An Adaptive Heuristic Approach for Distributed QoS-based Service Composition

Jing Li[1,2], Yongwang Zhao[2], Min Liu[2], Hailong Sun[2], Dianfu Ma[1,2]
State Key Lab of Software Development Environment[1]
Institute of Advanced Computing Technology[2]
Beihang University
Beijing, China
{lijing, zhaoyw, liumin, sunhl}@act.buaa.edu.cn dfma@buaa.edu.cn

***Abstract*—QoS-based service selection becomes a commonly accepted procedure to support rapid and dynamic web service composition. In this paper, we study the problem of QoS-based service selection in distributed QoS management environments where QoS values of alternative services are maintained by distributed QoS registries. A distributed heuristic approach is proposed to solve the problem efficiently with a high approximation ratio, and enable adaptability in distributed cross-organization environments with data privacy protection and a low cost of communication. The proposed approach consists of four stages in which variable elimination is used to reduce the size of the problem; constraint decomposition allows performing service selection independently on each QoS registry; supplementary service selection and concentrated optimization improve the approximation ratio. Performance analyses and simulation experiments show that the proposed approach performs efficiently with close-to-optimal results and fits well to distributed QoS management environments.**

Keywords-Web Services; QoS; service composition; service selection

I. INTRODUCTION

Web service composition provides an effective solution for integrating business applications across different organizations. In recent years, researchers have been increasingly interested in exploring approaches for rapid and dynamic web service composition.

Quality of service (QoS) refers to various non-functional properties of web services [16], such as response time, availability, price, as well as some domain-specific QoS attributes [7] (e.g. a traffic reporting service may have QoS attributes such as precision and refresh frequency). QoS management infrastructure has been widely studied in recent years [6, 12, 15, 25, 27]. There are two types of QoS management: centralized management and distributed management. Distributed QoS management infrastructure [6, 12, 15] is widely considered to be able to provide better scalability and performance, thus be more feasible to support cross-organization web service applications than centralized approaches.

With the growing of functional equivalent web services that have different QoS attributes, QoS-based service selection becomes a commonly accepted procedure to support rapid and dynamic web service composition. Let us consider a travel planning system that combines the tasks of flight booking, travel booking, hotel booking, and car rental which are to be executed according to a given workflow structure. Alternative services for each task are provided by different service providers (e.g. Air Europe and Lufthansa flight booking services) with different levels of quality. A traveler may desire to obtain a travel plan with the cheapest price, the highest popularity, or maybe a compromise between the two. Moreover, the traveler may specify constraints as follows:

- Destination: Riccione.
- Budget: below $3000.
- Hotel: 4-star or better.
- Overall execution time: less than 10 minutes.
- Atomicity of task execution (or transaction support [13]): either all tasks are successfully completed or all of them are cancelled.

In such applications, alternative services are provided by various institutions that belong to different regions or management domains. QoS values of these alternative services may be maintained by distributed QoS registries because of business boundaries and easy maintenance. The travel planning system has to select services by interacting with distributed QoS registries, such that user's objective is maximized while the aggregated QoS values of the selected services match the constraints. Figure 1 gives a conceptual overview of distributed QoS-based service selection. The user requests a *composite service* that is described in a workflow description language (e.g. BPEL [22]). For each

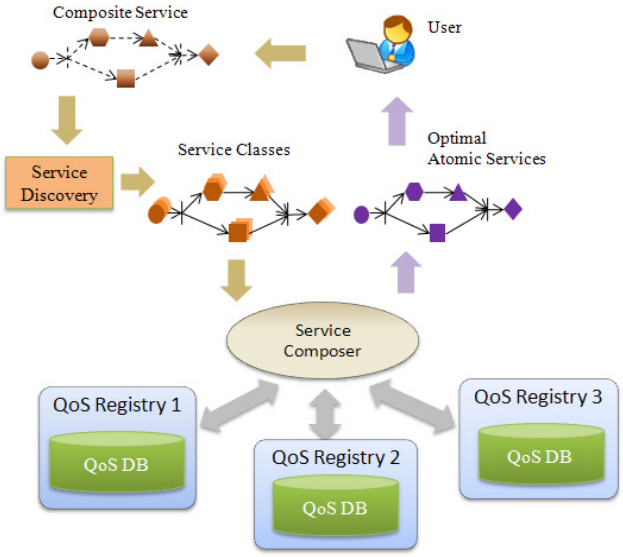

Figure 1. Overview of Distributed QoS-based Service Selection.

task in the *composite service*, a set of available atomic services (named as a *service class*) are located through the service discovery infrastructure (e.g. UDDI [21]). QoS values of each atomic service set are maintained by a group of distributed QoS registries. The goal of distributed QoS-based service selection is to select one *optimal atomic service* from each set of atomic services, such that the desired utility function is maximized and the aggregated QoS values satisfy the end-to-end QoS constraints.

