A QoS-based Selection Approach of Autonomic Grid Services

Jonatha Anselmi, Danilo Ardagna and Paolo Cremonesi
Politecnico di Milano, DEI
Via Ponzio 34/5, I-20133 Milan, Italy
{anselmi, ardagna, cremones}@elet.polimi.it

ABSTRACT

The Web service composition (WSC) is the process of building an instance of an abstract workflow by combining appropriate Web services that satisfies given QoS requirements. In general, QoS requirements consists of a number of constraints. The selection process requires global optimization and can be formalized as a mixed integer linear programming problem which cannot be solved in polynomial time. However, since the number of submitted workflows is large and the QoS is highly dynamic, the fast selection of composite Web Services is particularly important.

In this paper, we present a QoS broker-based framework for Web services execution in autonomic grid environments. The main goal of the framework is to support the broker in selecting Web services based on the required QoS. To achieve this goal, we propose a novel approach: since successive composed Web services requests can have the same task to Web service assignment, we address the Multiple Instance WSC (MI-WSC) problem optimizing simultaneously the set of requests which will be submitted to the system in the successive time interval instead of independently computing a solution for each incoming request.

Experimental results show that the proposed algorithm has better performance with respect to existing techniques. Moreover, the qualities of the selected composite Web services are not significantly different from the optimal ones.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles; C.4 [**Performance of Systems**]: Modeling Techniques

Keywords

Web Service Selection, Heuristics, Quality of Service, Grid, Autonomic Systems.

General Terms

Management, Performance, Algorithms.

SOCP'07, June 26, 2007, Monterey, California, USA.

1. INTRODUCTION

In Service Oriented Architectures (SOA) complex business applications can be described as composed business processes constitued by a set of individual *abstract* Web services. At run-time, SOA mechanisms select the best set of *concrete* Web services which support the given abstract descriptions in order to guarantee the fulfillment of Quality of Service (QoS) constraints. The Web service composition (WSC) problem is a combinatorical optimization problem which ensures the optimal mapping between each abstract Web service and the available concrete Web services which implement the abstract description [20, 19]. The mapping algorithm optimizes quality metrics perceived by end-users taking into account limited Web services capacity and QoS constraints defined by means of *Service Level Agreement* [7, 11, 3]. In general QoS requirements are difficult to satisfy since Internet application workloads can vary by orders of magnitude even within the same business day [10]. Such variations require self-managing techniques [15] which dynamically allocate resources among different services on the basis of workload predictions. Autonomic grid architectures provide basic mechanisms which dynamically re-configure service center infrastructures and can be exploited for the fulfillment of QoS requirements.

Many approaches have been proposed for the solution of the WSC problem (see, e.g., [3, 7, 11, 19]) and they perform the optimization considering a single business process invocation. This means that the WSC problem is independently solved for each incoming request. This approach can introduce a high overhead in the computation because it does not take into account that the optimal solution of the n-th request can be often significantly close to the optimal solution of the $(n+1)$-th request.

In this paper we extend the Single-Instance WSC (SI-WSC) problem we proposed in [3] and address the Multiple-Instance WSC (MI-WSC) problem optimizing the mapping between abstract and concrete Web services for a *set* of requests simultaneously and allowing the prior reservations of resources.

The framework proposed in this paper supports the execution of Web services applications in autonomic grid environments. Requests of composed Web service execution are submitted to grid brokers specifying preferences and the set of local and global constraints. A broker solves the MI-WSC problem ensuring the fulfillment of global and local constraints and taking into account limited resources, variable QoS profiles of Web services and the long term process execution. Our optimization is performed periodically within a given time interval and is based on short-term predictions on the number of incoming *instances*, i.e. requests, which will be submitted by users during the successive time interval.

We formulate the MI-WSC problem as a Mixed Integer Linear Programming (MILP) problem. Since the optimal solution requires

a strong computational effort, we propose a greedy heuristic which quickly computes a suboptimal solution. A comparison with a bound on the global optimum of the problem reveals the heuristic provides good results. Moreover, we experimentally show that our new approach significantly reduces computation overheads with respect to existing techniques.

The paper is organized as follows. Section 2 provides preliminary notation and definitions used in the paper. In Section 3 we illustrate the reference grid framework. In Section 4 we propose a MILP formulation for the MI-WSC problem and a heuristic for its solution. Experimental results are further presented in Section 5. Finally, Section 6 reviews other approaches proposed in the literature and Section 7 draws conclusions and outlines future research.

