http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (3): 399-411, 2010
ISSN 1812-5638

Optimizing Services Composition in Multi-Network Environment

[1]Keting Yin, [1]Bo Zhou, [1]Shuai Zhang, [1]Honghong Jiang and [2]Jerry Cristoforo
[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China
[2]State Street Corporation, Boston, MA, 02110, USA

Abstract: In this study, we research on QoS-aware services selection for web service composition in multi-network environment. We revise the web service QoS (Quality of Service) model and propose a service selection approach based on Integer Programming (IP) to meet new requirements in multi-network environment. Compared with traditional approaches, experimental results show that our approach can achieve a better solution with acceptable performance cost.

Key words: Web service, service composition, quality of service, multi-network, integer programming

INTRODUCTION

Web service (Curbera *et al*., 2002) is the default implementation of Service-Oriented Architecture (SOA) (Papazoglou, 2003) and is prevalently adopted due to its platform-independent, language-neutral, ease of integration and many other advantages. Moreover, basic web services could be used as building blocks to construct value-added services, which are customer-oriented and able to fulfill real-life tasks. This process is called web service composition (Rao and Su, 2005), which is a hot topic both in academic and industrial areas.

As more and more web services are available on the public internet (Fan and Kambhampati, 2005; Li *et al*., 2007) and the underlying industry-level standards (e.g., BPEL (Andrews *et al*., 2003), ebXML (Clark *et al*., 2001), WSLA (Keller and Ludwig, 2003), WS-Security (Atkinson *et al*., 2002) etc.) get mature, web service composition evolves from intra-corporation integration in early days to cross-organization and web-scale in recent years. A composite web service could be composed of elementary services that belong to different administrative domains and are located in different networks. Furthermore, even services composition inside a corporation may involve multi-network interactions since large enterprises tend to build data centers in distributed geographical locations.

Service composition in multi-networks is different from traditional one in the following three aspects:

- Conventional services composition solutions are not designed with the multi-network environment in mind, which generally neglect the service transmission time or simply simulate it using a random value for each service. In multi-network environment, service transmission time is no longer trivial. Cross-network data transmission could take a while depending on the connection condition between source and destination networks
- Since data are transmitted over the internet, more stringent security policies should be applied such as detection for attacks targeting at XML messages (Hines, 2006) in addition to basic user authentication and ordinary checking, which further incurs performance penalty
- The degree of trustworthiness for each network should be differentiated through security policies imposed on service selection and execution engine. For example, if some services are carrying out sensitive tasks, only those located in a highly trusted network are valid for selection

Consequently, service composition in multi-networks environments should be aware of the hosted network information in order to optimize performance and obey security policies.

In this study, we propose a solution for optimizing services composition in multi-network environment based on Integer Programming (IP). Our contributions can be stated as follows:

- QoS model for web service in multi-network environment. We discuss four important QoS dimensions for web service: delay time, price, reliability and reputation and revise their conventional modeling approach to better suit the multi-network environment. This QoS model can be extended to accommodate more QoS dimensions

Corresponding Author: Bo Zhou, College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China
Tel: +86-0571 87933387 Fax: +86-0571 28111775

- QoS and security aware service composition model for multi-network environment. We exploit the new requirements in performance and security for multi-network environment and redefine the classical QoS-aware service composition problem
- IP-based solution for optimizing services composition in multi-network environment. We also present an IP-based solution for the proposed QoS-aware service composition problem and the experiments show the effectiveness of our approach

PRELIMINARIES

Centralized vs. decentralized service composition: Web service composition can be orchestrated in centralized as well as decentralized modes (Chafle *et al*., 2004; Binder *et al*., 2006). In centralized service composition, component web services do not interact with each other directly but have to pass the data back and forth to the execution engine. The execution engine is responsible for managing all the control logic and distributing data to the needed web services. This mode suffers from single point of failure and performance bottleneck in the node where the engine is deployed. Contrarily, in decentralized service composition, the control logic is distributed among several execution engines and the data can be transmitted in peer-to-peer mode, which is more efficient in most cases and more robust. We use a scenario to illustrate the difference between two orchestration modes.

Figure 1 is a simple example for web service composition. It is aggregated from three web services in sequential. The control flow and data flow are denoted using directed solid and dashed lines respectively. Correspondingly, Fig. 2 is an illustration for the web service composition orchestrated in centralized and decentralized modes, which shows the difference in data flows intuitively.

Although, decentralized service composition has advantages in performance and reliability, centralized service composition is much easier to be implemented. Furthermore, some organizations require full control of the composition logic, which is effortless in centralized mode but highly difficult in decentralized one. For these reasons, the adoption of centralized mode is more widespread in industry nowadays. The optimizing processes of service composition in multi-networks for these two kinds of orchestration modes are not alike due to the differences in control and data flow. In this paper, our work aims at multi-network service composition in centralized mode.

Web service QoS model: Functionality of the web service was once the major concern when the service resources were sparse. Now as plenty of services are available and even more are emerging, it is very common that multiple web services have overlapping or identical functionality but different QoS levels. Meanwhile, business-level applications have very strict non-functional requirements such as delay time and reliability. Thus, it is essential to build the QoS model for web service, which has been discussed in many literatures (Menascé, 2002; Ran, 2003; Liu *et al*., 2004; Zeng *et al*., 2003, 2004). We discuss four important QoS dimensions for web service in multi-network environment, which are delay time, price, reputation and reliability. The QoS model is extensible to accommodate more dimensions. The aggregated QoS for composite web service is also discussed here for the basic sequential composition structure.

Delay time: Delay time measures the elapsed time between the moment the user requests a service and the

