Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition

Mohammad Alrifai
L3S Research Center
University of Hannover
Germany
alrifai@L3S.de

Thomass Risse
L3S Research Center
University of Hannover
Germany
risse@L3S.de

ABSTRACT

The run-time binding of web services has been recently put forward in order to support rapid and dynamic web service compositions. With the growing number of alternative web services that provide the same functionality but differ in quality parameters, the service composition becomes a decision problem on which component services should be selected such that user's end-to-end QoS requirements (e.g. availability, response time) and preferences (e.g. price) are satisfied. Although very efficient, local selection strategy fails short in handling global QoS requirements. Solutions based on global optimization, on the other hand, can handle global constraints, but their poor performance renders them inappropriate for applications with dynamic and real-time requirements. In this paper we address this problem and propose a solution that combines global optimization with local selection techniques to benefit from the advantages of both worlds. The proposed solution consists of two steps: first, we use mixed integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints. Second, we use distributed local selection to find the best web services that satisfy these local constraints. The results of experimental evaluation indicate that our approach significantly outperforms existing solutions in terms of computation time while achieving close-to-optimal results.

Categories and Subject Descriptors

H.3.5 [**On-line Information Services**]: Web-based services; H.3.4 [**Systems and Software**]: Distributed systems

General Terms

Management, Performance, Measurement

Keywords

Web Services, QoS, Optimization, Service Composition

1. INTRODUCTION

The service-oriented computing paradigm and its realization through standardized web service technologies provide a promising solution for the seamless integration of business applications to create new value-added services. Industrial practice witnesses a growing interest in the ad-hoc service composition in the areas of supply chain management, accounting, finances, eScience as well as in multimedia applications. With the growing number of alternative web services that provide the same functionality but differ in quality parameters, the composition problem becomes a decision problem on the selection of component services with regards to functional and non-functional requirements.

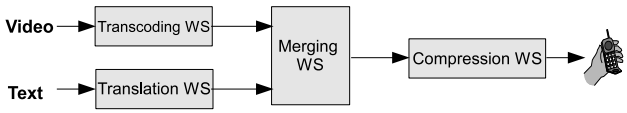

Figure 1: Web Service Composition Example

Consider for example the personalized multimedia delivery scenario (from [21]) in Figure 1. A smartphone user requests the latest news from a service provider. Available multimedia content includes a news ticker and topical videos available in MPEG 2 only. The news provider has no adaptation capabilities, so additional services are required to serve the user's request: a transcoding service for the multimedia content to fit the target format, a compression service to adapt the content to the wireless link, a text translation service for the ticker, and also a merging service to integrate the ticker with the video stream for the limited smartphone display. The user request can be associated with some end-to-end QoS requirements (like bandwidth, latency and price). The service composer has to ensure that the aggregated QoS values of the selected services match the user requirements at the start of the execution as well as during the execution. However, dynamic changes due to changes in the QoS requirements (e.g. the user switched to a network with lower bandwidth) or failure of some services (e.g. some of the selected services become unavailable) can occur at run-time. Therefore, a quick response to adaptation requests is important in such applications. The performance of the service selection middleware can have a great impact on the overall performance of the composition system.

Figure 2 gives a conceptual overview of the QoS-aware service composition problem. Given an abstract composition request, which can be stated in a workflow-like language (e.g. BPEL [19]), the discovery engine uses existing infrastructure (e.g. UDDI) to locate available web services for each task in the workflow using syntactic (and probably semantic) functional matching between the tasks and service descriptions. As a result, a list of candidate web services is obtained for each task. The goal of QoS-aware service selection middel-