2. PRELIMINARY NOTATION AND DEFINITIONS

In the remainder of the paper we refer to abstract Web services as *tasks* and to composed service execution requests as *instances*. We denote by I the number of tasks which belong to an instance, by R the number of quality dimensions, and by N the predicted number of instances which will be submitted to grid brokers during the successive control interval. Tasks and concrete Web services will be indexed respectively by i and j, Web services operations will be indexed by o, quality dimensions by r and instances by n. We denote by WS_i the set of indices of Web services able to execute task i and by OP_j the set of indices of operations implemented by Web service j. We consider $R = 4$ quality dimensions: *execution time*, *availability*, *price*, and *reputation* which have been the basis of other literature approaches [16, 17, 20, 7, 11].

The composed business process specification is written in BPEL. As discussed in [3], we introduce some annotations in order to identify:

- global and local constraints on quality dimensions;
- the maximum number of iterations for cycles;
- the expected frequency of execution of conditional branches;
- QoS user preferences $\{\omega_1, \ldots, \omega_R\}$ such that $\omega_r \geq 0$ and $\sum_r \omega_r = 1$;
- Web service dependency constraints.

A *global constraint* predicates on quality attributes at process level or on a subset of tasks of the specification while a *local constraint* specifies the quality requirement for a particular task. For example, a global constraint can impose that the execution time for the whole business process is less than or equal to a given threshold, while a local constraint can impose a limitation on the price of a single task. Cycles of BPEL specifications are unfolded according to the maximum number of iterations which can be evaluated from past executions by inspecting system logs or can be specified by the composite service designer as well as the frequency of execution of conditional branches. If an upper bound for cycles execution cannot be determined, then the optimization could not guarantee that global constraints are satisfied [20]. Web service dependency constraints represent the case of stateful Web services in which two tasks must be executed by the same Web service. Constraints and BPEL annotations are specified by WS-Policy [2].

For simplicity, in the following definitions we assume that a composite service is characterized by a single initial task and a single end task.

We refer to an *execution path* as a set of tasks indices $\{1, \ldots, i, \ldots, I\}$ such that index 1 represents the initial task, I the final task and no i_a, i_b belong to alternative branches. As shown in Figure 1, execution paths can include parallel sequences and will be indexed by k. We denote by $\mathcal{A}_k$ the set of indices of tasks included in execution path k and by K the number of different execution paths arising from the composed service specification. The evaluation of the r-th quality dimension along execution path k of instance n will be denoted by $q^k_{r,n}$. Note that the set of execution paths related to an instance identifies all the possible execution scenarios of the composite service. The optimization problem considers every possible execution paths according to their probability of execution p_k which can be evaluated by the product of the frequency of execution of branch conditions included in execution paths and annotated in the BPEL specification.

We refer to *global plan* as a set of ordered triples $\{< (i,n), (j,o), x_{i,n} >\}$ which associates each task executed by instance n, i.e. (i,n), to a Web service operation invocation (j,o) at time instant $x_{i,n}$ and satisfies constraints for every execution paths.

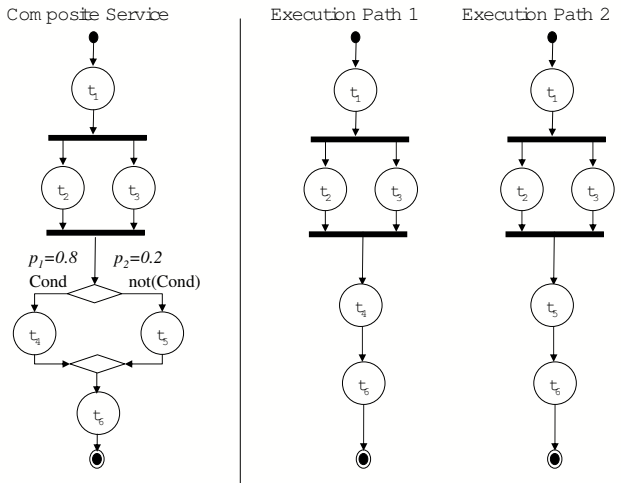

Figure 1: Execution Paths

3. THE REFERENCE GRID ENVIRONMENT

In our framework, the grid infrastructure is composed by a set of Virtual Organizations (VO) which share computing resources for the attainment of a given goal [13]. VOs and end-users establish SLA contracts for the service provisioning. VOs' resources are represented by concrete Web services which are physically deployed and executed by *Local Grids*. A VO supports a limited number of Web service operation invocations and can execute concrete Web service operations that are located in different VO sites.