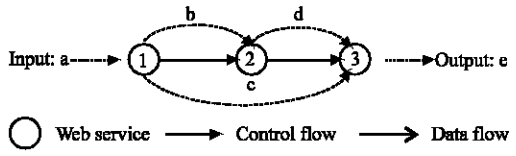

Fig. 1: Example web service composition

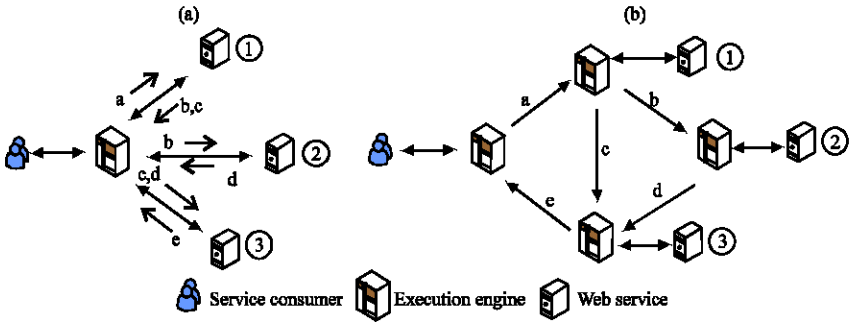

Fig. 2: Centralized vs. decentralized service composition. (a) Centralized service composition and (b) decentralized service composition

moment that a response is returned. It can be divided into execution time and transmission time. The execution time is the time used to process the service request, which will generally be advertised by the service provider and the transmission time is the amount of time taken to carry messages from the provider to the consumer (or in opposite direction), which depends on the network condition and security mechanisms applied in cross-network data transmission. So the delay time of a web service in multi-network environment can be expressed as:

$$\begin{aligned} &Delay(ws_i) = Tran(ns_j, ns_i) + Exe(ws_i) + \\ &Tran(ns_i, ns_j) + Tran(ns_i)*2 + Tran(ns_j)*2 \end{aligned} \quad (1)$$

where, ws_i represents the web service, ns_i and ns_j are the networks where ws_i and the consumer locates respectively, $Exe(ws_i)$ is the service execution time, $Tran(ns_i, ns_j)/Tran(ns_j, ns_i)$ is the cross-network transmission time and $Tran(ns_i)/Tran(ns_j)$ is the intra-net transmission time.

The transmission time for intra-net is much smaller than that for cross-network, i.e. Tran(nsi)/Tran(nsj) <<Tran(nsi, nsj)/Tran(nsj, nsi), thus Eq. 1 can be simplified as:

$$Delay(ws_i) = Tran(ns_j, ns_i) + Exe(ws_i) + Tran(ns_i, ns_j) \quad (2)$$

The transmission time between two networks can be estimated using historical execution records.

The aggregated value of the composite service for this quality dimension can be evaluated as sum of delay times of the component services.

Price: The fee has to be paid to the service provider in order to fulfill a service request. Pay per hit and subscription-based payments are the two basic pricing models for web services currently in use (Xignite Inc., 2009). The former charges according to the number of invocations of the service, while the latter charges a specified amount of money over a specified amount of time. Other pricing models such as auction-based are active topics in research (Zhang *et al*., 2007). In this study, we adopt the pay per hit payment mode for its popularity and simplicity. The aggregated value for this quality dimension can be simply evaluated as sum of the prices of the component services.

Reliability: Reliability is defined as the probability that the response for a service request can be received within the expected duration after invoking the web service, with a value range of [0, 1]. The reliability of a web service can be affected by the following three level factors: service-level, host-level and network-level. Service-level factors are specific to the individual service, such as excessive invocations of the service at peak time, logical errors in the service implementation or incompatibility issues. Host-level factors are related with the hosting server, such as overload of server or server crash. Network level factors are determined by participating networks, such as unreliable connection between the source and destination networks. Apparently, if two services are hosted on the same machine or located in the same network, their reliabilities are not independent.

With the assumption that reliabilities of different services are independent, the aggregated value for reliability can be evaluated as the product of the reliabilities of the component services. We will model the correlations of reliabilities between the services at the host and network level in our further work.

Reputation: Reputation of a web service is often used as a measure of its trustworthiness, which is generally calculated based on the consumers' feedbacks (Wang and Vassileva, 2007). Most existing reputation mechanisms are focused on the web service itself, but neglecting the impact of its provider. In fact, the reputation of the service provider is generally more stable than that of individual services and should be considered when evaluating the reputation of a web service (Yin *et al*., 2008):

$$RepE(ws_i) = (1-\delta) * Rep(ws_i) + \delta * Rep(sp_i) \quad (3)$$

where, $Rep(ws_i)$ is the reputation of a web service, $Rep(sp_i)$ is the reputation of the service provider, $RepE(ws_i)$ is enhanced to take the provider's reputation into account and $\delta \in [0, 1]$.

In multi-networks environment, web services inside a network share many characteristics in common due to physical (e.g., the same infrastructure) and/or logical (e.g., the same administrative domain) isomorphism. Thus, we can model the reputation of the network similar to that of the service provider and enhance the web service reputation with the network reputation. The aggregated

Table 1: Aggregated function of QoS for sequential composition structure

QoS dimension	Single web service	Composite web service
Delay time	$Delay(ws_i)$	$\sum_{i=1}^{n} Delay(ws_i)$
Price	$Price(ws_i)$	$\sum_{i=1}^{n} Price(ws_i)$
Reliability	$Rel(ws_i)$	$\prod_{i=1}^{n} Rel(ws_i)$
Reputation	$RepE(ws_i)$	$\frac{1}{n} * \sum_{i=1}^{n} RepE(ws_i)$

value for reputation can be evaluated as the arithmetic average of the component services' reputation.

Table 1 shows the method to aggregate the four QoS dimensions for sequential composition structure.

QoS aware service composition: We first introduce several concepts as the basis for discussion:

Abstract service class: Web services belong to the same class achieves the same functionality but may have different non-functional properties (delay time, reputation, location of network, etc.) and different providers. They are potential replacements for each other and called equivalent web services.

Process definition: The process definition specifies the composition structures using abstract service class, which is a skeleton of the executable service composition.

Elementary web service: Web service does not invoke other services to fulfill its task.

Composite web service: A composite web service is composed of other services and completes the task via the cooperation of its components services. Its component services can be elementary or composite web service. Composite web service can be generated by instantiating the process definition.

Web service composition can be staged in two phases. The first phase is functionality-oriented, in which the process definition is automatically generated given the initial and desired state using AI planning methods (Rao and Su, 2005) or created manually. The second phase is QoS aware services selection for the composition. In this phase, for each abstract service class in the process definition, web services which conform to the functional specifications will be discovered as candidate for selection. Then the services that optimize the user defined objective function (e.g., the shortest delay time) and meet the end-to-end QoS constraints for the composition will be selected. Figure 3 shows the process of QoS aware service composition.

