A Framework for Optimal Service Selection in Broker-based Architectures with Multiple QoS Classes

Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi
Università di Roma "Tor Vergata"
{cardellini,casalicchio}@ing.uniroma2.it, vgrassi@info.uniroma2.it

Raffaela Mirandola
Politecnico di Milano
mirandola@elet.polimi.it

Abstract

Service composition is one of the most promising advantages of the service-oriented paradigm. In a service market scenario, given a functional description of a service, different providers may offer diverse service implementations that match such a functional description but differ for some QoS attributes. A key point for the construction of a suitable composition is the selection of the services that best meet the QoS requirements of the composite service users.

In this paper, we consider a broker-based architecture for service composition, focusing on the service selection problem and assuming that the broker supports different QoS classes. We formulate the service selection as a constrained optimization problem, where each QoS class is modeled by suitable constraints. Differently from most of the existing approaches to service selection, in our approach the broker optimizes the overall QoS of a flow of requests rather than of a single request.

1. Introduction

The Service-oriented Architecture (SOA) paradigm foresees the creation of business applications from independently developed services. In this vision, providers offer similar competing services corresponding to a functional description of a service (the first referred to as *concrete services* and the latter as *abstract service*); these offerings can differ significantly in some Quality of Service (QoS) attributes [10]. On the other side, prospective users of services dynamically choose the best offerings for their purposes. Using the SOA paradigm to build applications, services can be dynamically selected and integrated at runtime, so enabling system properties like flexibility, adaptiveness, and reusability. In this context, the key point is to build applications through the composition of available services. This composition involves several activities: i) the selection of concrete services offering the required functionalities, ii) the definition of an integration schema yielding to the target application, and iii) the fulfillment of global QoS constraints, such as application response time and cost (a *global* constraint regards the whole composite service, while a *local* constraint refers to a single service component). Due to the high dynamism of the applications, the quality assessment cannot be deferred at the end of the development phase, but rather it should be incorporated in the selection and integration activities, to guarantee the construction of applications that satisfy the QoS constraints.

Current SOA approaches only partially address this global vision. While services are described and listed in public registries, there is little support for actually making quality-based service selection and integration. Therefore, QoS support for Web services has recently become a very active area of research and standardization, involving major challenges such as QoS-aware service description, composition, and selection (e.g., [6, 10]).

In this paper, we focus on the QoS-driven selection of concrete services, that plays an important role in the service composition problem. We propose a broker-based framework which allows an optimal service selection satisfying a set of global QoS constraints. In our framework, a broker offers to prospective users a single service that is actually realized as a composition of other services. Once discovered the alternative concrete services (and their values of the QoS attributes) that can be used to build the composition, the broker's main tasks include accepting requests for the advertised composite service and determining the optimal set of concrete services that maximizes some global benefit while satisfying the QoS constraints. We assume that the broker manages different, but fixed, global levels for the QoS attributes characterizing the operated composite service, thus supporting multiple classes of requestors.

We formulate the selection of the concrete services from a larger set of possible concrete services as a constrained multi-criteria optimization problem, considering various QoS attributes, such as execution time, cost, and reputation of the composite service. We introduce a graph model for the composite service considering both the abstract work-

flow and the concrete realization of the services and describe how the problem of optimally selecting the concrete services can be formulated on this graph. The broker can obtain the information needed to solve the optimization problem (e.g., the probability to take each branch in the workflow) by monitoring the composite service execution. We evaluate the computational cost of solving the optimization problem and show that it is acceptable even for a consistent number of concrete services, especially considering that in our approach the optimization problem needs to be solved again only when some parameters used in the formulation change significantly their values.

Our approach differs from previous works which have tackled the service selection as an optimization problem [1, 14, 15] in that our optimization is performed on a per-flow rather than per-request basis. In case of high volumes of service requests, per-request service selection approaches may suffer from scalability problems because of the computational overhead for solving the optimization problem for each single requests (also more times per request, according to some proposal [15]). On the contrary, in our approach the solution of the optimization problem (i.e., a given selection of concrete services) holds for all the requests in a flow, and is recalculated only when some significant event occurs (e.g., a change in the availability or the QoS values of the selected concrete services). Moreover, in our proposal the broker solves the optimization problem taking into account simultaneously the flows of requests generated by multiple requestors, with possibly different QoS constraints.

On the other hand, our approach is able to give only a statistical guarantee to each request that its QoS goal will be actually met. Hence, our approach is suitable for scenarios where "soft" rather than "hard" QoS goals must be satisfied. Moreover, being our approach more broker- rather than requestor-oriented, it does not allow the service requestors to define autonomously the QoS parameters for their QoS class as in [15].

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 describes the system model and Section 4 presents the formulation of the optimization problem for the service selection. Section 5 discusses some examples of solution of the optimization problem and analyzes its cost. Finally, Section 6 concludes.

