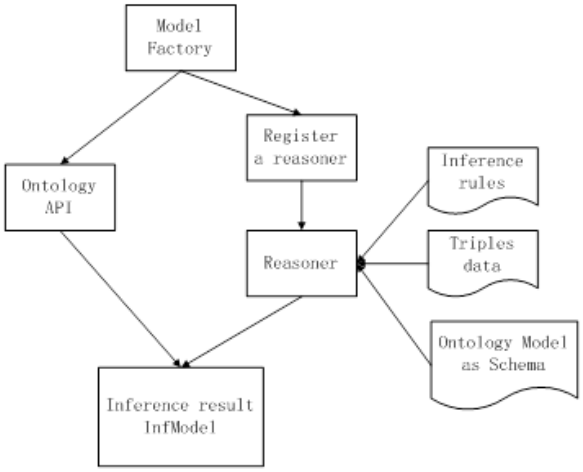


Fig. 5: the flow of inference engine

When the inference engine binds the schema and reads in the data, it can infer new information which contains the original instance data and is not included in the read-in data according to the inference rules. The method InfModel.Difference (data) is used to acquire the information which is different from data.

It is more accurate by using the ontology reasoning method to verify the temporal data in service selection than the way of keyword matching. Because it solves the problem of 'one word with two meanings' in the level of semantic matching, it gets relatively higher search efficiency than keyword matching. Figure 6 depicts the comparison of search efficiency between ontology reasoning and keyword matching.

The experiment results show that the accuracy of candidate services' selection is improved by an average of 161%.

3 The algorithm of WSC

3.1 Pretreatment of web service information

After the services returned from UDDI by using the search strategy, the value of QoS need be optimized, especially the time information should satisfy the real-time properties

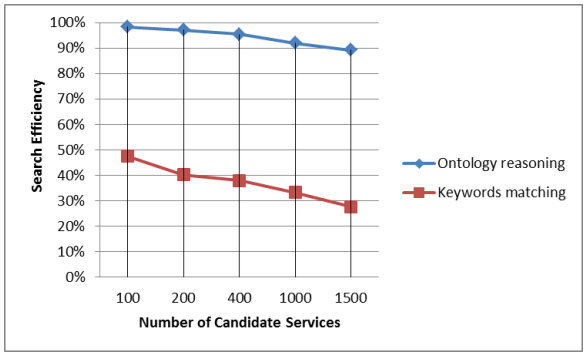


Fig. 6: Comparison of search efficiency between ontology reasoning and keyword matching

of service composition. In the paper, the properties of time, cost, availability and reputation are taken into account.

Property of the QoS has a wide and floating range, so it need to normalize the value in order to calculate and compare the property with weight, which is provided by users. In the normalization, count the range of the parameter x firstly to get its maximum as 'a' and minimum as 'b'. Then parameter x is in the interval of $[a,b]$. The value of properties range in $[0,1]$, the formula 8 represents the normalization.

$$X = \frac{x-a}{b-a} \tag{8}$$

3.2 Composition algorithm based on QoS

3.2.1 Coding of service flow

There are four types of logical structure of service composition, and they are 'sequence', 'choice', 'parallel' and 'cycle'. Figure 7 shows the structures.

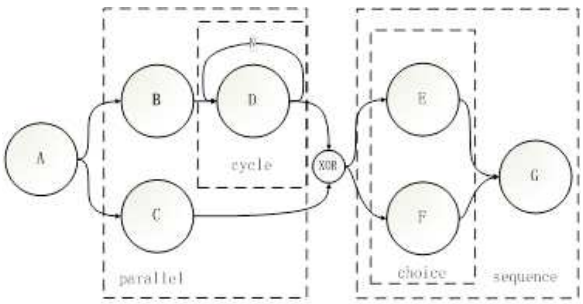


Fig. 7: Structure of service composition

In the algorithm, the services in the candidate service lists should be encoded, and each code identifies one service uniquely. The length of code is not fixed, but it depends on the composition structure and the number of service lists according to users' requisition instead. For example, if there are n services in sequence structure, the code can be a k-bit binary number. The relation between n and k is expressed in formula 9. The code of cycle structure is the same as the sequence.