QoS-aware service composition in multi-networks environment has the following additional requirements:

- The host networks of the component services should be considered during the selection process since the delay time between networks will affect the overall delay time of the composite web service
- In centralized execution mode, services interact via the execution engine so the location of the engine should also be carefully selected to minimize the transmission time
- Moreover, the consumers are distributed among multi-networks and they send requests to and receive responses from the engine. Thus, the distribution of the consumers is also a factor to decide the location of the engine
- Lastly, security policies should be conformed during the selection of service and the host network for the execution engine

OPTIMIZING SERVICES COMPOSITION IN MULTI-NETWORK ENVIRONMENT

Problem statement: Equivalent web services may appear in the same network or in multiple networks. The composition task is to choose the concrete web services for the defined process as well as the location of the

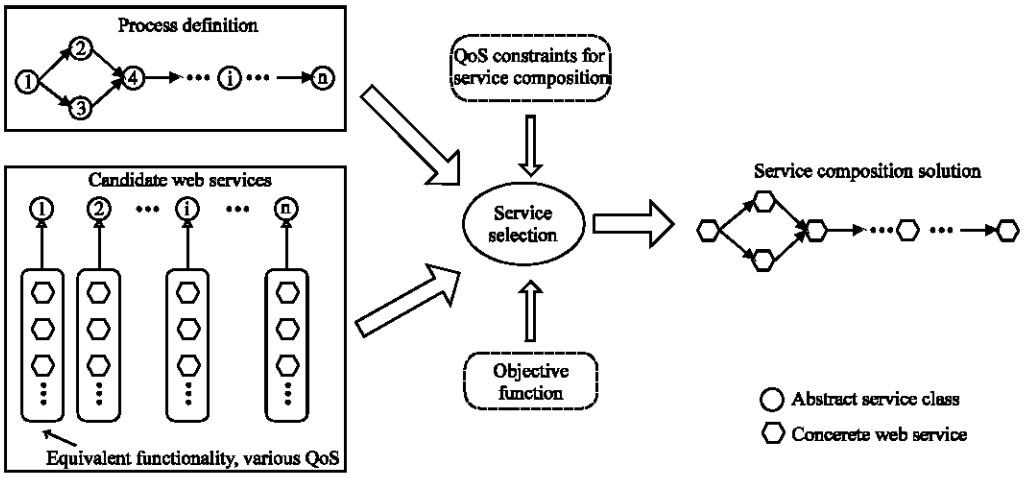

Fig. 3: Traditional QoS aware service composition

execution engine in order to optimize performance (i.e., shortest delay time) from the perspective of end users. The optimizing process should meet the following constraints:

- The end-to-end QoS constraints for the service composition required by the users should be met
- Only the services deployed in the networks whose trust level is above the threshold defined by the security policies are valid for selection
- Only the networks whose trust level is above the threshold defined by the security policies are valid locations to deploy the execution engine

Optimization process: Assuming the process definition is already given and candidate component services have been discovered, the optimization process starts with the following five steps. Related concepts and notations are defined at the end of the paper.

Step 1: Define networks: Networks are identified according to their connection conditions, security policies as well as administrative domains. Each network is assigned with a trustworthiness value by the process owner or third authorization parties

Step 2: Selecting valid networks: Define sensitivity level for the abstract service class and execution engine. A web service ws_{ij} is valid for selection if $Trust(NS_{ij}) >= Sensitivity(S_i)$, where $ws_{ij} \in S_i$. For ease of discussion, we assume any web service mentioned is already valid for selection in the rest of this section. A network could be used to deploy the execution engine if $Trust(ns_i) > Sensitivity(e)$. We define

$$K' = \{p \mid Trust(ns_p) > Sensitivity(e)\} \quad (4)$$

Thus K' is the set of identifiers of the qualified networks for deploying the execution engine.

Step 3: Preliminary selection and evaluation: In order to speed the service selection process, the service classes which have only one valid candidate web service could be instantiated at this point. Then the aggregated QoS of those instantiated service classes could be estimated and compared against the end-to-end QoS constraints. If any constraint is not met at this step, there is no possibility that a solution would exist and the optimization process terminates here, otherwise, the process continues in step 4.

Step 4: Optimizing services selection and composition based on integer programming: IP-based optimizing approach is the key step of the whole process. The objective is to optimize the mean delay time between requesting the service and receiving the result by the users under the conditions that end-to-end QoS constraints are satisfied, via selecting the web services and location of the execution engine appropriately in the multi-networks defined above. The integer programming formulation will be discussed in detail in the following section.

Step 5: Runtime adjustment and re-optimizing: Runtime verification and adjustment needs to be applied after deploying the execution engine and selecting the web services to ensure the performance goal is indeed achieved. Moreover, web services are subject to changes in the open environment. Existing services may become unavailable or degrade in QoS and new services with better QoS criteria could appear. So in order to maintain or improve the composite web service continuously, re-optimizing should be performed periodically or triggered by monitored events.

Integer programming formulation: Linear Programming (LP) is a mathematical model for the optimization of a linear objective function, subject to linear equality and linear inequality constraints, where all variables are continuous (Wikipedia, 2009). Frequently, for many problems, the variables can take only integer values. These problems are called Integer Programming (IP) (Wolsey, 1998) problem, which is NP-complete. An IP problem is formulated by defining the variables, objective and constraints properly. Here, we formulate the problem of service composition optimization in multi-networks as an integer programming problem.

Preprocessing: IP problem allows only linear relationships in the objective and constraints. Since the location of the execution engine, concrete web services and their locations are all unknown, non-linear relationships will be introduced when formulating the problem that violates the IP rules. Fortunately, in our problem, the valid locations for the execution engine are generally limited. If the location of the execution engine is given, the optimization problem for services composition in multi-networks could be formulated as an IP problem. Thus, we could use the IP technique to get an optimum solution for each possible location of the execution engine and obtain a global optimum solution finally.

Once the location of the execution engine is specified, the transmission time between the service/user and the engine could be prepared using our preprocessing program.

Objective function: The decision variables are defined as: z_{ij}: = 1 when the jth service is selected for the ith service class, $\forall i \in A, \forall j \in S_i$; 0 otherwise, where $A = \{x \mid x <= n, x \in N^+\}$

Then the objective function is defined as follows:

$$\min T(user) + T(cws) \tag{5}$$

Where:

$$T(user) = \sum_{p=1}^{k} \delta_{,p} * (duser_{p,e} + duser_{e,p})$$

$$T(cws) = \sum_{i=1}^{n} \sum_{je Si} z_{ij} * (dws_{ij,e} + T\_Exe(ws_{ij}) + dws_{e,ij})$$

The objective function contains two parts: T(user), the transmission time between the users and the execution engine, which is related to the distribution of the users; T(cws), the delay time of the composite web service, which consists of the service execution time and transmission time between the component services and the execution engine.

Allocation constraint: For each service class defined in the process, only one concrete web service will be bound to it, which is guaranteed by the allocation constraint:

$$\sum_{je Si} z_{ij} = 1, \qquad \forall i \in A \tag{6}$$

QoS constraints: The end-to-end QoS requirements for the web service composition are specified by the users.