2. Related Work

QoS-driven service discovery and selection have seen a flurry of recent research activity. The different approaches that have been followed so far span from the use of QoS ontology [9], to the definition of trust frameworks (e.g., [12]) and the use of data mining [7], the proposal of ad-hoc methods in some general framework (e.g., [11, 13]), and the exploitation of different kinds of optimization algorithms for the selection of concrete services in a composite service [1, 5, 8, 14, 15]. We briefly review works belonging to the latter class, since they are the closest to our approach.

Yu and Lin [14] discuss selection algorithms for multiple QoS attributes defining the problem as a multi-dimension multi-choice 0-1 knapsack one as well as a multi-constraint optimal path problem. Zeng et al. [15] present a global planning approach to select an optimal execution plan by means of integer programming. Ardagna and Pernici [1] model the service composition as a mixed integer linear problem where both local and global constraints are taken into account. Their approach is formulated as an optimization problem handling the whole application instead of each execution path separately. Claro et al. [8] propose the use of multiobjective optimization techniques to find a set of optimal Pareto solutions from which a requestor can choose. Finally, Canfora et al. [5] adopt a quite different strategy for optimal selection based on genetic algorithms.

3. System Model

In this section, we present the broker-based architecture for the service selection and introduce the graph model used for the optimal selection of the concrete services.

3.1. Broker-based Architecture

Figure 1 illustrates a high-level architectural view of the system model we consider. The *service broker* [11, 13] acts as an intermediary between service requestors and providers and its main goal is to select the concrete services of a composite service on the basis of some QoS constraints negotiated with the requestors. The broker is independently operated and maintained by a third party; it can be a Web service itself and advertise the offered composite service in a public registry [11]. Once the broker has discovered the concrete services which are candidates for the selection and have obtained their values of the QoS attributes (we do not focus on these issues, addressed in [11, 13]), its main tasks include: (i) accepting requests for a composite service and negotiating the QoS class with the requestors; (ii) determining the optimal service selection that satisfies the QoS constraints. In our architecture, the service broker manages different, but fixed, QoS classes for the operated composite service.

As the broker generally acts on behalf of a significant amount of requestors, it is able to identify recurrent requests for typical compositions of services, as well as usage patterns of these compositions. We assume that this knowledge is embodied in a workflow, which models both a given set of abstract services, needed to accomplish the composite service, and its usage pattern. In the considered scenario, the broker receives a flow of requests from the clients concerning the composite service. Each request includes the indica-

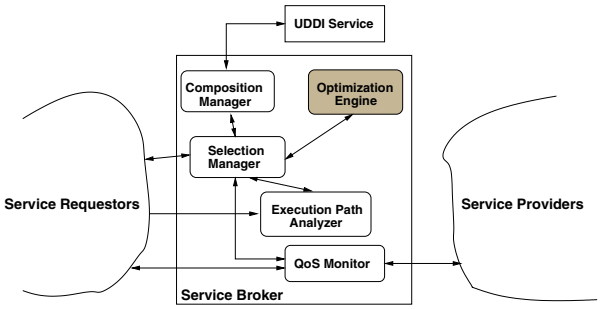

Figure 1. System architecture.

tion about some required QoS level. Requestors could also not be aware of the set of abstract services needed to carry out that composite service; however, this knowledge is necessary if we want to include the possibility for requestors to specify some local QoS requirement for single services in that set. Our framework may be used to consider both cases of requestor awareness and unawareness about the set of needed services. However, in the following we assume that the broker manages only global QoS constraints.

The main modules that the broker embeds are: the *Composition Manager*, the *Selection Manager*, the *Optimization Engine*, the *Execution Path Analyzer*, and the *QoS Monitor*. The main function of the Composition Manager is the service composition and the discovery of the candidate concrete services. The Selection Manager is the broker front-end to the service requestors. It is responsible for binding each request to the concrete services that meet the required QoS level; moreover, it may trigger a new solution of the optimization problem, when some relevant environmental change is detected. The Optimization Engine determines the selection of the concrete services by solving the optimization problem. In this paper we focus on methodologies underlying the implementation of this part of the broker architecture (shadowed in Figure 1). The solution of the optimization problem is used by the Selection Manager to determine the suitable concrete services for each QoS class. Finally, the Execution Path Analyzer and the QoS Monitor are responsible for collecting information about the service usage and performance, and about the requestor perceived performance. This information is used by the Selection Manager to find out whether a new solution of the optimization problem is required. In a dynamic environment, service providers can change their services at any time in order to remain competitive. This implies that there is no guarantee that the QoS obtained at run time for a particular concrete service is again a valid value. However, we do not focus on this problem here, but rather assume that the broker can use some QoS monitoring service.