Finding a solution for the above problem is known to be NP-hard [9]. An efficient service selection approach with low time complexity is required to achieve high performance and scalability. Moreover, due to business boundaries, data sharing among distributed cross-organization QoS registries may be restricted (or require payment) to protect data privacy and proprietary interests. Hence, the execution of an effective service selection approach should not rely on sharing a large amount of QoS data or a high frequency of interactions among the distributed QoS registries.

The existing approaches can be classified into two categories: optimization approaches and heuristic approaches. Optimization approaches refer to methods aiming at finding the optimal solution based on user's QoS requirements. Some of these methods [3, 26, 27] suffer from poor scalability due to the exponential time complexity. Other more efficient optimization approaches [8, 10] are difficult to be applied in a distributed environment, due to the requirement of global data visibility. Heuristic approaches are proposed to find a close-to-optimal solution with a polynomial time complexity. Among them, local selection strategies [6, 15] perform efficiently and fit well to distributed environments. However, these methods fail in handling global constraints. Other heuristic approaches [1, 25] that succeed in handling global constraints and performing efficiently. However, these approaches rely on sharing a large amount of data or a high cost of communication that is not desirable in distributed cross-organization environments.

To address the challenges above, we propose a distributed heuristic approach. The basic idea is to decompose the global optimization problem into local ones, then perform service selection on each QoS registry independently, finally collect the local results and perform centralized optimization. In our approach, variable elimination is used to reduce the size of the problem; constraint decomposition allows performing service selection independently on each QoS registry; supplementary service selection and concentrated optimization improve the approximation ratio. Performance analyses and simulation experiments show that the proposed approach solves the problem efficiently with close-to-optimal results and fits well to distributed QoS management environments.

The rest of the paper is organized as follows. In the next section we discuss the related works. In Section 3, we introduce the system model and give a problem statement. Our heuristic approach is presented in Section 4. In Section 5, Performance analyses and simulation experiments are performed to compare our solution with existing solutions. Finally, Section 6 gives conclusions and future work.

II. RELATED WORK

In recent years, QoS-based service selection in service-oriented applications has been studied by many researchers. The existing approaches can be classified into two categories: optimization approaches and heuristic approaches. Optimization approaches refer to methods aim at finding the optimal solution. Mathematical models such as multidimensional multi-choice knapsack problem (MMKP) [17] have been used to model this problem [3, 26, 27]. In the works of Zeng et al. [26, 27], linear programming methods [9] are used to find the optimal solution. Similar to this approach, Ardagna et al. [3] studies an approach for service selection with QoS constraints in global view. These optimization methods are very effective when the size of the problem is small. However, these methods suffer from poor scalability due to the exponential time complexity. To achieve higher scalability, Genetic Algorithms [19] and Dynamic Programming [5] are used to find optimal solution more efficiently. Canfora et al. [8] propose a scalable approach by adopting genetic algorithms. In [10], Gao et al. transform the optimization problem into the problem of selecting the longest path in a weighted multistage graph and adopt dynamic programming to solve the problem. However, due to the requirement of global data visibility, these methods are difficult to be applied in a distributed environment. Heuristic approaches are proposed to find a close-to-optimal solution with a polynomial time complexity. In [6] and [15], local selection strategies are put forward to match the needs of distributed environments. Despite of high efficiency and fitness to distributed environments, these approaches are not suitable to handle global constraints. Other heuristic approaches that succeed in handling global constraints are proposed to solve the problem with a low time complexity. In [25], the problem of QoS-based service selection is modeled in two ways: the combinatorial model and the graph model. A heuristic algorithm is proposed for each model. The time complexity of the heuristic algorithm (MCSP-K) for the graph model is exponential. The heuristic algorithm (WS_HEU) for the combinatorial model achieves a polynomial time complexity. However, the iterative service selection strategy used in WS_HEU is not suitable for distributed QoS-based service selection due to a high frequency of interactions. Alrifai et al. [1] design a hybrid QoS-based service selection approach by combining global optimization with local selection. They define "quality levels" for each constraint, and then apply mixed integer programming (MIP) [9] technique to choose a set of optimal "quality levels" as local constraints. This hybrid approach is able to reach a low time complexity. However, solving the MIP model of choosing "quality levels" depends on sharing a large amount of data among QoS registries that is not desirable in distributed cross-organization environments.

III. SYSTEM MODEL AND PROBLEM STATEMENT

In our system model, we assume that QoS values are maintained by a group of distributed QoS registries that can communicate with the service composer. As a basis, this paper focuses on the sequential composition structure.

Solutions for other structures of web service composition can be extended from the proposed approach by handling more QoS aggregation formulas [14].

A. Definitions

The following definitions are used in the rest part of this paper:

Atomic service (component service) (s): A concrete web service which provides services to users independently without rely on other web services. It is the basic unit in web service composition. Each atomic service is associated with a QoS vector $Q(s)=\{q^1(s),q^2(s),...,q^v(s)\}$, which contains v QoS attributes such as price, availability and reliability. These QoS attributes can be classified into two types: positive and negative QoS attributes. The values of positive attributes need to be maximized (e.g. throughput and availability), whereas the values of negative attributes need to be minimized (e.g. price and response time).