Budget constraint:

$$Price(cws) = \sum_{i}^{n} \sum_{je Si} z_{ij} * Price_{ij} <= B, \; B>0 \tag{7}$$

where, B is the budget and $Price_{ij}$ represents the execution price of web service ws_{ij}. The budget constraint indicates that the execution price of the composite web service should not exceed B.

Reputation constraint: Enhanced reputation discussed in previous section is adopted, considering it is more suitable for the multi-network environment.

$$Rep(cws) = \frac{1}{n} \sum_{i}^{n} \sum_{je Si} z_{ij} * RepE_{ij} >= R, \qquad R>0 \tag{8}$$

$$RepE_{ij} = (1-\delta) * Rep(NS_{ij}) + \delta * Rep_{ij}, \; \delta \in [0, 1] \tag{9}$$

where, R is the minimum reputation required, Rep_{ij} and $RepE_{ij}$ represent the reputation and enhanced reputation of the web service ws_{ij} respectively, while $Rep(NS_{ij})$ is the reputation for the network where ws_{ij} locates.

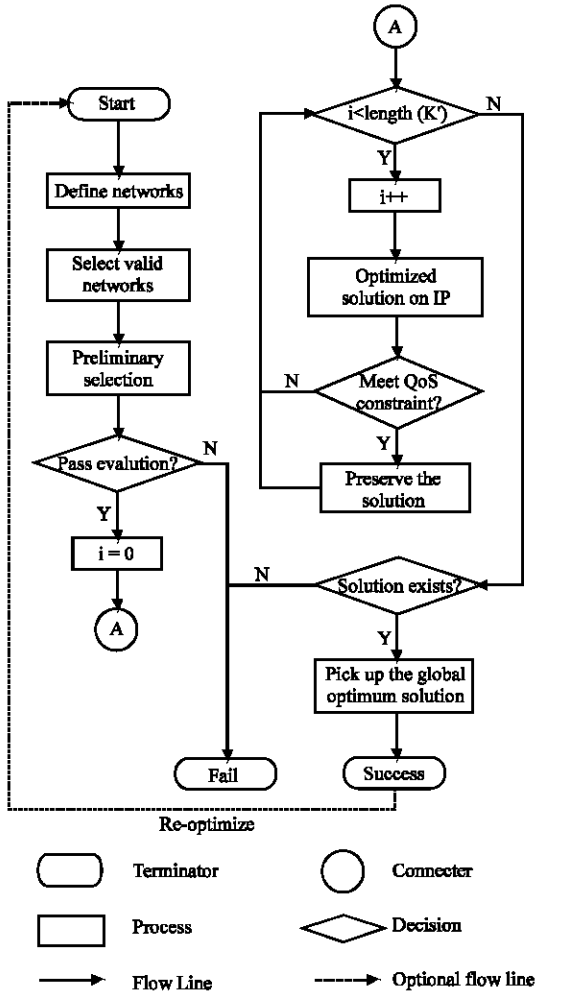

Fig. 4: Optimization process of services composition in multi-network environment

Reliability constraint: The reliability of composite web service is aggregated as the product of the reliabilities of the component services, which is non-linear:

$$Rel(cws) = \prod_{i=1}^{n} (\prod_{je Si} Rel_{ij}^{Zij}) \tag{10}$$

By applying the logarithm function, this constraint could be linearized:

$$ln(Rel(cws)) = \sum_{i=1}^{n} ln(\prod_{je Si} Rel_{ij}^{Zij}) = \sum_{i=1}^{n} \sum_{je Si} z_{ij} * ln(Rel_{ij})$$

let $Rel'_{ij} = ln(Rel_{ij})$, Rel'(cws) = ln(Rel(cws)), then:

$$Rel'(cws) = \sum_{i=1}^{n} \sum_{j \in Si} z_{ij} * Rel'_{ij} >= S, \quad S>0 \qquad (11)$$

where, S is the reliability threshold and Rel_{ij} represents the reliability for web service ws_{ij}.

Figure 4 is the flow chart for the whole optimization process.

EXPERIMENTAL EVALUATION

Evaluation methodology: The purpose of the evaluation is to validate that our IP-based services selection method achieves better performance in multi-network environment within reasonable computation time. Other two services selection methods are introduced for comparison:

Naive approach by exhaustive search: This approach attempts to find the optimum solution via exhaustive search. Since the target problem is NP-hard, which cannot be solved in polynomial time, the computation cost of this method is expected to increase very fast, which makes it only suitable for problems of rather limited size.

Traditional IP-based services selection method: As far as we know, traditional IP-based services selection method for services composition is first proposed in (Zeng *et al*., 2004), which solves the classical QoS-aware service composition problem very well but is not designed with the multi-network environment in mind.

We have created a number of test cases for the evaluation. Each test case represents a QoS aware service composition problem for multi-networks: the process definition consists of n abstract service classes, m candidate services for each class and they are distributed in k networks, the number of valid networks for the execution engine is v and the user distribution is also given. By varying these parameters, we conducted extensive experiments in respect of computation cost and effectiveness and recorded the experimental results.

Experiment settings: The experiments were conducted on a LENOVO machine with 2 Intel Duo 2.33GHz processors and 2 GB RAM. The machine is running under Windows XP SP3, with Java 2 Standard Edition V1.6.0. The open source integer linear programming system lpsovle 5.5 (Berkelaar and Notebaert, 2009) was used for solving the IP problems.

We designed and implemented a simulation program in Java to generate the test cases. Qualities values of candidate web services were randomly generated with reasonable Gaussian distributions empirically. The services and users are distributed among the networks randomly. Also, the transmission times between the networks were randomly generated with a Gaussian distribution. The simulation program also performs the tasks of security-based selection and preparing lpslove-compatible data for further processing.

Computation cost evaluation: The series of experiments in this section aims at evaluating the computation cost of our method by comparing with the naïve and traditional approaches. When applying the traditional method, we reduced the multi-network service composition problem into traditional QoS-aware service composition problem by ignoring the cross-network transmission times while keeping other conditions unchanged. The intention is to evaluate the performance impair when apply the IP-based method in multi-network environment. For each test case, we executed each selection method 10 times and computed the average computation cost.