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

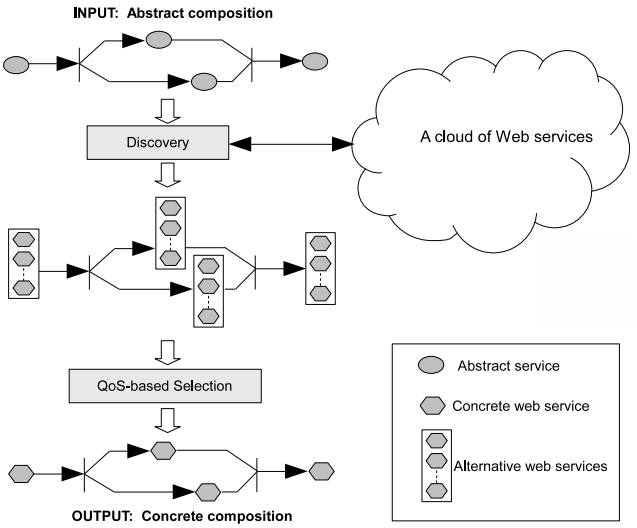

Figure 2: Conceptual Overview

ware is to select one component service from each list such that the aggregated QoS values satisfy the user's end-to-end QoS requirements. In service oriented environments, where deviations from the QoS estimates occur and decisions upon replacing some services has to be taken at run-time (e.g. in the multimedia application above), the efficiency of the applied selection mechanism becomes crucial. The focus of this paper is on the selection of web services based on their non-functional properties and the performance of the applied techniques.

Local Selection vs. Global Optimization

Two general approaches exist for the QoS-aware service composition: local selection and global optimization.

Local Selection: The local selection approach is especially useful for distributed environments where central QoS management is not desirable and groups of candidate web services are managed by distributed service brokers [8, 14]. The idea is to select one service from each group of service candidates independently on the other groups. Using a given utility function, the values of the different QoS criteria are mapped to a single utility value and the service with maximum utility value is selected. This approach is very efficient in terms of computation time as the time complexity of the local optimization approach is $O(l)$, where l is the number of service candidates in each group. Even if the approach is useful in decentralized environments, local selection strategy is not suitable for QoS-based service composition, with end-to-end constraints (e.g. maximum total price), since such global constraints cannot be verified locally.

Global Optimization: The global optimization approach was recently put forward as a solution to the QoS-aware service composition problem [24, 25, 5, 6]. This approach aims at solving the problem on the composite service level. The aggregated QoS values of all possible service combinations are computed, and the service combination that maximizes the aggregated utility value, while satisfying global constraints is selected. The global selection problem can be modeled as a *Multi-Choice Multidimensional Knapsack problem* (MMKP), which is known to be NP-hard in the strong sense [20]. Consequently, it can be expected that an optimal solution may not be found in a reasonable amount of time [16]. The exponential time complexity of the proposed solutions [24, 25, 5, 6] is only acceptable if the number of service candidates is very limited. Already in larger enterprises and even more in open service infrastructures with a few thousands of services the response time for a service composition request could already be out of the real-time requirements.

Contribution of the Paper

Since the QoS requirements (e.g. response times, throughput or availability) are only approximate, we argue that finding a "reasonable" set of services that avoid obvious violations of constraints at acceptable costs is more important than finding "'the optimal"' set of services with a very high cost. In addition, we advocate that the selection of component services should be carried out in a distributed fashion, which fits well to the open web service environment, where central management is not feasible.

The contribution of this paper can be stated as follows:

- A *distributed QoS computation model for web services*. Unlike existing solutions that model the QoS-aware service composition problem as a conventional global optimization problem, we exploit the special structure of the web service composition problem to reduce the cost of QoS optimization. The QoS optimization in our model is carried out by a set of distributed service brokers. The idea is to decompose QoS global constraints into a set of local constraints that will serve as a conservative upper/lower bounds, such that the satisfaction of local constraints by a local service broker guarantees the satisfaction of the global constraints.

- An *efficient QoS-aware service selection approach*. We propose an efficient and scalable mechanism for selecting web services for a given composition request from a collection of service candidates, such that the fulfillment of user's end-to-end QoS requirements and preferences can be ensured. By combining global optimization with local selection our approach is able to efficiently solve the selection problem in a distributed manner. Experimental evaluations show that our hybrid approach is able to reach close-to-optimal results much faster than existing 'pure' global optimization approaches.

The rest of the papers is organized as follows. In the next section we discuss related work. Section 3 introduces the system model and gives a problem statement. Our approach for efficient and distributed QoS-aware service selection is presented in Section 4. Performance analysis and experimental evaluations for comparing our solution against existing solutions are presented in Section 5. Finally, Section 6 gives conclusions and an outlook on possible continuations of our work.