$$k = \lceil \log_2 n \rceil \tag{9}$$

In 'choice' structure, the value of former part depends on the number of alternative paths, recording the path of current service in. The latter represents every service in a list. 'n' is the count of paths and'm'is the maximal value of the service number in all paths. The code of this service in the structure is expressed in formula 10.

$$k = \lceil \log_2 n \rceil + \lceil \log_2 m \rceil \tag{10}$$

In 'parallel' structure, if there are n paths, and in each path there are x_i services, the code of the service is shown in formula 11.

$$k = \sum_{i=1}^{n} \lceil \log_2 x_i \rceil \tag{11}$$

For all the codes of the above structure, let $k = k+1$ to reserve the extension of representation.

3.2.2 Improved Simulated Annealing Algorithm

It is known that the solution of WSC based on QoS is an NP-hard problem [10,11]. In this paper, we consider to use simulated annealing (SA) to solve the problem. SA is an algorithm that simulating the metal annealing process [12]. According to heat theory, when temperature is T, the probability of the appearance of energy difference is $P(\Delta E)$. The formula expression of it is shown as formula 12, in which k is constant and $\Delta E < 0$, so we can assure the value of $P(\Delta E)$ is in range $(0,1)$, and it will decrease when T decreases.

$$P(\Delta E) = \exp\left(\frac{\Delta E}{kT}\right) \tag{12}$$

In the algorithm, a topical random transforming way to generate a new solution is used, a few nodes to generate new code are chosen randomly and others remained unchanged. Each new solution will be verified its legitimacy and existence, and then will be evaluated by its value.

The function of $Evaluate()$ will calculate each solution's value according to the weight which is assigned to each property of QoS by users. The attributes of time, cost, availability and reputation are considered, in which V represents the value of QoS attributes, N is the number of structural nodes of a path, s_i denotes the service selected of a node. The formulas for them can be shown in the following formula $13-16$.

(1) Time

$$V_{Time} = \sum_{i=1}^{m} V_{Time}(s_i) \tag{13}$$

(2) Cost

$$V_{Cost} = \frac{1}{n}\sum_{i=1}^{N} V_{Cost}(s_i) \tag{14}$$

(3) Availability

$$V_{Avail} = \frac{1}{n}\sum_{i=1}^{N} V_{Avail}(s_i) \tag{15}$$

(4) Reputation

$$V_{Repu} = \frac{1}{n}\sum_{i=1}^{N} V_{Repu}(s_i) \tag{16}$$

Since the algorithm aims at real-time WSC, the value of 'time' is the sum and the other parameters are average. Therefore two cases should be considered. If user has provided the maximal time duration $V_{TimeMax}$, then the time must be shorter than $V_{TimeMax}$; if user has not given the $V_{TimeMax}$ but assigned the weight to time, we set the $V_{TimeMax}$ as a negative number and calculate all the parameters with their weights, in which 'W' is for weight, and evaluation function can be written as formula 17.

$$Evaluate(p) = \begin{cases} -\infty & V_{Time} > V_{TimeMax} \\ \dfrac{V_{Avail}W_{Avail} + V_{Repu}W_{Repu}}{V_{Cost}W_{Cost}} & V_{Time} \le V_{TimeMax} \\ \dfrac{V_{Avail}W_{Avail} + V_{Repu}W_{Repu}}{V_{Cost}W_{Cost} + V_{Time}W_{Time}} & V_{TimeMax} < 0 \end{cases} \tag{17}$$

The algorithm of improved SA for real-time service composition (WSC-ISA) is described in Table 2.