In Fig. 5, we compared the performance of our approach for multi-network services composition with the other two methods. The figure plots the computation cost for each test case. In the experiments, the number of networks was fixed to 20 and the number of valid networks for deploying the engine was fixed to 3. For (a), we varied the number of abstract service classes from 10 to 100 with steps of 10 while the number of candidate

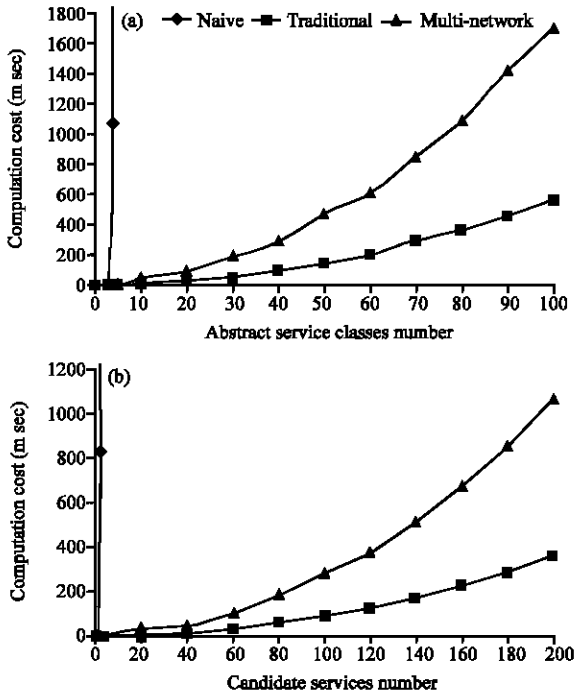

Fig. 5: Computation cost w.r.t. (a) the number of abstract service classes (b) the number of candidate services

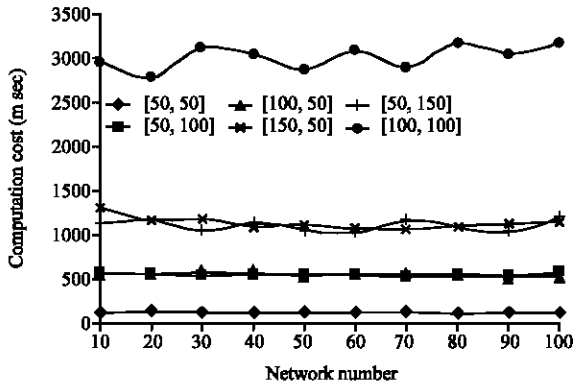

Fig. 6: Computation cost w.r.t the number of network

services for each abstract service class was fixed to 50; and for (b) we varied the number of candidate services from 20 to 200 with steps of 20 while the number of abstract service classes was fixed to 20.

In both experiments for all approaches, the computation cost increases with the growing of the number of service classes and candidate services. As expected, the result showed that the computation cost of naïve approach became very high even for problems of small scale. The results also indicated that in all test cases, the computation cost of our approach is approximately 3 times that of the traditional approach. For example, in the test case with 80 service classes, 50 candidate service for each class, 20 networks and 3 valid networks for deploying the engine, the computation time is 1080ms for our approach and 359ms for traditional approach, where 1080/359 ˜ 3.0. This is because in our optimizing process, the IP method is executed for each valid network for deploying the engine (in the test cases, it was fixed to be 3), which implied the computation cost of our approach is the computation cost of the traditional approach multiplied by the number of the valid locations of the engine. Since the number of valid networks for deploying the engine would be generally limited as we discussed in previous section, the computation cost of our approach should also be acceptable.

We also evaluated the impact of network number on the computation cost for various problem scales. In Fig. 6, the computation costs were plotted against the number of networks which varied from 10 to 100 with steps of 10. The number of valid networks for deploying the engine was fixed to 10 and each line in the figure uniquely corresponded with a pair of service class number and candidate service number (e.g., for the line [50, 100], the number of service classes was 50 and the number of candidate services was 100). In all these scales in aspect of the number of service classes and candidate services,

Table 2: Available web services in the illustrating test case

Candidate service ID	Service class ID	Execution time (m sec)	Price (cents)	Enhanced reputation	Reliability (%)	Network ID
ws1	1	120	6.0	8.0	99.80	1
ws2	1	110	9.0	8.1	99.85	1
ws3	1	100	7.0	7.5	99.95	2
ws4	2	200	19.0	7.0	99.88	2
ws5	2	220	16.0	6.0	99.91	2
ws6	2	245	17.0	6.8	99.91	3

Table 3: Transmission time between networks for the illustrating test case

Network ID	1	2	3
1	0	100	25
2	100	0	90
3	75	80	0

the results indicates that the computation cost remains relatively constant as the number of network grows, which meant our approach is scalable with regard to the network number.

Effectiveness evaluation: The second series of experiments in this section aims at evaluating the effectiveness of our approach by comparing with the traditional approach. Abundant test cases are performed to evaluate the impact of various factors on the effectiveness of our approach, such as the number of abstract services in the process, the number of candidate services for each service class, the number of networks, the network delay and service execution time ratio and the number of valid networks for the engine. For each test case, we executed both selection methods. The traditional approach is unaware of the multi-network information and its result has to be mapped to the real environment (i.e., multi-network environment) to get the realistic value. We use a simple case to illustrate the process.

Table 2 is the list of available web services in the test case, in which six web services belong to two service classes distributed among three networks. QoS values for each service are also given in the table.

Table 3 presents the transmission times between networks. The value in the cell of row m, column n represents the transmission time from network m to network n. As discussed in previous section, intra-net transmission time is negligible, so the values in the diagonal cells are all zero.

The process definition in this test case is very simple, which consists of the service classes 1 and 2 in sequence and the QoS constraints for the composite web service are the following:

B = 25 (budget),
R = 7.0 (reputation),
S = (ln99.8) * (ln99.8) (reliability)

when applying the traditional approach that ignores the multi-network information, the solution is ws1 for service

class 1 and ws4 for service class 2, where the objective value is 320 m sec. The solution is further processed to obtain the real value for the multi-network environment. We use T_{trad} to denote the real value, then:

$$Ttrad = \sum_{p=1}^{k} \delta_p * (duser_{p,e} + duser_{e,p}) + \sum_{i=1}^{n} \sum_{jeSi} z_{ij} * (dws_{ij,e} + T\_Exe(ws_{ij}) + dws_{e,ij}) \quad (12)$$

where

$$\sum_{i=1}^{n} \sum_{jeSi} z_{ij} * T\_Exe(ws_{ij})$$

is achieved by the traditional IP approach, which is 320 in this case. And we assume the engine is located in the first network (it is randomly chosen) and the user distribution is 0.3, 0.4, 0.3, for network 1, 2 and 3, respectively. Then, we have $T_{Trad} = 110 + (100 + 320 + 100) = 630$ m sec.