3.2. Multi-Class Composite Service Model

The set of atomic services that form a composite service and the relationships among them can be represented using a workflow. We model it by a direct weighted graph, called *decisional service graph* (DSG). Each macro-node $i \in \mathcal{V}$ in this graph (depicted as a rectangular box in Figure 2) represents an atomic *abstract service*. The directed edge from the macro-node r to the macro-node s represents a sequencing constraint that is, it indicates that service r must complete before service s may begin. The DSG has usually a source node (modeling a root service) and a sink node (modeling a collector service); an execution of the composite service thus consists of the invocation of the services on a path from the source to the sink.

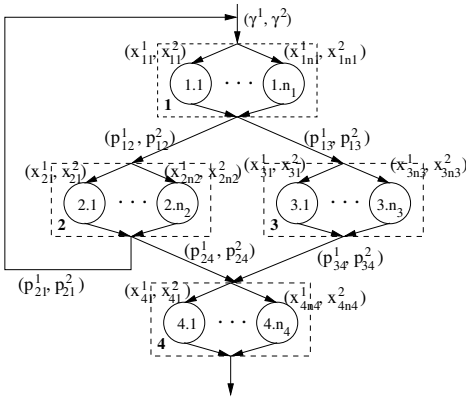

Figure 2. Example of DSG for two QoS classes.

Multiple edges exiting from a macro-node r are weighted by a probability, that provides a statistical information about the next abstract service required by a client of the composite service. The broker can use its observation of the execution patterns generated by the clients to estimate these probabilities[1]. As different QoS classes may lead to stressing different execution paths in the workflow, we assume that each edge from r to s has an associated probability vector $\mathbf{p}_{rs} = (p^1_{rs}, ..., p^m_{rs})$, where m is the number of different QoS classes ($m = 2$ in Figure 2) and p^k_{rs} denotes the probability that service s is executed after the completion of service r for the QoS class k. For each macro-node r, it holds that $\sum_{s \in succ(r)} p^k_{rs} = 1$ for each class k. In case of only one outgoing edge from r, the probability is equal to 1 and we omit its value in the graph.

Each DSG macro-node contains the *concrete services* (shown in Figure 2 as circles inside the rectangular box representing the abstract service), that correspond to specific

[1] In this probabilistic model of the workflow execution pattern, we do not include the parallel execution of services. We are currently working toward the inclusion of this execution pattern in our model.

implementations of a given abstract service. Concrete services for the same abstract service may be offered by different providers and may differ for the values of some QoS attribute. We denote by I_i the set of all concrete services that implement the abstract service $i \in \mathcal{V}$ (where $n_i = |I_i|$) and by $i.j \in I_i$ the j-th concrete service for i.

The goal of the broker is to select, for each QoS class, the concrete service $i.j$ that must be used to fulfill a request for the abstract service i. We model this selection by associating with each DSG macro-node i a vector $\mathbf{x}_i = (\mathbf{x}_i^1, ..., \mathbf{x}_i^m)$, where $\mathbf{x}_i^k = [x_{ij}^k]$ and $i.j \in I_i$. Each entry x_{ij}^k of $\mathbf{x}_i^k$ denotes the probability that the concrete service $i.j$ will be invoked by the class-k request when the workflow reaches the stage indicated by the macro-node i. It holds the constraint that $\sum_{j \in I_i} x_{ij}^k = 1$, for each class k and $i \in \mathcal{V}$. With this model, we assume that, in general, the broker can probabilistically route to different concrete services the requests (belonging to a same QoS class k) for an abstract service i. The deterministic selection of a single concrete service corresponds to the case $x_{ij}^k = 1$ for a given $i.j \in I_i$.

4. Optimal Service Selection

4.1. General Optimization Problem

We denote by K the set of QoS classes, with $m = |K|$. Class-k requests ($k \in K$) for the composite service arrive at the broker at rate γ^k, being $\boldsymbol{\gamma} = (\gamma^1, ..., \gamma^m)$ the overall arrival rate from outside to the broker. For each class-k request, the Selection Manager in the broker has to assign a concrete service $i.j$ for each abstract service i in the DSG, under given global QoS constraints. We recall that, in general, the assignment can also be probabilistic. Therefore, given the DSG our objective is to determine the optimal values for the vector $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ with $n = |\mathcal{V}|$, in such a way that the Selection Manager can use these values to select the concrete services fulfilling its QoS goals.

Let λ_i^k be the rate of class-k requests that arrive at the abstract service $i \in \mathcal{V}$. Using well-known flow conservation arguments, we get the following set of linear equations for the request rates, that can be used to calculate λ_i^k:

$$\boldsymbol{\lambda}^k = \mathbf{P}^{k\mathrm{T}} \boldsymbol{\lambda}^k + \gamma^k \, \mathbf{e}_1 \quad \text{for each } k \in K \tag{1}$$

where $\boldsymbol{\lambda}^k = (\lambda_1^k, ..., \lambda_n^k)$ and $\mathbf{e}_1 = (1, 0, ..., 0)$ are column vectors and $\mathbf{P}^k = [p_{rs}^k]$ is the $n \times n$ routing probability matrix for class-k requests. We assume that a request may not change dynamically its class during the fulfillment of the composite service. Given a flow of requests λ_i^k for the abstract service i, the broker splits it among the corresponding concrete services $i.j \in I_i$ according to the $\mathbf{x}_i^k = [x_{ij}^k]$ probabilities. Hence, $x_{ij}^k \lambda_i^k$ is the flow of requests for the concrete service $i.j$ generated by clients in the QoS class k.