2. RELATED WORK

The requirements for composition of web services can be stated in a workflow language such as Business Process Execution Language (BPEL) [19]. In [26, 9] ontology-based representations for describing QoS properties and requests were proposed to support semantic and dynamic QoS-based discovery of web services. Quality of service management has been widely discussed in the area of middleware systems [7, 11, 12, 13]. Most of these works focus on QoS specification and management. Recently, the QoS-based web service selection and composition in service-oriented applications has gained the attention of many researchers [24, 25, 5, 6, 15, 23]. In [15] the authors propose an extensible QoS computation model that supports open and fair management of QoS data. The problem of QoS-based composition is not addressed by this work. The work of Zeng at al. [24, 25] focuses on dynamic and quality-driven selection of services. The authors use global planning to find the best service components for the composition. They use (mixed) linear programming techniques [18] to find the optimal selection of component services. Similar to this approach Ardagna et al. [5, 6] extends the linear programming model to include local constraints. Linear programming 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 of the applied search algorithms [16]. In [23] the authors propose heuristic algorithms that can be used to find a near-to-optimal solution more efficiently than exact solutions. The authors propose two models for the QoS-based service composition problem: 1) a combinatorial model and 2) a graph model. A heuristic algorithm is introduced for each model. The time complexity of the heuristic algorithm for the combinatorial model (WS_HEU) is polynomial, whereas the complexity of the heuristic algorithm for the graph model (MCSP-K) is exponential. Despite the significant improvement of these algorithms compared to exact solutions, both algorithms do not scale with respect to an increasing number of web services and remain out of the real-time requirements. Any distributed implementation of these algorithms would raise a very high communication cost. The WS_HEU for example, is an improvement of the original heuristic algorithm for solving general *Multi-Choice Multi-dimensional Knapsack* problems named M-HEU [1]. The WS_HEU algorithm starts with a pre-processing step for finding an initial feasible solution, i.e. a service combination that satisfies all constraints but not necessarily is the best solution. A post-processing step improves the total utility value of the solution with one upgrade followed by one or more downgrades of one of the selected component services. Applying this algorithm in a distributed setting where the QoS data of the different service classes is managed by distributed service brokers would raise very high communication cost among these brokers to find the best composition. In this paper, we propose a heuristic algorithm that solves the composition problem more efficiently and fits well to the distributed environment of web services.

3. SYSTEM MODEL

In our model we assume that we have a universe of web services $\mathbb{S}$ which is defined as a union of *abstract service classes*. Each abstract service class $S_j \in \mathbb{S}$ (e.g. flight booking services) is used to describe a set of functionally-equivalent web services (e.g. Lufthansa and Qantas flight booking web services). In this paper we assume that information about service classes is managed by a set of service brokers as described in [15, 14]. Web services can join and leave service classes at any time by means of a subscription mechanism.

3.1 Abstract vs. Concrete Composite Services

As shown in Figure 2 we distinguish in the composition process between the following two concepts:

- An abstract composite service, which can be defined as an abstract representation of a composition request $CS_{abstract} = \{S_1, \ldots, S_n\}$. $CS_{abstract}$ refers to the required service classes (e.g. flight booking) without referring to any concrete web service (e.g. Lufthansa flight booking web service).

- A concrete composite service, which can be defined as an instantiation of an abstract composite service. This can be obtained by binding each abstract service class in $CS_{abstract}$ to a concrete web service s_j, such that $s_j \in S_j$. We use CS to denote a concrete composite service.

3.2 QoS Criteria

In our study we consider quantitative non-functional properties of web services, which can be used to describe the quality criteria of a web service [24, 15]. These can include generic QoS attributes like response time, availability, price, reputation etc, as well as domain-specific QoS attributes like bandwidth for multimedia web services as long as these attributes can be quantified and represented by real numbers. We use the vector $Q_s = \{q_1(s), \ldots, q_r(s)\}$ to represent the QoS attributes of service s, where the function $q_i(s)$ determines the value of the i-th quality attribute of s. The values of these QoS attributes can be either collected from service providers directly (e.g. price), recorded from previous execution monitoring (e.g. response time) or from user feedbacks (e.g. reputation) [15]. The set of QoS attributes can be divided into two subsets: 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). For the sake of simplicity, in this paper we consider only negative attributes (positive attributes can be easily transformed into negative attributes by multiplying their values by -1).