When applying our approach, the result for multi-network environment is straightforward. We assume network 1 and 3 are valid for deploying the engine. Then the optimum solution is achieved when the engine is deployed in network 3 and ws1, ws6 are chosen for service class 1 and 2, respectively. The objective value is 563 m sec and we use T_{multi} to denote it.

In order to quantify the effectiveness our approach, we define the optimization rate as:

$$OR = \frac{(T_{trad} - T_{multi})}{T_{trad}} \quad (13)$$

where, higher OR value indicates our approach is more effective. In the illustrating case, we have OR = 10.63%.

In the rest of this section, we present the results of several experiments that show the impact of various factors on the optimization rate quantitatively.

In Fig. 7, we plotted the OR values for each test case. For in Fig. 7a, we varied the number of service classes from 10 to 100 with steps of 10. There are six lines in the figure and each line corresponds with a pair of the number of candidate service and networks. For example, line 50 (100) is for the experiment setting where the number of candidate services is 50 and the number of networks is 100. For in Fig. 7b we varied the number of candidate services from 20 to 200 with steps of 20. There are also six lines in the figure and each line corresponds with the number of service classes and networks similarly. For both (a) and (b), we set the number of valid networks for deploying the engine to 10.

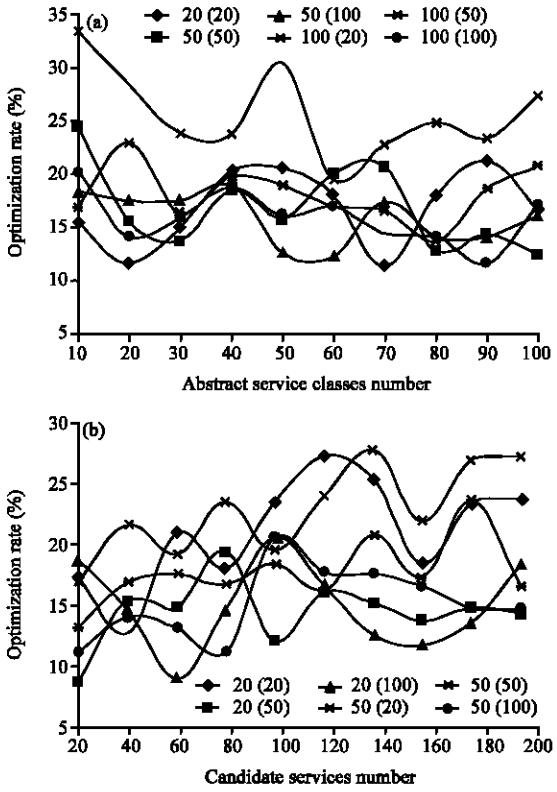

Fig. 7: Optimization rate w.r.t. (a) the number of abstract service classes (b) the number of candidate services

Figure 7 shows the achieved optimization rate is between 10 and 30% in all test cases for different number of service classes, candidate services as well as the networks, which indicates the general advantages of our approach.

In Fig. 8, we evaluated the impact of the ratio between network delay time and service execution time on the optimization rate. The name of each line indicates the experiment setting, for example, line [20, 50] (20) is conducted in the condition that the number of service classes is 20, the number of candidate services is 50 and the number of networks is 20. The OR values are plotted against N/(S+N) ratio, which is defined as:

$$N/(S+N) = \frac{Avg(Tran)}{Avg(Tran) + Avg(Exe)} \quad (14)$$

where, Avg (Tran) is the average value of transmission time between all networks and Avg(Exe) is the average execution time of all web services.

The overall service delay time consists of transmission time and service execution time. Figure 8 shows that the OR value increases with the N/(S+N) ratio for all test cases. The result indicates that our approach is

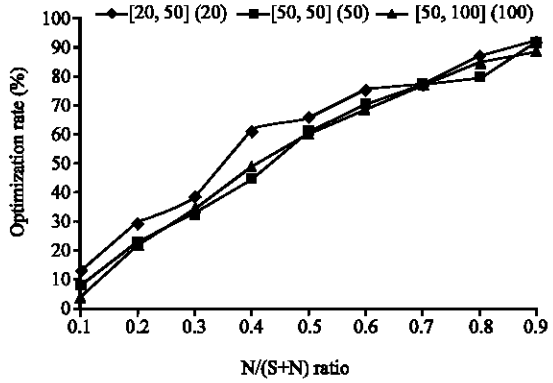

Fig. 8: Optimization rate w.r.t. N/(S+N) ratio

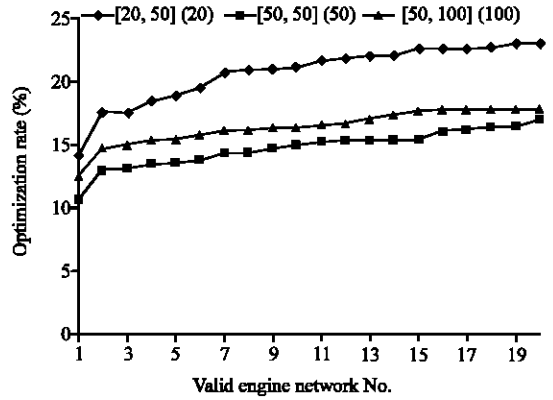

Fig. 9: Optimization rate w.r.t. the number of valid networks for deploying the engine

more effective when the transmission time dominates the delay time.

The last experiment evaluates the impact of the number of valid networks for deploying the engine. Figure 9 shows that for all test cases, expanding the number of valid engine networks can result in the improvement of the OR values effectively at first, but when the number reaches a certain threshold, the increase in OR value becomes relatively slow. Thus, the decision maker should allocate appropriate number of networks valid for deploying the engine appropriately to get better optimization rate while avoiding unnecessary cost.

DISCUSSION

QoS is one of the most important considerations when selecting web services for enterprise level applications. Consequently, QoS aware service selection has been an active research topic since the emerging of web services technologies. Ran (2003) introduces a web service discovery model enhanced with QoS, where a web service is chosen based on the functional as well as QoS requirements. This study also presents a web service QoS category according to the type of questions the particular QoS can address, such as QoS related to runtime, transaction support. Menascé (2002) discusses QoS issues in web services from the perspective of both users and providers and in his subsequent article (Menascé, 2004), he uses an simple example to illustrate the concept of QoS aggregation for composite web services based on the workflow structures, such as sequential, parallel structures.