Service class (S_i): A service class $S_i=\{s_{i1},s_{i2},...,s_{im}\}$ is the set of alternative atomic services (*alternative services*) that provide the same functionality and interfaces but differ in QoS attributes. m is the number of alternative services in each service class. QoS values of these atomic services are maintained by different QoS registries. Suppose there are e QoS registries. $S_i^d=\{s_{i1}^d,s_{i2}^d,...s_{in(i,d)}^d\}$ is a subset of service class S_i (S_i^d might equal to S_i as well), the QoS values of which are maintained by d-th QoS registry, $n(i,d)$ is the number of atomic services in S_i^d.

Composite service (S): A composite service is a 2-tuple *(S, P)*, where $S=\{S_1,S_2,...,S_n\}$, is the set of service classes, where n is the number of service classes. P is the structure information of S, which can be specified by a service oriented workflow description language, such as BPEL [22].

Constraints (C): $C=\{c^1,c^2,...,c^r\}$ is the set of user's end-to-end QoS constraints, which contains r elements. $C_n=\{c_n^1,c_n^2,...,c_n^{n(C_n)}\},C_n\subseteq C$ is the set of numerical QoS constraints (such as the constraints on budget and availability in the travel planning scenario) in C, where $n(C_n)$ is the number of elements in C_n. These numerical constraints can be expressed in terms of upper (or lower) bounds (e.g. Total price should be lower than \$2500). $C_{bs}=\{c_{bs}^1,c_{bs}^2,...,c_{bs}^{n(C_{bs})}\},C_{bs}\bigcup C_n=C$ is the set of non-numerical constraints (in this paper, we study two types: Boolean and String) in C, where $n(C_{bs})$ is the number of elements in C_{bs}.

Constraint attributes (Q_c): Each constraint $c^k\in C$ is associated with a QoS attribute $q^k\in Q$. $Q_c=\{q_c^1,q_c^2,...,q_c^{n(Q_c)}\},Q_c\subset Q$ is the set of constraint attributes. $Q_n=\{q_n^1,q_n^2,...,q_n^{n(C_n)}\},Q_n\subseteq Q_c$ is the set of numerical constraint attributes. $Q_{bs}=\{q_{bs}^1,q_{bs}^2,...,q_{bs}^{n(C_{bs})}\},Q_{bs}\bigcup Q_n=Q_c$ is the set of non-numerical constraint attributes.

Objective attributes (Q_o): The objective attributes $Q_o=\{q_o^1,q_o^2,...,q_o^{n(Q_o)}\},Q_o\subset Q$ is the set of QoS attributes that the user desire to maximize.

Objective quality: Given an atomic service s_{ij}, objective quality is an integrated value of its objective attributes. Objective quality of s_{ij} is computed as follows:

$$O(s_{ij})=\sum_{k=1}^{n(Q_o)}q_o^k(s_{ij})\cdot\omega_k \qquad \sum_{k=1}^{n(Q_o)}\omega_k=1 \tag{1}$$

where ω_k is the weight for objective attribute q_o^k, it represents user's priority on q_o^k. Then, the objective quality of service class S_i is computed as follows:

$$O(S_i)=\sum_{s_{ij}\in S_i}O(s_{ij})\cdot x_{ij} \qquad \sum_{s_{ij}\in S_i}x_{ij}=1 \qquad x_{ij}\in\{0,1\} \tag{2}$$

$x_{ij}=1$, when s_{ij} is selected to be the best atomic service for S_i.

Utility function ($U(S)$): Utility function of composite service $U(S)$ is the aggregation of $O(S_i)$. $U(S)$ is computed as follows:

$$U(S)=\sum_{S_i\in S}\sum_{s_{ij}\in S_i}O(s_{ij})\cdot x_{ij} \qquad \sum_{s_{ij}\in S_i}x_{ij}=1 \qquad x_{ij}\in\{0,1\} \tag{3}$$

B. Problem Statement

The constrained optimization problem we are addressing can be stated as follows:

For a given composition service *(S, P)*, the utility function $U(S)$, and the constraints C, selecting an atomic service s_{ij} from each service class S_i such that the utility function $U(S)$ is maximized, and the constraints C are satisfied. Formally, the problem is stated as follows:

$$\text{Maximize} \quad \sum_{S_i \in S} \sum_{s_{ij} \in S_i} O(s_{ij}) \cdot x_{ij}$$

$$\text{Subject to} \quad \forall k, 1 \le k \le n(C_n), \sum_{S_i \in S} \sum_{s_{ij} \in S_i} q^k_{ij\_n} x_{ij} \le c^k_n$$

$$\forall h, 1 \le h \le n(C_{bs}), \forall s_{ij} \in S, q^h_{ij\_bs} x_{ij} = c^h_{bs} \qquad (4)$$

$$\sum_{s_{ij} \in S_i} x_{ij} = 1$$

$$x_{ij} \in \{0,1\}$$

IV. APPROACH DESCRIPTION

Our distributed heuristic service selection approach consists of four stages. Figure 2 shows the cross-component process of this approach.