Each concrete service is characterized by a set of QoS attributes, such as the execution time, the cost, and the reputation [10, 15]. We assume that the broker knows the values of these attributes for all the concrete services, which are advertised by the service providers.

The global QoS that is experienced by the broker clients depends on both the total request flow $x_{ij}^k \lambda_i^k$ addressed to each concrete service, and by the value of the service QoS attributes. The broker can affect this QoS by appropriately setting the x_{ij}^k values, which are under its control. Given a vector $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, we denote by $F(\mathbf{x})$ the corresponding global QoS. Hence, the broker goal is to determine the value of $\mathbf{x}$ that minimizes (or maximizes) $F(\mathbf{x})$. Toward this end, it must also take into account some constraints that limit the set of feasible values for $\mathbf{x}$. This can be expressed by the following optimization problem:

$$\begin{aligned} &\text{Minimize } F(\mathbf{x}) \\ &\text{subject to } Q^\alpha(\mathbf{x}) \leq Q^\alpha_{max} \\ &\qquad\qquad Q^\beta(\mathbf{x}) \geq Q^\beta_{min} \\ &\qquad\qquad \mathbf{x} \in A \end{aligned} \tag{2}$$

where we denote by $Q^\alpha(\mathbf{x})$ and $Q^\beta(\mathbf{x})$ the QoS constraints, and by $\mathbf{x} \in A$ a set of functional constraints (e.g., this latter set includes the constraint $\sum_{j \in I_i} x_{ij}^k = 1$).

4.2. An Instance of the General Optimization Problem

In this section, we build an instance of the general optimization problem introduced in the previous section, to give a concrete example of the kind of problem that must be solved by the broker. For this purpose, we assume that the broker knows the value of the following QoS attributes for each concrete service $i.j$:

- the service rates $\boldsymbol{\mu}_{ij} = (\mu_{ij}^1, ..., \mu_{ij}^m)$, where $(\mu_{ij}^k)^{-1}$ is the mean execution time of the concrete service $i.j$ for the class k (without considering possible queueing delays);
- the cost $\mathbf{c}_{ij} = (c_{ij}^1, ..., c_{ij}^m)$, where c_{ij}^k is the price that a class-k request has to pay for each invocation of the concrete service $i.j$;
- the reputation $\mathbf{r}_{ij} = (r_{ij}^1, ..., r_{ij}^m)$, which is a measure of the trustworthiness of the concrete service $i.j$. and is a general opinion i.e., it aggregates the ratings of the service by other principals.

Other QoS attributes can be considered for the service selection. We focus on the attributes listed above without

any loss of generality of the proposed approach for service selection. The broker therefore characterizes the composite service it manages by the following QoS attributes for each class $k \in K$:

- the execution time E^k, which is the time needed to fulfil a class-k request for the composite service;
- the execution cost C^k, which is the price to be paid to fulfil a class-k request;
- the reputation R^k, which measures the trustworthiness of the composite service for a class-k request.

In this paper, as aggregated class-k functions for E^k, C^k, and R^k we choose respectively the average of the execution time, execution cost, and reputation. Specifically, we consider the following functions expressed in terms of the variables x_{ij}^k, the request arrival rates λ_i^k, and the QoS attributes of the concrete services:

$$E^k(\mathbf{x}) = \sum_{i \in \mathcal{V}} \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in I_i} \frac{x_{ij}^k / \mu_{ij}^k}{1 - \sum_{c \in K} x_{ij}^c \lambda_i^{c*} / \mu_{ij}^c} \tag{3}$$

$$C^k(\mathbf{x}) = \sum_{i \in \mathcal{V}} \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in I_i} x_{ij}^k c_{ij}^k \tag{4}$$

$$R^k(\mathbf{x}) = \frac{1}{\sum_{i \in \mathcal{V}} \frac{\lambda_i^{k*}}{\gamma^k}} \sum_{i \in \mathcal{V}} \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in I_i} x_{ij}^k r_{ij}^k = \tag{5}$$

$$= \frac{1}{\sum_{i \in \mathcal{V}} \lambda_i^{k*}} \sum_{i \in \mathcal{V}} \lambda_i^{k*} \sum_{j \in I_i} x_{ij}^k r_{ij}^k$$

where $\boldsymbol{\lambda}^{k*}$ is the solution of (1) and λ_i^{k*}/γ^k is the mean number of class-k invocations to the i-th abstract service.