In line 01,'SS' is the service with logic structure and the coding is according to the method of section 3.2.1. In line 02, the initial solution must comply with the time requirements. The simulated annealing process starts form line 03. In line 06, each new solution should be checked firstly if it has existed. The 'r' in line 07 is cooling coefficient, which can control the rate of annealing

Especially, in line 05 it accepts the deterioration solution in probability of $P(\Delta E)$ in order to reduce the concentration of solution and increase the probability to get the optimal solution. The function of $Evaluate()$ enlarges the utility of time.

4 The evaluation of performance

The instance shown in Figure 7 was adopted to verify the validity of the algorithm. The running environment is

CPU: Intel 2.20 GHZ, RAM: 2.0GB. Linear Programming (LP) algorithm [13] to compare with WSC-ISA algorithm in different number of candidate services of each node and the structure of the service path includes four type of logical flow.

The efficiency of the algorithm is measured by 'e', which $e = |s|/|C|$. $|C|$ is the times of service composition, and $|s|$ is the successful times denoted by formula 18. There is $|C| = 500, W_x = 0.25$, the number of candidate services for each node ranges from 200 to 1000.

$$s = \begin{cases} 1, & V_{Time}(CS) \leq 0.6 \times V_{TimeMax}, V_{TimeMax} > 0 \\ 1,, & V_{Time}(CS) \leq 0.6 \times V_{Time}, V_{TimeMax} < 0 \\ 0, & others \end{cases} \tag{18}$$

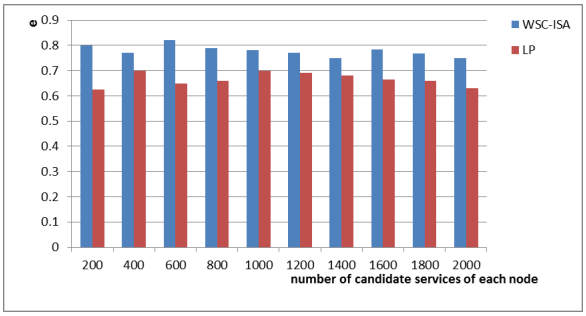


Fig. 8: Comparison of efficiency of WSC-ISA and LP

The processing time of simulated annealing can be controlled by the simulated temperature and cooling coefficient. The higher T, r and T_{min} we have, the longer and more meticulous the simulating process will be. T and r should be relatively raised if the number of services of each node rises up, to ensure the simulating process is enough. Table 3 and 4 denote the relations of T, r and number of services. Figure 8 shows the comparison of efficiency for WSC by and WSC-ISA and LP.

It can be seen from the experiments that WSC-ISA obtains the optimal solution or approximate optimal solution for time priority in a high probability. LP algorithm is usually used in sequence structure and it cannot be applied to multiple branching paths, when using in complex structures, the nodes must be transformed into simple structures firstly. The efficiency of WSC for the test by WSC-ISA is higher than LP's in average 14.4%.

Table 2: Algorithm **WSC-ISA**

Input: SS
Output: CS
Begin
01: Code(SS) ;
02: Generate the initial solution path P_0 and create *Initlist*;
03: Set the current temperature $T = T_{max}$;
04: Select a solution from *Initlist* as the current state Pi
05: $Pi+1$ =neighbor(Pi);
$\Delta E = Evaluate(Pi+1) CEvaluate(Pi)$;
$P(\Delta E) = \exp(-\Delta E/kT)$;
If $\Delta E > 0$ insert $Pi+1$ into *ResultList* in order;
ELSE If ($\Delta E <= 0$ & $P(\Delta E) >$ random) insert
$Pi+1$ into *ResultList* with the probability of $P(\Delta E)$;
06: Insert the generated new solution to the previous path list;
07: Set $T = rT$;
If $T > T_{min}$ $Initlist = ResultList$; go to (3);
ELSE exit the process.
08: CS $<-$ DecodeOptimal (***Evaluate***(*ResultList*));
09: RETURN CS;
end

Table 3: Relations between r and number of nodes

Number of nodes	< 5	$6-9$	$10-13$	$13-15$	> 16
r	$1-10^{-3}$	$1-10^{-4}$	$1-10^{-5}$	$1-10^{-6}$	$1-10^{-7}$

Table 4: Relations between T and number of candidate services

Number of services	< 1000	$1000-1500$	$1500-2000$	$2000-2500$	> 2500
T	10^4	10^5	10^6	10^7	10^8

In order to verify the performance of improved SA, the experiments were carried out to compare the efficiency with the not improved. In the SA, the V_{time} is computed by the same way of other attributes, and the deterioration solution was not accepted.