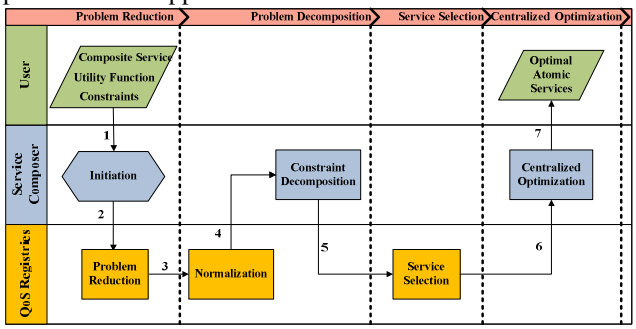

Figure 2. Cross-component Process.

In the first stage, problem reduction is performed on each QoS registry. Variable elimination is used to reduce to size of the problem. In the second stage, the global optimization problem is approximately decomposed into local ones that are to be solved on the distributed QoS registries. In the third stage, service selection is performed independently on each QoS registry. A set of optimal atomic services that combines an essential service and a set of supplemental services are selected for each service class. The final stage of our approach is to perform centralized optimization on the service composer.

A. *Problem Reduction*

The first step of our approach is to perform variable elimination on each QoS registry based on the non-numerical constraints.

For each atomic service s_{ij}, if any non-numerical QoS attribute of which do not satisfy the relevant constraint ($\exists k, 1 \le k \le n(C_{bs}), q^k_{ij\_bs} \ne c^k_{bs}$), delete s_{ij} from its service class S_i ($S_i = S_i - \{s_{ij}\}$).

After variable elimination, the size of the constrained optimization problem is reduced. This stage can be skipped if there is no non-numerical constraint in C.

B. *Problem Decomposition*

1) *Normalization*

As mentioned before, QoS values are maintained by a group of distributed QoS registries. In order to void local optimization, the next step is to generate normalized values for the objective and constraint attributes. Objective attributes are normalized to positive values that need to be maximized. Constraint attributes are normalized to negative values that should satisfy the normalized constraints as upper bounds. For each atomic service s_{ij}, normalized QoS values are generated as follows:

For positive objective attributes and negative constraint attributes:

$$q^k(s_{ij}) = \begin{cases} \dfrac{q^k(s_{ij}) - q^k_{\min}(S_i)}{q^k_{\max}(S_i) - q^k_{\min}(S_i)}, & \text{if } q^k_{\max}(S_i) - q^k_{\min}(S_i) \ne 0 \\ 1, & \text{if } q^k_{\max}(S_i) - q^k_{\min}(S_i) = 0 \end{cases}$$

$$q^k_{\max}(S_i) = \max_{s_{ij} \in S_i} q^k(s_{ij}),\ q^k_{\min}(S_i) = \min_{s_{ij} \in S_i} q^k(s_{ij}) \qquad (5)$$

$$1 \le k \le v$$

For negative objective attributes and positive constraint attributes:

$$q^k(s_{ij}) = \begin{cases} \dfrac{q^k_{\max}(S_i) - q^k(s_{ij})}{q^k_{\max}(S_i) - q^k_{\min}(S_i)}, & \text{if } q^k_{\max}(S_i) - q^k_{\min}(S_i) \ne 0 \\ 1, & \text{if } q^k_{\max}(S_i) - q^k_{\min}(S_i) = 0 \end{cases}$$

$$q^k_{\max}(S_i) = \max_{s_{ij} \in S_i} q^k(s_{ij}),\ q^k_{\min}(S_i) = \min_{s_{ij} \in S_i} q^k(s_{ij}) \qquad (6)$$

$$1 \le k \le v$$

Then, we compute the normalized objective quality as follows:

$$O_n(s_{ij}) = \sum_{k=1}^{n(Q_o)} q^k_o(s_{ij}) \cdot \omega_k \qquad (7)$$

For each service class S_i, the average objective quality is computed as follows:

$$O_{ave}(S_i) = \frac{\sum_{s_{ij} \in S_i} O_n(s_{ij})}{n(S_i)} \qquad (8)$$

where $n(S_i)$ is the number of atomic services in S_i. These average objective qualities are then sent to the service composer to perform constraint decomposition.

2) *Constraint Decomposition*

The idea of constraint decomposition is taken from [1]. Different from their method where constraints are decomposed through solving a MIP model among the service composer and QoS registries, the process of constraint decomposition in our approach is performed independently on the service composer. This more loosely-coupled design enables higher adaptability in distributed QoS management environments.