The expressions for C^k and R^k are self-explanatory. The expression for E^k i.e., the mean execution time for class-k requests, has been obtained under the hypothesis of the BCMP theorem [4]. If the queueing discipline is FCFS (the clients are served by the concrete services in the order in which they arrive), it is necessary to impose the restriction that all the classes have the same mean service rate for each invocation (i.e., the service rates are class-independent) that is, $\mu_{ij}^1 = ... = \mu_{ij}^m$ and that the service times are exponentially distributed[2].

As already pointed out, we focus on execution time, cost, and reputation; however, our problem formulation can be generalized and easily extended to take into account other QoS attributes, such as availability and reputation, applying, if necessary, a linearization method as in [6, 15].

[2]If the conditions of the BCMP theorem do not hold, (3) can be still used as a measure of the congestion at the concrete service $i.j$, that can be used to avoid highly congested nodes [3].

We assume that the broker wants, in general, to optimize multiple QoS attributes (which can be either mutually independent or possibly conflicting); therefore, the optimal service selection results in a multi-objective optimization. We tackle the multi-objective problem by transforming it into a single objective problem through the weighted sum approach, which is the most widely used scalarization method. Specifically, we consider as objective function a weighted sum of the mean execution time, the mean reputation, and the mean cost of the composite service averaged over all the classes, subject to the QoS constraints on the execution time, the cost, and the reputation of the composite service for each class. Therefore, the objective function to be minimized is expressed as follows:

$$F(\mathbf{x}) = w_e \frac{E(\mathbf{x}) - E_{min}}{E_{max} - E_{min}} + w_c \frac{C(\mathbf{x}) - C_{min}}{C_{max} - C_{min}} + \\ + w_r \frac{R_{max} - R(\mathbf{x})}{R_{max} - R_{min}} \tag{6}$$

where w_e, w_c, w_r are the weighting coefficients representing respectively the relative importance of execution time, cost, and reputation in the service composition; it holds $w_e, w_c, w_r \geq 0$ and $w_e + w_c + w_r = 1$. Note that the expression for the reputation has been converted to a minimization form. E_{max} (E_{min}), C_{max} (C_{min}) and R_{max} (R_{min}) denote respectively the maximum (minimum) value for the execution time, the cost, and the reputation. We explain how to determine them after having introduced the constraints of the optimization problem.

The overall mean execution time, cost, and reputation of the composite service are given by, respectively:

$$E(\mathbf{x}) = \frac{1}{\Gamma} \sum_{k \in K} \gamma^k E^k(\mathbf{x})$$

$$C(\mathbf{x}) = \frac{1}{\Gamma} \sum_{k \in K} \gamma^k C^k(\mathbf{x})$$

$$R(\mathbf{x}) = \frac{1}{\Gamma} \sum_{k \in K} \gamma^k R^k(\mathbf{x})$$

where $\Gamma = \gamma^1 + ... + \gamma^m$ and $E^k(\mathbf{x})$, $C^k(\mathbf{x})$, and $R^k(\mathbf{x})$ are given by (3), (4), and (5). The decision variables of the optimization problem are the variables x_{ij}^k ($k \in K$, $i \in \mathcal{V}$, $j \in I_i$), that are subject to the following constraints:

$$x_{ij}^k \geq 0 \qquad \text{for } i \in \mathcal{V},\ j \in I_i,\ k \in K \tag{7}$$

$$\sum_{j \in I_i} x_{ij}^k = 1 \qquad \text{for } i \in \mathcal{V},\ k \in K \tag{8}$$

$$\sum_{k \in K} \frac{x_{ij}^k \lambda_i^k}{\mu_{ij}^k} < 1 \qquad \text{for } i \in \mathcal{V},\ j \in I_i \tag{9}$$

$$E^k(\mathbf{x}) \leq e^k_{max} \qquad \text{for } k \in K \qquad (10)$$
$$C^k(\mathbf{x}) \leq c^k_{max} \qquad \text{for } k \in K \qquad (11)$$
$$R^k(\mathbf{x}) \geq r^k_{min} \qquad \text{for } k \in K \qquad (12)$$

Equations (7)-(9) are functional constraints, corresponding to those generically indicated as $\mathbf{x} \in A$ in (2); specifically, the latter is the condition that guarantees the system stability. Equations (10)-(12) are the QoS constraints on execution time, cost, and reputation, where e^k_{max}, c^k_{max} and r^k_{min} are respectively the maximum execution time, the maximum cost, and the minimum reputation accepted by the class-k clients. These constraints are instances of the generic QoS constraints indicated in (2).