As the widespread adoption of SOA and web service technology in recent years, many web services with overlapping or identical functionalities but different QoS properties appear. QoS aware service selection for the composition thus becomes an urgent technology and is studied extensively. The QoS optimization methods for web service composition can be classified into local and global approaches. The local optimization approach is done at the task level, where component web services are selected for each task separately (Liu *et al*., 2004; Zeng *et al*., 2004; Ardagna and Pernici, 2005). The advantages of this approach lie in its simplicity and efficiency, but it fails to guarantee the end user's QoS constraints and can not find global optimal solution.

On the contrary, global optimization technique aims at finding global optimum solution for the composition which meets the end user's QoS constraints. QoS values for the composite web service can be aggregated from QoS values of its component services, where different QoS dimensions have different aggregation rules. Jaeger *et al.* (2004, 2005) systematically present the aggregation methods for common QoS dimensions (e.g., execution time, cost and reputation) in web service composition using workflow patterns.

Since this global optimization problem is NP hard in nature (Ardagna and Pernici, 2005), it is impractical to obtain the solution by enumerating exhaustively. Researchers have applied optimization techniques to solve the problem.

As far as we know, Zeng *et al.* (2003, 2004) firstly introduces the IP method for the QoS aware web service composition problem. In the papers, the web service composition is modeled as the state chart and split into multiple execution paths to eliminate conditional branching. Then the IP method is used to achieve an optimum execution plan. Based on Zeng' works, Ardagna and Pernici (2005, 2007) improve the IP based approach for service composition. They point out the limitation of (Zeng *et al*., 2004) approach in which each execution path is optimized separately and the global plan is achieved by composing separate solutions according to the frequency

of execution. To overcome the defect, the objective in their IP formulation is defined as the weighted average of the scores of execution paths where weights are given by execution path frequency. Ardagna and Pernici (2007) extend their previous work by introducing advanced techniques, such as loop peeling, negotiation and web service dependencies constraints allowing the execution of stateful web services. Rosenberg *et al.* (2009) presents the VRESCo runtime that supports an end-to-end approach for QoS-aware service composition. In their approach, the functional and QoS requirements are specified using a domain-specific language called Vienna Composition Language (VCL). The optimization process employs both the constraint programming and integer programming methods to search the best solution within the boundaries given by the user constraints.

Other techniques are also applied to optimize the QoS-aware services composition optimization, such as genetic programming (Canfora *et al.*, 2004, 2005), negotiation (Ren *et al.*, 2009), workflow partition strategy (Jang *et al.*, 2006).

However, all above-mentioned methods are designed for classical QoS aware service composition, which does not consider multi-network environment. Yang *et al.* (2009) proposes an approach to improve SOA performance in multi-network environment. The optimum deployment strategy for service components is achieved via hill-climbing clustering algorithm based on connectivity strength with parameter, frequency and deployment space. However, their optimization approach is designed from the perspective of software components deployment, which does not involve the concept of web service composition. In contrast, our optimization method in this paper is for QoS aware service composition in multi-network environment, which has different optimization goal and has to consider the end-to-end QoS constraints.

CONCLUSIONS

In this study, we model the QoS-aware service composition problem for the multi-network environment, discuss its QoS model and propose an IP-based approach for searching optimum solutions.

As more and more services composition appears on the Internet, which presents new requirements in performance and security, the classical QoS-aware services composition methods fall short of achieving an optimum solution. Our modeling and approach are a generalization of the classical ones, which characterize the cross-organization and web-scale services composition by introducing the concept of multi-network. The multi-network method is superior to the classical ones in the following aspects: the transmission time caused by connection conditions and security checking mechanisms between different networks is modeled appropriately, the deployment location of the execution engine can be chosen among several networks, the selection of web services and execution engine deployment network is security-aware. The experiments show that our approach is effective.

The transmission time between networks is not constant and tends to change frequently, which will be researched in our following work. Also the correlations of reliabilities and other QoS properties between the component services will be investigated. Furthermore, we will do the corresponding work for the decentralized service composition in multi-network environment.

CONCEPTS AND NOTATIONS

Concepts and notations used in this study are defined here.

Entities: Entities involved in the optimizing process include the web service, execution engine and user, which are denoted as ws_{ij}, e and u respectively. For ws_{ij}, i is the index of the abstract service class and j is the index of the concrete service in an abstract service class.

Networks:

$$NS = \{ns_1, ns_2, \ldots ns_p, \ldots, ns_k\}$$

where, NS is the set of networks, ns_p is the pth network and k is the number of the networks.

- **NS(ws_{ij}):** The network where web service ws_{ij} is located. Simplified as NS_{ij} if no ambiguity arises
- **Trust(ns_i):** The degree of trustworthiness for the network ns_i, represented by an integer in the range 1-10, where a higher value means this network is more trustable

User_dist = $\{\delta_1, \delta_2, \ldots, \delta_p, \ldots, \delta_k\}$, where,

$$\sum_{p=1}^{k} \delta_p = 1$$

and $\delta_p \in [0, 1]$. The distributions of the user among the networks are denoted by the percentage.

Service class and composition:

$S_i = \{ws_{i1}, ws_{i2}, \ldots ws_{ij}, \ldots\}$: the abstract service class

- **Sensitivity (S_i):** The sensitivity of an abstract service class, represented by an integer in the range 1-10, where a higher value means the service is more sensitive. The sensitivity value is assigned by the process owner according to the business logic implemented by the service and security policies. And we use Sensitivity(e) to denote the sensitivity value of the execution engine

Delay time

Transmission time: The transmission time is determined by the connection condition between the source and destination networks assuming the intranet transmission time is negligible as discussed in previous section. In addition, the time for carrying out security policies of related domains accounts for part of the transmission time. Note transmission time of two networks is not a symmetrical property, which is caused by the difference in the uplink and downlink speed as well as the different security policies. We differentiate the transmission time into four types.

- **$dws_{ij,e}$:** The transmission time from web service ws_{ij} to the engine
- **$dws_{e,ij}$:** The transmission time from the engine to web service ws_{ij}
- **$duser_{p,e}$:** The transmission time from the users in the network ns_p to the engine
- **$duser_{e,p}$:** The transmission time from the engine to the users in the network ns_p
- **T_Exe(ws_{ij}):** The execution time of the web service ws_{ij}
- **T(cws):** The delay time of the composite web service, which consists of transmission time and component services execution time

REFERENCES

Andrews, T., F. Curbera, H. Dholakia, Y. Goland and J. Klein *et al.*, 2003. Business process execution language for web services version 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel/.

Ardagna, D. and B. Pernici, 2005. Global and local qos guarantee in web service selection. Proceedings of Business Process Management Workshops, Sept. 5, Springer Verlag, Berlin, Germany, pp: 32-46.