In this stage, for each service class S_i, a set of objective quality bounds denoted as $B^k_{obj} = \{b^k_{1\_obj}, b^k_{i\_obj}, ..., b^k_{n\_obj}\}$ are generated by decomposing the numerical constraints according to the average objective quality $O_{ave}(S_i)$.

$$b^k_{i\_obj} = \frac{O_{ave}(S_i)}{\sum_{S_i \in S} O_{ave}(S_i)} \times c^k_n, 1 \le k \le r \qquad (9)$$

Then, we find a set of quality bounds $B^k = \{b_1^k, b_2^k, ..., b_n^k\}$ such that the values of these quality bounds are mostly close to the values of objective quality bounds. The problem of finding quality bounds is modeled as a linear-constrained non-linear programming model [13]. Formally:

$$\begin{aligned} &\text{Minimize} && \sum_{i=1}^{n} (b_i^k - b_{i\_obj}^k)^2 \\ &\text{Subject to} && \min_{s_{ij} \in S_i} q_{ij}^k \le b_i^k \le \max_{s_{ij} \in S_i} q_{ij}^k \\ & && \sum_{i=1}^{n} b_i^k \le c_n^k \\ & && 1 \le k \le n(C_n) \end{aligned} \tag{10}$$

Because of linear constraints and a small number of variables, this model can be solved efficiently by using any non-linear programming technique [13]. After solving this model, we get a set of quality bounds. These quality bounds are then sent to each QoS registry as the local constraints for service selection.

C. Service Selection

In this stage, service selections are performed in parallel on the distributed QoS registries.

1) Essential Service Selection

For each service class S_i^d on d-th QoS registry, we find one essential atomic service $s_{i\_ess}^d$ which has the maximum objective quality and satisfy all the quality bounds:

$$\begin{aligned} & O(s_{i\_ess}^d) = \max_{s_{ij}^d \in S_i^d} O(s_{ij}^d) \\ & q^k(s_{i\_ess}^d) \le b_i^k \\ & 1 \le k \le n(C_n) \end{aligned} \tag{11}$$

Then, we get a set of essential atomic services $S_{ess}^d = \{s_{1\_ess}^d, s_{2\_ess}^d, ..., s_{n(d)\_ess}^d\}, 1 \le n(d) \le n$, where $n(d)$ is the number of essential atomic services on d-th QoS registry (A QoS registry may not have QoS data for all service classes). The selection of essential service for each service class ensures that at least one solution could be found in the final stage.

2) Supplementary service Selection

In order to improve the approximation ratio, for each service class S_i^d , we select a set of additional atomic services named supplementary services ($S_{i\_\sup}^d$). We assume that each QoS registry would be able to provide QoS information of a small number (e.g. no more than 5) of supplementary services. We choose supplementary services by finding atomic services, the constraint attributes of which are not all satisfy but are proximate to the quality bounds.

$S_{i\_up}^d \subset S_i^d$ is the set of atomic services whose constraint attributes do not all satisfy the quality bounds ($\exists k, 1 \le k \le n(C_n) : q^k(s_{ij\_up}^d) > b_i^k$). In $S_{i\_up}^d$, we compute the constraint attribute distance $d(s_{ij\_up}^d)$ for each atomic service $s_{ij\_up}^d$ as follows:

$$d(s_{ij\_up}^d) = \sum_{k=1}^{n(C_n)} (q^k(s_{ij\_up}^d) - b_i^k)^2 \tag{12}$$

Suppose we select α supplementary services for each service class on each QoS registry. For each service class S_i^d , we sort the atomic services by $d(s_{ij\_up}^d)$ in ascending order. Then the first α atomic services with the smallest constraint attribute distances are selected as the supplementary services $S_{i\_\sup}^d$.

Finally, for each service class S_i^d , we get a set of optimal services $S_{i\_opt}^d$ by combining the essential service with the set of supplementary services:

$$S_{i\_opt}^d = \{s_{i\_ess}^d\} \bigcup S_{i\_\sup}^d \tag{13}$$

These optimal services are then sent to the service composer to perform centralized optimization.

D. Centralized Optimization

The optimal service set S_{i_opt} for service class S_i is the union of optimal service set $S_{i\_opt}^d$ from each QoS registry:

$$\begin{aligned} & S_{i\_opt} = \bigcup_{d=1}^{e} S_{i\_opt}^d \\ & S_{opt} = \bigcup_{i=1}^{n} S_{i\_opt} \end{aligned} \tag{14}$$

where S_{opt} is the set of universal optimal services.

Then we use MIP model to select the best atomic services from S_{opt} . We use binary decision variable x_{ij} for each optimal atomic service $s_{ij\_opt} \in S_{i\_opt}$, such that $x_{ij} = 1$ if s_{ij_opt} is selected as the best atomic service for S_i , and $x_{ij} = 0$ otherwise. Formally the MIP model is:

$$\begin{aligned} &\text{Maximize} && \sum_{S_i \in S} \sum_{s_{ij} \in S_i} O(s_{ij\_opt}) \cdot x_{ij} \\ &\text{Subject to} && \forall k, 1 \le k \le n(C_n), \sum_{S_i \in S} \sum_{j \in S_i} q_{ij\_opt}^k x_{ij} \le c_n^k \\ & && \sum_{s_{ij} \in S_i} x_{ij} = 1 \\ & && x_{ij} \in \{0,1\} \end{aligned} \tag{15}$$

The number of variables in this model is $n \cdot e \cdot (\alpha + 1)$. We can assume that when $e \cdot (\alpha + 1) \ll m$, solving this MIP model is much more efficient than solving MIP models (where the number of variables is $m \cdot n$) in the optimization approaches [3, 26, 27].