Grid middleware provides basic mechanisms to manage the overall infrastructure of a service center, adapting the physical configuration to the requirements of varying incoming workloads, implementing service differentiation and performance isolation of multiple Web services.

As depicted in Figure 2, our framework exploits local grids monitoring infrastructure and includes a *Service Registry*, and a *Service Broker*.

Low level information provided by the *Grid Monitoring Infrastructure* is used to identify requests of different Web services operations and to estimate requests service times (i.e., the CPU and disk time required by the physical infrastructure to execute each operation).

The *Service Registry* stores the variable QoS profiles and the number of available invocations. As discussed in [3], QoS profiles

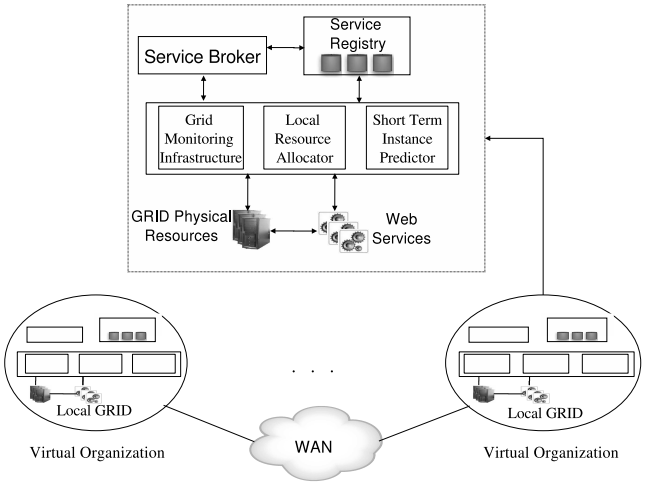

Figure 2: The reference grid environment

follow a discrete stepwise function that is periodic with period T (see Figure 3) and are obtained taking into account seasonal workload variations. The discrete time interval will be denoted by u,

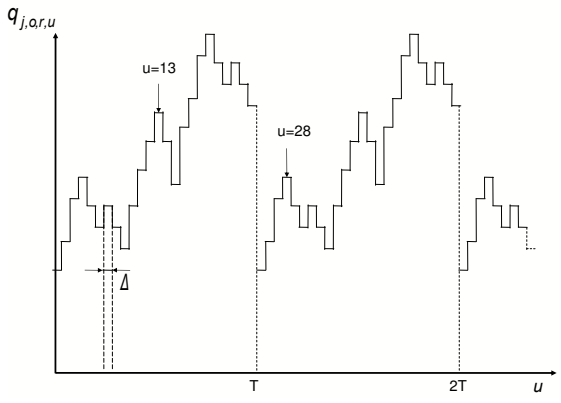

Figure 3: Example of a periodic QoS profile

the discretization interval size by Δ. We assume that QoS profiles are constant in each interval u. The quality value for dimension r, operation o of Web service j in time interval u will be denoted by $q_{j,o,u,r}$. In autonomic systems, Δ is about half an hour [1], and if we assume that the incoming workload has a daily seasonal component, T is 24 hours. Index u ranges in $\{1, \ldots, U\}$, where $U = \lceil E/\Delta \rceil$ and E denotes the execution time global constraint for the composed process. We denote by $\mathcal{N}_{j,o,u}$ the number of available operation o invocations for concrete Web service j which can be executed in the time interval u.

In order to simultaneously schedule multiple requests, the *Short Term Instance Predictor* forecasts the number of instances N which will be submitted to the *Service Broker* in the next control interval u. As in [5], the Short Term Instance Predictor has been implemented combining smoothing-exponential techniques, auto-regressive-moving-average and polynomial regression models. The *Local Resource Allocator* reserves local grid physical resources to different Web service operation invocations in order to meet QoS requirements [3]. Since the high variability of the Internet workloads, the allocator employs autonomic techniques for the dynamic allocation of grid physical resources among different Web service invocations. Application requirements are met adapting the physical infrastructure by exploiting grid middleware primitives (e.g. Globus toolkit GRAM or EGEE WMS, [18]).

Finally, the *Broker* receives composed Web service execution requests from VO members and external users, consults the local and remote Service Registries in order to obtain QoS profiles $q_{j,o,u,r}$ and available Web service operation invocations $\mathcal{N}_{j,o,u}$, receives the prediction on the number of instances N for the next control interval, and determines the global plan. Local grid resources are subsequently reserved on the basis of the global plan identified by the broker and $\mathcal{N}_{j,o,u}$s are updated accordingly.