The maximum and minimum values of the QoS attributes in the objective function (6) are determined as follows. E_{max}, C_{max}, and R_{min} are simply expressed respectively in terms of e^k_{max}, c^k_{max}, and r^k_{min}. For example, the maximum execution time E_{max} is given by:

$$E_{max} = \frac{1}{\Gamma} \sum_{k \in K} \gamma^k e^k_{max}$$

Similar expressions hold for C_{max} and R_{min}. The values for E_{min}, C_{min}, and R_{max} are determined by solving a constrained optimization problem in which the objective function is the QoS attribute of interest for the corresponding minimum (maximum) value, subject to the functional constraints (7)-(9). For example, the minimum execution time E_{min} is given by the solution of the following constrained optimization problem:

$$\text{Minimize } E(\mathbf{x})$$

$$\text{subject to } x^k_{ij} \geq 0 \qquad \text{for } i \in \mathcal{V},\ j \in I_i,\ k \in K$$
$$\sum_{j \in I_i} x^k_{ij} = 1 \qquad \text{for } i \in \mathcal{V},\ k \in K$$
$$\sum_{k \in K} \frac{x^k_{ij} \lambda^k_i}{\mu^k_{ij}} < 1 \qquad \text{for } i \in \mathcal{V},\ j \in I_i$$

Similar optimization problems have to be solved to obtain the values for C_{min} and R_{max}. In the latter case, the objective function $R(\mathbf{x})$ has to be maximized.

The constrained optimization problem we formulate is easily solved with standard optimization techniques of nonlinear programming [2], as both the objective function and the constraint sets are convex. We omit the demonstration for space reasons; anyway, the only function whose convexity has to be proved is the one for the execution time, being the functions for the cost and the reputation linear.

4.3 Service Selection Process

When the Selection Manager triggers the solution of the optimization problem, the Optimization Engine executes the following steps. First, it solves the system of linear equations (1); then, it solves the optimization problem (6), constrained by (7)-(12).

To carry out the service selection, the Selection Manager uses the solution of the optimization problem as follows. First, it negotiates with the requestor his service class: let k be the service class. Then, it considers only the elements of the solution vector $\mathbf{x}$ that pertains to class k. If for a given abstract service i there is more than one $x^k_{ij} \neq 0$, the Selection Manager selects randomly, using the probability values x^k_{ij}, one concrete service to which it binds the requestor.

The service selection provided by the Optimization Engine is valid until some event that requires a new solution of the optimization problem occurs. Indeed, if the DSG changes, the optimal service selection has to be recomputed. This happens in the following cases. First, some routing probability p^k_{rs} changes; these probabilities are periodically recomputed by the Execution Path Analyzer. Second, the service composition changes, because either an abstract service or a concrete service is added or removed. Finally, the QoS Monitor identifies some significant change in the QoS level provided by the concrete services.

5. Numerical Results

In this section, we use an example-driven approach to show how the service selection behaves for different objective functions. We also discuss the computational cost of the proposed optimization problem.

We consider a simple, but general enough, DSG composed of 4 abstract services and 7 concrete services. The DSG resembles that of Figure 2, where $n_1 = n_2 = n_3 = 2$ and $n_4 = 1$. Requests for the composite service are classified as *silver* and *gold*, denoted by the superscripts 1 and 2. The routing probability matrixes for the two QoS classes are $\mathbf{P}^1 = \begin{smallmatrix} 0 & 0.7 & 0.3 & 0 \\ 0.2 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{smallmatrix}$ and $\mathbf{P}^2 = \begin{smallmatrix} 0 & 0.5 & 0.5 & 0 \\ 0.1 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{smallmatrix}$. Users in the gold class accept to pay a higher cost (c^2_{max}) to get better execution time (e^2_{max}) and reputation (r^2_{min}), while users in the silver class accept possible worse execution time (e^1_{max}) and reputation (r^1_{min}) to gain a lower cost (c^1_{max}). The concrete services differ in terms of service rate, cost, and reputation. Table 1 summarizes the system parameters.

In the first set of experiments, we analyze the behavior of the decisional process operated by the service broker. We consider 4 different objective functions (see Figure 3) that the broker could want to optimize: in *case 1* it wants to minimize the mean execution time ($w_e = 1$); in *case 2* it wants to minimize the mean cost ($w_c = 1$); in *case 3* and *case*

Table 1. Parameters of the example.

Parameters	Values
(γ^1, γ^2)	(0.1, 0.01)
$(\mu^1_{11}, \mu^1_{12}, \mu^1_{21}, \mu^1_{22}, \mu^1_{31}, \mu^1_{32}, \mu^1_{41}, \mu^2_{11}, \mu^2_{12}, \mu^2_{21}, \mu^2_{22}, \mu^2_{31}, \mu^2_{32}, \mu^2_{41})$	(0.2, 0.8, 0.5, 0.5, 0.4, 0.6, 1, 0.4, 1.6, 1, 1, 0.8, 1.2, 2)
$(c^1_{11}, c^1_{12}, c^1_{21}, c^1_{22}, c^1_{31}, c^1_{32}, c^1_{41}, c^2_{11}, c^2_{12}, c^2_{21}, c^2_{22}, c^2_{31}, c^2_{32}, c^2_{41})$	(0.3, 0.8, 0.4, 0.6, 0.4, 0.5, 1, 0.6, 1.6, 0.8, 1.2, 0.9, 1, 2)
$(r^1_{11}, r^1_{12}, r^1_{21}, r^1_{22}, r^1_{31}, r^1_{32}, r^1_{41}, r^2_{11}, r^2_{12}, r^2_{21}, r^2_{22}, r^2_{31}, r^2_{32}, r^2_{41})$	(3, 3, 4, 1, 2, 2, 1, 6, 6, 8, 2, 4, 4, 2)
(e^1_{max}, e^2_{max})	(10, 6)
(c^1_{max}, c^2_{max})	(2.5, 5)
(r^1_{min}, r^2_{min})	(2, 3)