3.3 QoS Computation of Composite Services

The QoS value of a composite service is decided by the QoS values of its component services as well as the composition model used (e.g. sequential, parallel, conditional and/or loops). In this paper, we focus on the sequential composition model. Other models may be reduced or transformed to the sequential model. Techniques for handling multiple execution paths and unfolding loops from [10], can be used for this purpose.

The QoS vector for a composite service CS is defined as $Q_{CS} = \{q'_1(CS), \ldots, q'_r(CS)\}$. $q'_i(CS)$ represents the estimated value of the i-th QoS attribute of CS and can be aggregated from the expected QoS values of its component

services. In our model we consider three types of QoS aggregation functions: 1) summation, 2) multiplication and 3) minimum relation. Table 1 shows examples of these aggregation functions.

Aggregation type	Examples	Function
Summation	Response time	$q'(CS) = \sum_{j=1}^{n} q(s_j)$
	Price	
	Reputation	$q'(CS) = 1/n \sum_{j=1}^{n} q(s_j)$
Multiplication	Availability	$q'(CS) = \prod_{j=1}^{n} q(s_j)$
	Reliability	
Minimum	Throughput	$q'(CS) = \min_{j=1}^{n} q(s_j)$

Table 1: Examples of QoS aggregation functions

3.4 Global QoS Constraints

Global QoS constraints represent user's end-to-end QoS requirements. These can be expressed in terms of upper (and/or lower) bounds for the aggregated values of the different QoS criteria. As mentioned earlier, we only consider negative QoS criteria. Therefore in our model we only have upper bound constraints.

Definition 1. (Feasible Selection) Let $CS_{abstract}$ be a given composition request and $C' = \{c'_1, \ldots, c'_m\}, 0 \leq m \leq r$, be a vector of global QoS constraints on $CS_{abstract}$. Let CS be an instantiation of $CS_{abstract}$, in which a concrete web service is selected for each service class. We consider CS a feasible selection iff $q'(CS) \leq c', \forall c'_k \in C'$, i.e. all global constraints are satisfied.

3.5 Utility Function

In order to evaluate the multi-dimensional quality of a given web service a utility function is used. The function maps the quality vector Q_s into a single real value, to enable sorting and ranking of service candidates. In this paper we use a Multiple Attribute Decision Making approach for the utility function: i.e. the *Simple Additive Weighting (SAW)* technique [22]. The utility computation involves scaling the QoS attributes' values to allow a uniform measurement of the multi-dimensional service qualities independent of their units and ranges. The scaling process is then followed by a weighting process for representing user priorities and preferences. In the scaling process each QoS attribute value is transformed into a value between 0 and 1, by comparing it with the minimum and maximum possible value according to the available QoS information of service candidates. For a composite service $CS = \{S_1, \ldots, S_n\}$, the aggregated QoS values are compared with minimum and maximum possible aggregated values. The minimum (or maximum) possible aggregated values can be easily estimated by aggregating the minimum (or maximum) value of each service class in CS. For example, the maximum execution price of CS can be computed by summing up the execution price of the most expensive service candidate in each service class in CS. Formally, the minimum and maximum aggregated values of the k-th QoS attribute of CS are computed as follows:

$$Qmin'(k) = \sum_{j=1}^{n} Qmin(j,k) \quad (1)$$

$$Qmax'(k) = \sum_{j=1}^{n} Qmax(j,k)$$

with

$$Qmin(j,k) = \min_{\forall s_{ji} \in S_j} q_k(s_{ji}) \quad (2)$$

$$Qmax(j,k) = \max_{\forall s_{ji} \in S_j} q_k(s_{ji})$$

where $Qmin(j,k)$ is the minimum value (e.g. minimum price) and $Qmax(j,k)$ is the maximum value (e.g. maximum price) that can be expected for service class S_j according to the available information about service candidates of this class.