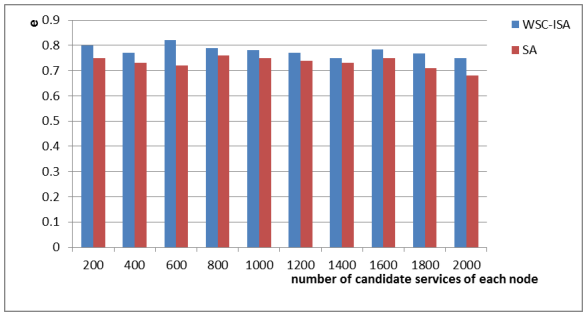


Fig. 9: Comparison of efficiency by WSC-ISA and SA

According to statistical results from Figure 9, the efficiency of WSC for the test by WSC-ISA is higher than SA's about 6%.

The composition time is another evaluation indicator for real-time WSC. The comparison of time by using WSC-ISA and LP is shown in Figure 10.

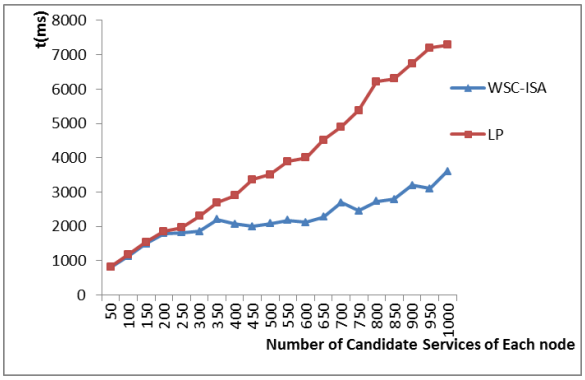


Fig. 10: Comparison of composition time by WSC-ISA and LP

The composition time of WSC-ISA and LP are both prolonged with the number of candidate services' increasing and the time of WSC using WSC-ISA is shorter than LP by an average of 33.8%.

The comparison of time by using WSC-ISA and SA is shown in Figure 11.

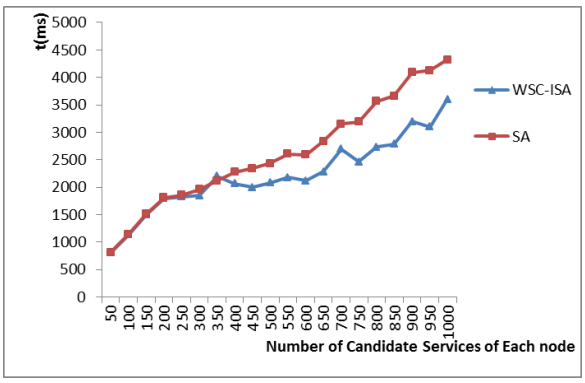


Fig. 11: Comparison of composition time by WSC-ISA and SA

The figure shows that in the experiment, both computing of composition time can be accomplished in 5 seconds, and the time of WSC-ISA is shorter than SA by an average of 12.3%, which shows algorithm WSC-ISA reduces the computation time while maintains the high efficiency.

5 Conclusions

In real-time systems, punctuality is the primary requirement. Especially in solving the NP problem of web service composition for QoS, it must guarantee that time is short enough and accuracy is high enough.

In this paper, we propose a composition algorithm based on semantic web services framework. The UDDI is extended and the time ontology inference is builtthe selection of candidate services by reasoning gets more accuracy. In the process of composition, the algorithm of WSC-ISA with time priority is designed, which obtains higher combination efficiency in shorter time. The algorithm improves the accuracy for time matching and reduces the composition time, which can meet the needs of real-time services in a certain extent.