Ardagna, D. and B. Pernici, 2007. Adaptive service composition in flexible processes. IEEE Trans. Software Eng., 33: 369-384.

Atkinson, B., G. Della-Liaera, S. Hada, M. Hondo and P. Hallam-Baker *et al.*, 2002. Web services security. Microsoft, IBM and Verisign. http://www.ibm.com/developerworks/library/specification/ws-secure/.

Berkelaar, K.E.M. and P. Notebaert, 2009. lpSolve: Open source (mixed-integer) linear programming system. Sourceforge.

Binder, W., I. Constantinescu and B. Faltings, 2006. Decentralized orchestration of composite web services. Proceedings of 2006 IEEE International Conference on Web Services, Sept. 18-22, Chicago, Illinois, USA., pp: 869-876.

Canfora, G., M.D. Penta, R. Esposito and M.L. Villani, 2004. A lightweight approach for QoS-aware service composition. Proceedings of the 2nd International Conference on Service Oriented Computing, Nov. 15-19, ACM, New York, USA., pp: 36-47.

Canfora, G., M.D. Penta, R. Esposito and M.L. Villani, 2005. An approach for QoS-aware service composition based on genetic algorithms. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, June 25-29, ACM, New York, pp: 1069-1075.

Chafle, G.B., S. Chandra, V. Mann and M.G. Nanda, 2004. Decentralized orchestration of composite web services. Proceedings of the 13th International Conference World Wide Web (WWW), May 17-20, ACM, New York, USA., pp: 134-143.

Clark, J., C. Casanave, K. Kanaskie, B. Harvey, N. Smith, J. Yunker and K. Riemer, 2001. ebXML business process specification schema version 1.01. UN/CEFAC-T and OASIS. http://www.ebxml.org/specs/ebBPSS.pdf.

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, 2002. Unraveling the web services web: An introduction to SOAP, WSDL and UDDI. IEEE Internet Comput., 6: 86-93.

Fan, J.C. and S. Kambhampati, 2005. A snapshot of public web services. SIGMOD Record, 34: 24-32.

Hines, B., 2006. The (XML) threat is out there. IBM Websphere Dev. Technical J.,

Jaeger, M.C., G. Rojec-Goldmann and G. Muhl, 2004. QoS aggregation for web service composition using workflow patterns. Proceedings of 8th IEEE International Enterprise Distributed Object Computing Workshop, Sept. 20-24, IEEE, Piscataway, USA., pp: 149-159.

Jaeger, M.C., G. Rojec-Goldmann and G. Muhl, 2005. QoS aggregation in web service composition. Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service, Mar. 29-Apr. 1, IEEE, Piscataway, USA., pp: 181-185.

Jang, J.H., D.H. Shin and K.H. Lee, 2006. Fast quality driven selection of composite web services. Proceedings of 4th IEEE European Conference on Web Services, Dec. 4-6, IEEE, Piscataway, USA., pp:87-98.

Keller, A. and H. Ludwig, 2003. The WSLA framework: Specifying and monitoring service level agreements for web services. J. Network Syst. Manage, 11: 57-81.

Li, Y., Y. Liu, L.J. Zhang, G. Li, B. Xie and J.S. Sun, 2007. An exploratory study of web services on the internet. Proceedings of IEEE International Conference on Web Services, Jul. 9-13, IEEE, Piscataway, USA., pp: 380-387.

Liu, Y., A.H. Ngu and L. Zeng, 2004. Qos computation and policing in dynamic web service selection. Proceedings of 13th International Conference World Wide Web (WWW), May 17-20, ACM, New York, NY, USA, pp: 66-73.

Menascé, D.A., 2002. QoS issues in web services. IEEE Internet Comput., 6: 72-75.

Menascé, D.A., 2004. Composing web services: A QoS view. IEEE Internet Comput., 8: 88-90.

Papazoglou, M.P., 2003. Service-oriented computing: Concepts, characteristics and directions. Proceedings of 4th International Conference on Web Information Systems, Dec. 10-12, IEEE, Piscataway, USA., pp: 3-12.

Ran, S., 2003. A model for web services discovery with QoS. ACM SIGecom Exchanges, 4: 1-10.

Rao, J. and X. Su, 2005. A Survey of Automated Web Service Composition Methods. In: Lecture Notes in Computer Science, Rao, J. and S. Xiaomeng (Eds.). Vol. 3387, Springer, Berlin, pp: 43-54.

Ren, K.J., N. Xiao, J.Q. Song, C. Yang, M. Zhu and J.J. Chen, 2009. Gradual removal of qos constraint violations by employing recursive bargaining strategy for optimizing service composition execution path. Proceedings of the IEEE International Conference on Web Services, Jul. 6-10, IEEE, Piscataway, USA., pp: 485-492.

Rosenberg, F., P. Celikovic, A. Michlmayr, P. Leitner and S. Dustdar, 2009. An End-to-End approach for Qos-aware service composition. Proceedings of 13th IEEE International EDOC Conference, Sept. 1-4, IEEE, Piscataway, USA., pp: 151-160.

Wang, Y. and J. Vassileva, 2007. A review on trust and reputation for web service selection. Proceedings of 27th International Conference on Distributed Computing Systems Workshops, Jun. 25-29, IEEE, Piscataway, USA., pp: 25-25.

Wikipedia, 2009. Linear programming. http://en.wikipedia.org/wiki/Linear_programming.

Wolsey, L., 1998. Integer Programming. John Wiley and Sons, New York, USA.

Xignite Inc., 2009. Subscription. http://www.xignite.com/Support/FAQ.aspx?faqtype=Company&faqcat=Subscription.

Yang, X.H., J.F. Li and J.G. Yang, 2009. Optimizing SOA performance in multi-network environment based on hill climbing clustering algorithm. J. Zhejiang Univ: Eng. Sci.

Yin, K.T., B. Zhou, S. Zhang, Y.X. Chen and D.X. Chu, 2008. An effective approach to select trustable web services. Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing, Oct. 12-14, IEEE, Piscataway, USA., pp: 1-6.

Zeng, L., B. Boualem and D. Marlon, 2003. Quality driven web services composition. Proceedings of the 12th International Conference on World Wide Web, May 20-24, Hungary, pp: 411-421.

Zeng, L., B. Benatallah, A. Ngu and M. Dumas, 2004. QoS-aware middleware for web services composition. IEEE Trans. Software Eng., 30: 311-327.

Zhang, J., N. Zhang and L.J. Zhang, 2007. Auction-based pricing model for web service providers. Int. J. Web Services Res., 3: 82-107.