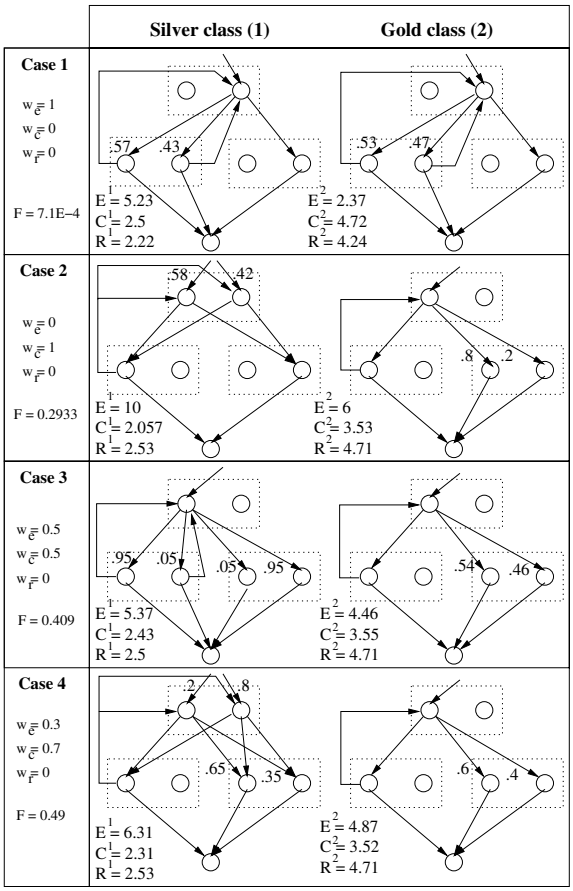

Figure 3. Solution of the optimization problem in the four cases.

4 it wants to minimize both the mean execution time and the mean cost with the same ($w_e = w_c = 0.5$) and different ($w_e = 0.3$, $w_c = 0.7$) weights, respectively. Figure 3 shows the solution of the optimization problem in the four cases; the values within the graphs are those of the variables x^k_{ij} when different from 1.

In case 1, the broker always selects the fastest concrete service 1.2. The services 2.1 and 2.2 have the same service rate but different costs: the request flow is partitioned between the two, assigning more to the cheapest one.

In case 2, silver and gold requests are managed differently. For gold requests, the cheapest concrete services 1.1 and 2.1 are selected, while the request flow is partitioned between the two concrete services that implement 3. Indeed, 80% of gold requests are assigned to 3.1, which has the lower cost, but the remaining 20% is assigned to 3.2, that provides an higher performance. This allows to satisfy the constraint on the execution time for the gold class. For silver requests, the cheapest concrete services 2.1 and 3.2 are deterministically selected. The requests for service 1 are instead almost equally assigned to 1.1 (58%) and 1.2 (42%). This selection allows to balance the load between the concrete services (considering that all the gold requests are assigned to 1.1) as well as to satisfy the time constraint.

In case 3, the broker aims at minimizing both the execution time and the cost. Silver requests invoke the concrete service with the lower cost for services 1 and 2 (2.1 is assigned 95% of time). This choice allows to maintain the execution time close to the optimal value (see case 1) and produces a small increase with respect to the optimal cost (see case 2). For gold requestors, the choice operated by the broker allows to maintain the global cost close to the optimal value (see case 2). However, even if the constraints on the execution time are largely satisfied, the corresponding global value is far from the minimum one (see case 1).

In case 4, the service selection resembles that of case 2 for both the two classes, being the only difference in the request partitioning. For the gold class, more requests are assigned to the concrete service 3.2 to reduce the execution time. For the same reason, more silver requests are assigned to 1.2, and 3.2 is also used. This choice allows to drastically reduce the execution time, with a small increase in the cost. This case confirms that the optimal selection behaves as expected: an objective function where the cost weights more than the execution time allows to achieve a good performance paying a cost close to the minimum one.

In the second set of experiments, we estimate the computation cost of solving the optimization problem by evaluating the number of iterations needed to find the solution. The experiments have been carried out by increasing the number of concrete services available for the selection, under the hypothesis that the concrete services are uniformly distributed among the abstract services. We solve the optimization problem using two tools: the Matlab function `fmincon` and SNOPT, which is a Fortran routine integrated into the Tomlab package for Matlab. Both tools are based on the Sequential Quadratic Programming methods [2], but use different merit functions in the line search phase.