Acknowledgement

This work has been supported by the National Science Foundation of China (Grant No. 61101214, 61371195), the Key Project of National Defense Basic Research Program of China (Grant No. B1120132031) and the Fundamental Research Funds for the Central Universities (Grant No. 20120842003, 20110842001). Thanks for the help.

References

[1] B.Al-Shargabi, A.Sabri and A.El Sheikh, Web Service Composition Survey: State of the Art Review, Recent Patents on Computer Science, **3**, 91-107 (2010).

[2] H. Wang, Y. Shi, X. Zhou, Web Service Classification using Support Vector Machine, IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, 3-6 (2010).

[3] R. Karunamurthy, F. Khendek, R H.Glitho, A Novel Architecture for Web Service Composition, Journal of Network and Computer Applications, **35**, 787-802 (2012).

[4] H. Tian, K. Liu, Research on Semantic Web Service Composition, IEEE World Automation Congress, WAC 2012, 1-4 (2012).

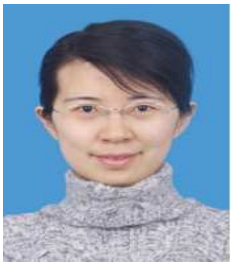
[5] Y.B. Kim, Real-time Estimation and Analysis of Time-based Accessibility and Usability for Ubiquitous Mobile-Web Services, KSII Transactions on Internet and Information Systems, **5**, 938-958 (2011).

[6] C. Okutan, N.K. Cicekli, A Monolithic Approach to Automated Composition of Semantic Web Services with the Event Calculus, Knowledge-Based Systems, **23**, 440-454 (2010).

[7] Q. Yu, L. Wang, D. Huang, Fishery Web Service Composition Method based on Ontology, Journal of Integrative Agriculture, **11**, 792-799 (2012).

[8] D. Paulraj, S. Swamynathan, M. Madhaiyan, Process Model-based Atomic Service Discovery and Composition of Composite Semantic Web Services using Web Ontology Language for Services (OWL-S), Enterprise Information Systems, **6**, 445-471 (2012).

[9] H. Fethallah, C. Amine, B. Amine, Automated Discovery of Web Services: an Interface Matching Approach based on Similarity Measure, Proceedings of the 1st International Conference on Intelligent Semantic Web Services and Applications, 13 (2010).
[10] F. Lecue, N. Mehandjiev, Satisfying End User Constraints in Service Composition by Applying Stochastic Search Methods, International Journal of Web Services Research, **7**, 41-63 (2010).
[11] J. J. Hu, X. Zhao, Y. D. Cao, Research on Transaction Web Service Selection Algorithm in WSC, Applied Mathematics & Information Sciences, **7**, 725-731 (2013).
[12] X. Q. Fan, X. W. Fang, C. J. Jiang, Research on Web service selection based on cooperative evolution, Expert Systems with Applications, **38**, 9736-9743 (2011).
[13] Bellman R. DYNAMIC PROGRAMMING AND A NEW FORMALISM IN THE CALCULUS OF VARIATIONS. Proc Natl Acad Sci U S A., **40**, 231-235 (1954).



Hu jingjing received the PhD degree in Computer science from Beijing Institute of Technology, Beijing, China. She is currently a lecturer in the school of Software of Beijing Institute of Technology. Her research interests are in the areas of service computing, multi-agent systems, and GPU-based computer tomography.

Ma siying is a postgraduate in the school of Software, Beijing Institute of Technology, China. Her research interests include artificial intelligence, services computing, software engineering, etc.

Zhao xing received the Ph.D degree in Computer science from University of Science & Technology of China, HeFei, China. He is currently an associate professor in the school of Mathematical sciences of Capital Normal University. His research interests are in the areas of computer tomography, service computing.

Cao yinyin is a postgraduate in the school of Software, Beijing Institute of Technology, China. Her research interests include parallel computing, services computing, software engineering, etc.