Now the utility of a component web service $s \in S_j$ is computed as

$$U(s) = \sum_{k=1}^{r} \frac{Qmax(j,k) - q_k(s)}{Qmax(j,k) - Qmin(j,k)} \cdot w_k \quad (3)$$

and the overall utility of a composite service is computed as

$$U'(CS) = \sum_{k=1}^{r} \frac{Qmax'(k) - q'_k(CS)}{Qmax'(k) - Qmin'(k)} \cdot w_k \quad (4)$$

with $w_k \in \mathbb{R}_0^+$ and $\sum_{k=1}^{r} w_k = 1$ being the weight of q'_k to represent user's priorities.

Definition 2. (Optimal Selection) The optimal selection for a given web service composition request $CS_{abstract}$ and a given vector of global QoS constraints $C' = \{c'_1, \ldots, c'_m\}, 0 \leq m \leq r$, is a feasible selection (according to Definition 1) with the maximum overall utility value U'.

However, finding the optimal composition requires enumerating all possible combinations of service candidates. For a composition request with n service classes and l service candidate per class, there are l^n possible combinations to be examined. Performing exhaustive search can be very expensive in terms of computation time and, therefore, inappropriate for run-time service selection in applications with many services and dynamic needs.

3.6 Problem Statement

The problem of finding the best service composition without enumerating all possible combinations is considered as an optimization problem, in which the overall utility value has to be maximized while satisfying all global constraints. Formally, the optimization problem we are addressing can be stated as follows:

For a given composition request $CS_{abstract} = \{S_1, \ldots, S_n\}$ and a given set of m global QoS constraints $C' = \{c'_1, \ldots, c'_m\}$, find an implementation $CS = \{s_1, \ldots, s_n\}$ by binding each S_j to a concrete service $s_j \in S_j$ such that:

1. The overall utility $U'(CS)$ is maximized, and
2. The aggregated QoS satisfy: $q'_k(CS) \leq c'_k, \forall c'_k \in C'$