The analysis has been conducted considering both lin-

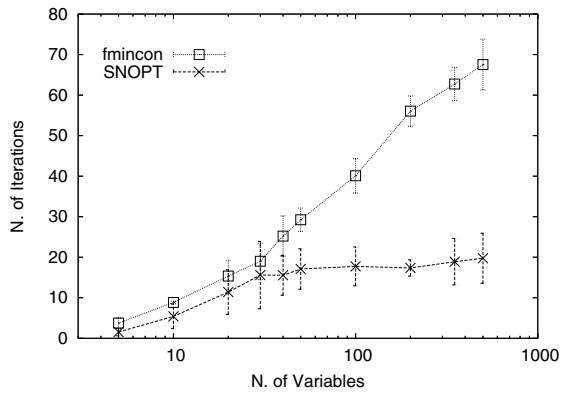

Figure 4. Number of iterations vs number of variables in the optimization problem.

ear ($w_e = 0$) and non-linear ($w_e \neq 0$) objective functions, with only linear constraints and with both linear and non-linear constraints. For space reasons we report only the worst case, that corresponds to have a non-linear objective function and both linear and non-linear constraints. Figure 4 shows that the number of iterations needed to solve the problem using SNOPT is one third of the iterations required by `fmincon`. For 500 variables, the average number of iterations is 19.75 with SNOPT and 67.55 with `fmincon`. The execution time to solve the problem obviously depends on the characteristics of the machine used; in our case (AMD Athlon 2.2Ghz, 1Gb RAM) and a 500 variables problem, we measured on average 33.8 seconds with SNOPT and 511.86 seconds with `fmincon`.

6. Conclusions

We have addressed the problem of selecting service implementations in a composite service managed by a broker which supports multiple QoS classes. We have defined a graph that models the candidate concrete services among which the broker can statistically divide the flow of requests for the composite service. Using this graph, we have formulated a constrained multi-criteria optimization problem which embeds various QoS attributes that may be mutually independent or possibly conflicting. Our problem formulation is extensible and can be modified to take into account other QoS attributes.

We have analyzed the solution of the optimization problem in a simple and intuitive service scenario and evaluated its computational cost. Our results show that, using an optimized routine, we have obtained a low computational cost. This makes suitable our approach to manage service selection in a real operating broker-based architecture, where the broker efficiency in replying to the requestors cannot be neglected. This paper is part of an ongoing research; future work concerns the inclusions of other execution patterns in our workflow model, the realization of the broker-based architecture, and the analysis and modeling of a multi-broker scenario, where the brokers cooperate or compete in the use of the same concrete services.

References

[1] D. Ardagna and B. Pernici. Global and Local QoS Guarantee in Web Service Selection. In *Proc. of Business Process Management Workshops*, pages 32–46, 2005.

[2] D. Bertsekas. *Nonlinear Programming, 2nd Ed.* Athena Scientific, 1999.

[3] D. Bertsekas and R. Gallager. *Data Networks, 2nd Ed.* Prentice Hall, 1991.

[4] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains*. J. Wiley, 1998.

[5] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service Composition Based on Genetic Algorithms. In *Proc. of Genetic and Computation Conf.*, June 2005.

[6] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, and K. J. Kochut. Modeling Quality of Service for Workflows and Web Service Processes. *Web Semantics J.: Science, Services and Agents on the World Wide Web*, 1(3):281–308, 2004.

[7] F. Casati, M. Castellanos, U. Dayal, and M.-C. Shan. Probabilistic, Context-sensitive, and Goal-oriented Service Selection. In *Proc. of 2nd Int'l Conf. on Service Oriented Computing*, pages 316–321, 2004.

[8] D. B. Claro, P. Albers, and J.-K. Hao. Selecting Web Services for Optimal Composition. In *Proc. of 2nd Int'l Workshop on Semantic and Dynamic Web Processes*, July 2005.

[9] E. M. Maximilien and M. P. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, Sept./Oct. 2004.

[10] D. A. Menasce. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72–75, Nov./Dec. 2002.

[11] M. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui. A QoS Broker Based Architecture for Efficient Web Services Selection. In *Proc. of 2005 Int'l Conf. on Web Services*, pages 113–120, July 2005.

[12] L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-based Service Selection and Ranking with Trust and Reputation Management. In *Proc. of 2005 IEEE Int'l Conf. on Cooperative Information Systems*, Nov. 2005.

[13] T. Yu and K. J. Lin. A Broker-Based Framework for QoS-Aware Web Service Composition. In *Proc. of 2005 IEEE Int'l Conf. on e-Technology, e-Commerce and e-Service*, Mar. 2005.

[14] T. Yu and K. J. Lin. Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. In *Proc. of 3rd Int'l Conf. on Service Oriented Computing*, pages 130–143, Dec. 2005.

[15] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, Aug. 2004.