After solving this model using any MIP solver method [8], we get one best atomic service for each service class as the final solution.

V. PERFORMANCE EVALUATION

The evaluation of performance consists of two parts: Performance analyses on data sharing and communication cost show that our approach fits well to the distributed

environments; Simulation experiments verify that our approach performs efficiently with close-to-optimal results.

A. Performance Analysis

1) Data Sharing

In our approach, data sharing among the service composer and QoS registries mainly occur in the second and the third stages.

In the second stage, statistical data are collected from QoS registries for problem decomposition. Maximum and minimum values of objective and constraint attributes are collected to generate normalized values; average objective qualities are collected to compute quality bounds. In the third stage, QoS values of optimal services selected on each QoS registry including essential services and supplemental services are sent to the service composer to perform centralized optimization. The number of optimal services provided by each QoS registry is $n \cdot (\alpha + 1)$, where n is the number of service classes, α is the number of supplementary services for each service class. α is a small integer which can be set by the owner of each QoS registry according to their data sharing policy.

Comparing to approaches that require collecting a large amount of QoS information, this design provides more adequate data privacy protection in distributed cross-organization environments.

2) Communication Cost

To evaluate the cost of communication, we count the number of messages required for interactions among the service composer and QoS registries.

Figure 3 illustrates the interaction process in this stage.

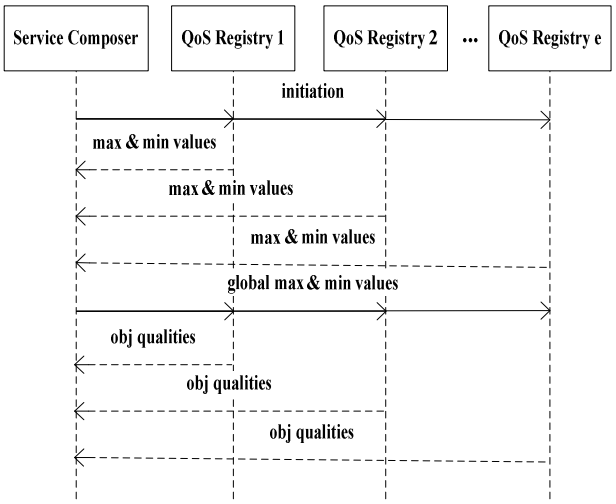

Figure 3. Interaction Process in Stage 2

In the first stage, the service composer sends the user's requirements to each QoS registry to initiate problem reduction. This requires *e* messages, where *e* is the number of QoS registries. In the second stage, to generate normalized QoS values and calculate the average objective qualities, statistical data needs to be collected from QoS registries. This takes 4 times of interactions between the service composer and each QoS registry.

Thus, the number of messages required in this stage is $4 \cdot e$. Then, sending quality bounds to each QoS registry requires *e* messages. In the third stage, service selection is performed independently on each QoS registry. Then sending optimal services to the service composer requires *e* messages. Finally, centralized optimization is performed on the service composer. Consequently, the total communication cost of our approach is $7 \cdot e$, which is independent of both the number of alternative services and the number of service classes. This provides high performance and scalability for large-scale distributed applications.

B. Experiments

In the following experiments, we evaluate the computation time and the quality of results of our approach by comparing them with the results of the optimization approach proposed in [3, 26, 27]. We use the label "Heuristic_ α " to refer to our approach with selection of α supplementary services for each service class, and the label "Optimal" to refer to the optimization approach. The approximation ratio is computed as follows:

$$approximation\ ratio = \frac{U(heuristic)}{U(optimal)} \times 100\% \qquad (16)$$

where $U(heuristic)$ is the value of utility function achieved by using our heuristic approach, $U(optimal)$ is the value of utility function achieved by using the optimal approach.

1) Experiment Settings

The experiments are conducted on a DELL Q9500 machine with 2 Intel Quad 2.83GHz processors and 4GB RAM. The machine is running under Windows XP and Java 1.6. The open source Mixed Integer Linear Programming (MILP) solver lp_solve version 5.5 [18] is used for solving the MIP model in both approaches. MATLAB Optimization Toolbox 4.3 [24] is used for solving the non-linear programming problem of quality bounds computation in our approach.

2) Experiment Parameters

The following 10 QoS attributes are used in the simulations. The values of numerical QoS attributes including *Price*, *Availability*, *Popularity*, *Response time*, and *Throughput* are generated randomly based on normal distribution [2]. The values of three Boolean QoS attributes including *Transaction support*, *Fault-tolerant support*, and *Encryption support* are randomly set to "True" or "False". The values of two String QoS attributes including *Location* and *Service provider* are randomly chosen from {Rome,