Note that, within the proposed framework, the execution plan of an arriving request is pre-determined since the selection is performed in the previous time interval. At the end of every control time interval, the grid reserves resources according to the global plan.

4. FORMULATION AND ALGORITHMS FOR THE MI-WSC PROBLEM

In our framework, the broker of each VO, see Figure 2, solves a MI-WSC problem for each time interval u finding the optimal mapping between tasks and Web service operations for N instances. Clearly, the broker solves the problem within interval u.

Unless otherwise specified, we assume that indices j and o respectively range in sets WS_i and OP_j, where reference indices i and j will be clear from the context. Recall that if not otherwise stated indices i, u, r, k and n respectively range in $\{1, \ldots, I\}$, $\{1, \ldots, U\}$, $\{1, \ldots, R\}$, $\{1, \ldots, K\}$ and $\{1, \ldots, N\}$. Execution time will be indexed by $r = 1$.

In Section 4.1 we give a MILP formulation of the MI-WSC problem. In Section 4.2 we show that the MILP optimal solution requires a strong computational effort and makes unfeasible the analysis of real world workloads. Therefore, in Section 4.3 we propose a greedy heuristic which quickly computes a suboptimal solution.

4.1 MILP Formulation

In the present paper we solve the MI-WSC for a fixed business process specification, i.e. we assume that users require the execution of the same application and imposes the same QoS constraints. The goal of the MI-WSC problem is to maximize the average aggregated value of QoS for all instances. The average is obtained by considering all the possible execution scenarios, i.e. all the execution paths k arising from the composed service specification, and their probability of execution p_k. As we discussed in [4], the aggregated value of QoS is obtained by applying the Simple Additive Weighting (SAW) technique, one of the most widely used techniques to obtain a scalar score from a list of dimensions. SAW technique is applied to normalized aggregated values of quality.

The decision variables of our MILP model are

- $y_{i,j,o,u,n} \in \{0, 1\}$ is equal to 1 if task i is executed by Web service $j \in WS_i$ with operation $o \in OP_j$ in time interval u for the instance n, otherwise 0;
- $w_{i,u,n} \in \{0, 1\}$ is equal to 1 if task i is executed in time interval u for the instance n, otherwise 0;
- $x_{i,n} \in \mathbb{R}^+$ represents the time instant in which instance n executes task i.

The objective function to maximize is:

$$\sum_n \sum_k p_k \sum_r w_r v^k_{r,n} \quad (1)$$

which is subject to

$$\sum_j \sum_o \sum_u y_{i,j,o,u,n} = 1 \qquad \forall i,n \quad (2)$$

$$w_{i,u,n}(u-1)\Delta \le x_{i,n} \le w_{i,u,n} u\Delta + E(1-w_{i,u,n}) \qquad \forall i,u,n \quad (3)$$

$$w_{i,u,n} = \sum_j \sum_o y_{i,j,o,u,n} \qquad \forall i,u,n \quad (4)$$

$$\sum_n \sum_i y_{i,j,o,u,n} \le \mathcal{N}_{j,o,u} \qquad \forall j,o,u \quad (5)$$

$$q_{i,r,n} = \sum_j \sum_o \sum_u q_{j,o,u \bmod (T/\Delta),r} y_{i,j,o,u,n} \qquad \forall i,r,n \quad (6)$$

$$x_{i_b,n} - (q_{i_a,1,n} + x_{i_a,n}) \ge 0 \qquad \forall t_{i_a} \to t_{i_b}, \forall n \quad (7)$$

$$x_{I,n} + q_{I,1,n} \le E \qquad \forall n \quad (8)$$

$$q^k_{r,n} = \sum_{i \in \mathcal{A}_k} \gamma^k_{i,r} q_{i,r,n} \qquad \forall r,n,k \quad (9)$$

$$q^k_{r,n} \; [\ge \mid \le] \; Q^r \qquad \forall r \ne 1, \forall n,k \quad (10)$$