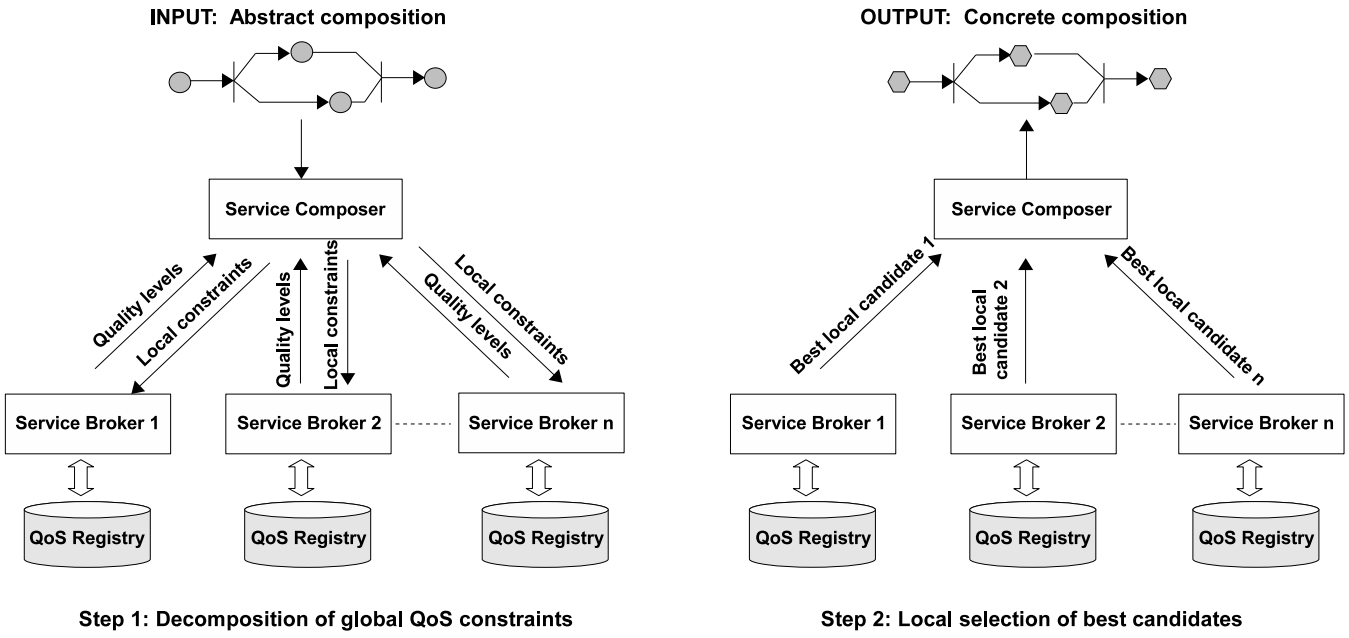

Figure 3: Distributed QoS-aware Service Selection

4. QOS-AWARE SERVICE COMPOSITION

The use of mixed integer programming [18] to solve the QoS-aware service composition problem has been recently proposed by several researchers [24, 25, 5, 6]. Binary decision variables are used in the model to represent the service candidates. A service candidate s_{ij} is selected in the optimal composition if its corresponding variable x_{ij} is set to 1 in the solution of the model and discarded otherwise. By re-writing (4) to include the decision variables, the problem of solving the model can be formulated as a maximization problem of the overall utility value given by

$$\sum_{k=1}^{r} \frac{Qmax'(k) - \sum_{j=1}^{n}\sum_{i=1}^{l} q_k(s_{ji}) \cdot x_{ji}}{Qmax'(k) - Qmin'(k)} \cdot w_k \qquad (5)$$

subject to the global QoS constraints

$$\sum_{j=1}^{n}\sum_{i=1}^{l} q_k(s_{ji}) \cdot x_{ji} \leq c'_k, 1 \leq k \leq m \cdot w_k \qquad (6)$$

while satisfying the allocation constraints on the decision variables as

$$\sum_{i=1}^{l} x_{ji} = 1, 1 \leq j \leq n. \qquad (7)$$

Because the number of variables in this model depends on the number of service candidates (number of variables = $n \cdot l$), this MIP model may not be solved satisfactorily, except for small instances. Another disadvantage of this approach is that it requires that the QoS data of available web services be imported from the service broker into the MIP model of the service composer, which raises high communication.

To cope with these limitations, we divide the QoS-aware service composition problem into two sub-problems that can be solved more efficiently in two subsequent phases. Figure 3 gives an overview on our approach. In the first phase, the service composer decomposes each global QoS constraints into local constraints on the component services level and sends these constraints to the involved service brokers. These constraints also include user's preferences, which are expressed in terms of weights of the QoS attributes. In the second phase, each service broker performs local selection to find the best component services that satisfy these local constraints. The two phases of our approach are described in the next subsections in more details.

4.1 Decomposition of Global QoS Constraints

To ensure the fulfillment of global QoS constraints in a service composition problem without enumerating all possible combinations of component web service, we decompose each QoS global constraint c' into a set of n local constraints $c_1, \ldots, c_n$ (n is the number of abstract service classes in the composition request). The local constraints serve as a conservative upper bounds, such that the satisfaction of local constraints guarantees the satisfaction of global constraints.

A naive decomposition algorithm would be to divide each global constraint c' into n equal local constraints such that: $c_j = c'/n$, $1 \leq j \leq n$. However, as different service classes can have different QoS value ranges, a more sophisticated decomposition algorithm is required. Furthermore, in order to avoid discarding any service candidates that might be part of a feasible composition, the decomposition algorithm needs to ensure that the local constraints are relaxed as much as possible while meeting global constraints. We solve this problem by modeling the QoS constraint decomposition problem as an optimization problem. The goal of this optimization problem is to find a set of local constraints for each service class that cover as many as possible service candidates, while their aggregation does not violate any of the global constraints. To this end, we divide the quality range of each QoS attribute into a set of discrete quality values, which we call *quality levels*. We then map each global