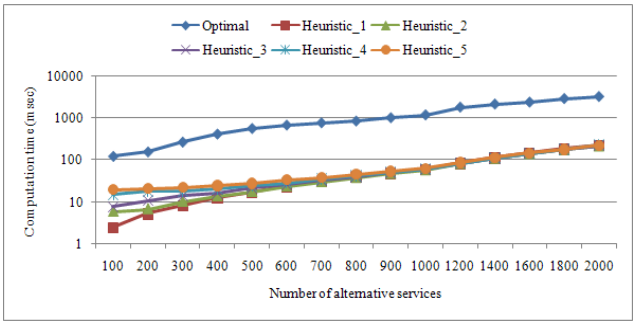

Figure 4. Computation time w.r.t. the number of alternative services

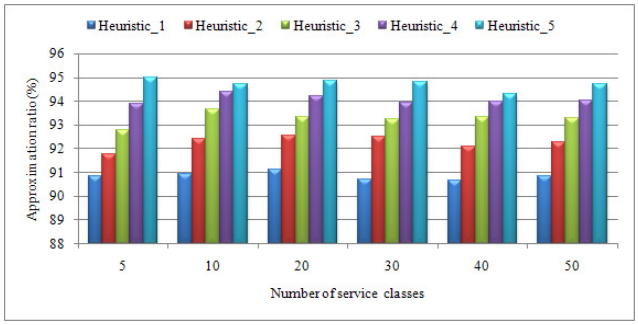

Figure 7. Approximation ratio w.r.t. the number of service classes

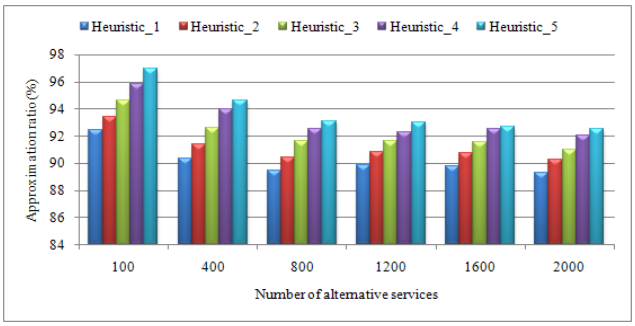

Figure 5. Approximation ratio w.r.t. the number of alternative services

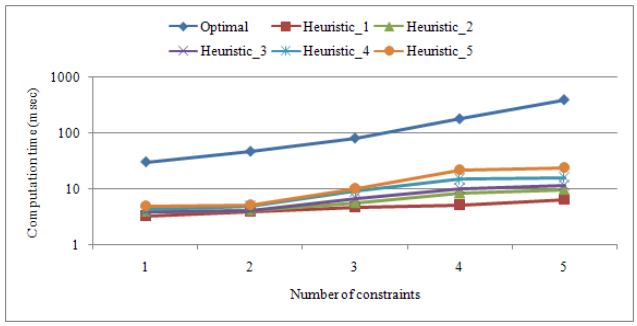

Figure 8. Computation time w.r.t. the number of constraints

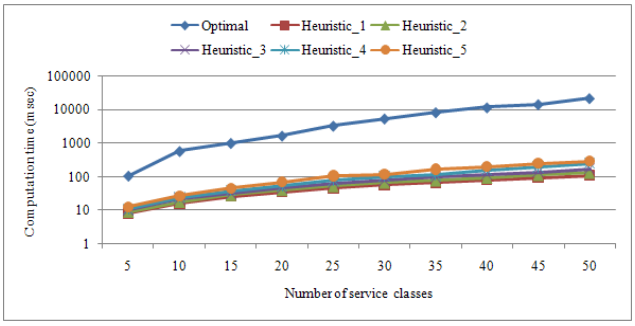

Figure 6. Computation time w.r.t. the number of service classes

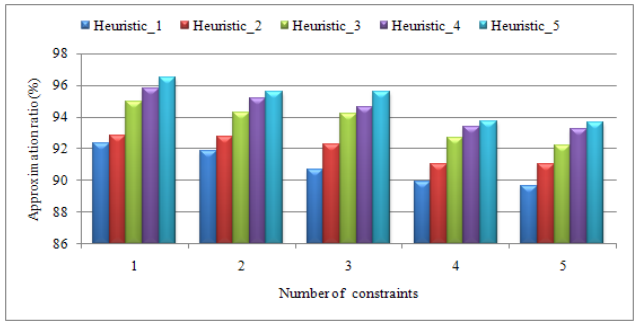

Figure 9. Approximation ratio w.r.t. the number of constraints

Venice, Milan, Riccione} and {BA, LH, VS, AY, IG}. We execute each approach 50 times and get the average value.

3) Experiment Results

In the first experiment, we study the computation time and approximation ratio with respect to an increasing number of alternative services. The number of alternative services per class varies from 100 to 2000. The number of service classes is fixed to 10. Alternative services are randomly allocated to 5 QoS registries. The number of supplementary services per service class varies from 1 to 5.

The results displayed in Figure 4 and Figure 5 show that, by increasing the number of alternative services, our approach perform much more efficiently with slowly increasing computation time comparing to the optimal approach (note that the vertical axis is on a logarithmic scale). Meanwhile, results of our heuristic approach reach around 92% approximation ratios.

In the second experiment, we study the computation time and approximation ratio with respect to an increasing number of service classes. The number of service classes varies from 5 to 50. The number of alternative services per class is fixed to 500. The other parameters are set in the same way as the last experiment.

The results displayed in Figure 6 and Figure 7 show that, by increasing the number of service classes our heuristic approach performs much more efficiently and achieves around 93% approximation ratios.

Finally, we evaluate the computation time and approximation ratio with respect to an increasing number of constraints. The number of constraints varies from 1 to 5. The number of alternative services per class is fixed to 500. The number of service classes is fixed to 10. The other parameters are set in the same way as the last experiment.

The results displayed in Figure 8 and Figure 9 show that, by increasing the number of constraints our heuristic approach performs much more efficiently and achieves around 92% approximation ratios.

VI. Conclusion and Future Work

This paper studies the problem of QoS-based web service composition in distributed environments where QoS values are maintained by a group of distributed QoS registries. A distributed heuristic approach is proposed to achieve the requirements on both problem-solving effectiveness and distributed-environment adaptability. Performance analyses and simulation experiments show that the proposed approach solves the problem efficiently while achieving close-to-optimal results, and effectively limit the interactions among the service composer and QoS registries. This is especially useful for distributed applications with time sensitive requirements. As a part of our future work, we will explore methods to handle more QoS aggregation formulas so as to support various structures of web service composition. We also intend to investigate how our approach can be extended to the situation where inter service conflicts and dependencies may impact on the aggregated QoS values of service compositions.

Acknowledgment

We acknowledge the reviewers for their useful comments and suggestions. This work was partially supported by the 863 program of China under Grant 2007AA010301, the Key Project of Chinese National Programs for Fundamental Research and Development under Grant 2005CB321901 and the NFSC program China National Natural Science Funds under Grant 60903149.

References

[1] M. Alrifai, and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition" Proc. the International World Wide Web Conference, 2009, pp. 881–890.

[2] B. L. Amstadter, Reliability mathematics: fundamentals, practices, procedures, McGraw-Hill Inc., 1971.

[3] D. Ardagna and B. Pernici, "Adaptive servicecomposition in flexible processes", IEEE Transactions on Software Engineering, vol. 33, pp.369–384, 2007.

[4] M. Avriel, Nonlinear Programming: Analysis and Methods. Dover Publishing, 2003.

[5] R. Bellman, Dynamic Programming, Princeton University Press, Dover paperback edition, 2003.

[6] B. Benatallah, Q. Z. Sheng, A. H. H. Ngu, and M. Dumas. "Declarative composition and peer-to-peer provisioning of dynamic web services". Proc. the International Conference on Data Engineering, 2002, pp. 297–308.

[7] G. Canfora, and R. Esposito, "A Lightweight Approach for QoS–Aware Service Composition", Proc. the 2nd International Conference on Service Oriented Computing, 2004, pp. 36–47.

[8] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms", Proc. Genetic and Evolutionary Computation Conference, June 2005, pp. .

[9] V. Chandru and M.R. Rao, Algorithms and Theory of Computation Handbook, CRC Press, 1999.

[10] Y. Gao, J. Na, B. Zhang, L. Yang, and Q. Gong, "Optimal Web Services Selection Using Dynamic Programming", Proc. IEEE Symp. on Computers and Communications, 2006, pp. 365 – 370.

[11] M. Garey and D. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, W.H. Freeman,1979.

[12] N. Gibelin, M. Makpangou, "Efficient and Transparent Web-Services Selection", LNCS, Springer, 2005, vol. 3826, pp. 527–532.

[13] J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques . Morgan Kaufmann, San Mateo, CA, 1992.

[14] M. C. Jaeger, G. Rojec-Goldmann, and G. M¨uhl, "QoS Aggregation for Web Service Composition using Workflow Patterns", Proc. the International Enterprise Distributed Object Computing Conference, 2004.

[15] F. Li, F. Yang, K. Shuang, and S. Su, "Q-peer: A decentralized qos registry architkecture for web services" Proc. the International Conference on Services Computing, 2007, pp. 145–156.

[16] Y. Liu, A.H. Ngu, and L.Z. Zeng, "QoS computation and policing in dynamic web service selection", Proc. the International World Wide Web Conference, 2004, pp. 66-73.

[17] S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations, New York: Wiley, 1990.

[18] K. E. Michel Berkelaar and P. Notebaert, Open source linear programming system, Sourceforge. http://lpsolve.sourceforge.net/.

[19] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

[20] G. L. Nemhauser and L. A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, New York, USA, 1988.

[21] OASIS, UDDI Version 3.0, 2004. http://uddi.org/pubs/uddi_v3.htm

[22] OASIS, Web services business process execution language, April 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[23] J. K. Patel and C. B. Read, Handbook of the Normal Distribution. New York: Dekker, 1982.

[24] The Math Works Inc., MATLAB Optimization Toolbox 4.3, http://www.mathworks.com/products/optimization/.

[25] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints", ACM Transactions on the Web, vol. 1, 2007.

[26] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition", Proc. the International World Wide Web Conference, 2003, pages 411–421.

[27] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition", IEEE Transactions on Software Engineering, vol. 30, pp. 311–327, 2004.