Terms $v^k_{r,n}$ in (1) are obtained through a normalization process with a computational complexity which is linear in the number of task [20]. A quality dimension is *positive* (*negative*) if the higher the value, the higher (the lower) the quality. For instance, the *availability* is a positive, while the *price* is a negative quality dimension. Thus, if r is a positive quality dimension we normalize the aggregated value $q^k_{r,n}$ as:

$$v^k_{r,n} = \begin{cases} 1 & \text{if } \max q^k_{r,n} = \min q^k_{r,n} \\ \dfrac{q^k_{r,n} - \min q^k_{r,n}}{\max q^k_{r,n} - \min q^k_{r,n}}, & \text{otherwise} \end{cases}$$

or vice versa:

$$v^k_{r,n} = \begin{cases} 1 & \text{if } \max q^k_{r,n} = \min q^k_{r,n} \\ \dfrac{\max q^k_{r,n} - q^k_{r,n}}{\max q^k_{r,n} - \min q^k_{r,n}} & \text{otherwise} \end{cases}$$

where $\min q^k_{r,n} = \min_{k,r,n} q^k_{r,n}$ and $\max q^k_{r,n} = \max_{k,r,n} q^k_{r,n}$. Constraints family (2) ensures the mapping between each task i and an operation of a concrete Web service. Constraints family (3) relates variables $x_{i,n}$ and $w_{i,u,n}$ and guarantees that if task i is executed in u by instance n, i.e. $w_{i,u,n} = 1$, then the execution of i starts in interval u, otherwise $x_{i,n}$ can assume an arbitrary value in $[0, E]$. Constraints family (4) relates variables $w_{i,u,n}$ and $y_{i,j,o,u,n}$: if for a generic instance n task i is executed by invoking in interval u the operation o of Web service j, i.e. $y_{i,j,o,u,n} = 1$, then $w_{i,u,n}$ is raised to 1. Constraints family (5) takes into account the grid finite resources limiting the number of invocations for a given Web service operation o executed in interval u. Constraints family (6) represents the quality of every task in term of the quality of the selected service. Constraints family (7) represents precedence constraints for subsequent tasks in the activity diagram. If task i_b is a direct successor of task i_a (indicated as $i_a \to i_b$), then the execution of task i_b starts after task i_a termination.

Finally, constraints family (8) guarantees that the execution time of each composed process is less than or equal to the execution time global constraint E while constraints family (10) represents the global constraints to be fulfilled for the remainder quality dimensions evaluated along the k-th execution path. The inequality is $\ge$ ($\le$) for positive (negative) quality parameters. Coefficients $\gamma^k_{i,r}$ depend on the composition rule of the quality parameter. For instance, the reputation is evaluated as the average of the reputation of the Web service operation invocations, i.e. $\gamma^k_{i,r} = 1/|\mathcal{A}_k|$, while the price is given by the sum of invocations prices, i.e. $\gamma^k_{i,r} = 1$ [20].

In order to handle *Web services dependency constraints* which represent the possibility that two task i_a and i_b must be executed by the same Web service, we introduce constraints

$$\sum_o \sum_u y_{i_a,j,o,u,n} = \sum_o \sum_u y_{i_b,j,o,u,n}, \; \forall j \in WS_{i_a} \cap WS_{i_b}, \forall n$$

$$\sum_o \sum_u y_{i_a,j,o,u,n} = 0, \; \forall j \in WS_{i_a} \setminus WS_{i_b}, \forall n$$

$$\sum_o \sum_u y_{i_b,j,o,u,n} = 0, \; \forall j \in WS_{i_b} \setminus WS_{i_a}, \forall n$$

In order to handle local constraints on quality dimension r for a single task i, i.e. Q^r_i, we also introduce constraint:

$$\sum_j \sum_o \sum_u q_{j,o,u,r} y_{i,j,o,u,n} \; [\ge \mid \le] \; Q^r_i$$

which predicates on each instance n. As constraints (10), the inequality is $\ge$ ($\le$) for positive (negative) quality parameters.

4.2 MI-WSC Complexity

In [4] we have shown that the problem of Web services selection for a single instance when the quality profiles are constant is equivalent to a Multiple-Choice Multi-Dimensional Knapsack Problem which is NP-hard. Thus also the MI-WSC problem is NP-hard.

For the SI-WSC problem [3], which can be easily obtained imposing $N = 1$ in the MI-WSC problem, the optimum solution is still hard to compute, however good results have been obtained limiting the computation to a minute: we have shown that the average percentage relative error between the approximate solution and the global optimum is 1.5%.

Brokers have to determine global plans within time interval u. The introduction of the free variable n significantly increases the computational effort required to obtain the optimum solution and the employment of commercial solvers does not suffice, since memory and solution time limit are reached even for very small problem size instances. Therefore, we propose a greedy heuristic which quickly computes a suboptimal solution which is discussed in the next section.

4.3 The Heuristic Solution

The heuristic we propose for the MI-WSC problem is a greedy algorithm based on the solution of the SI-WSC problem [3] which maximizes the average aggregated value of QoS for a single instance. The SI-WSC solution is obtained by running the MI-WSC problem with $N = 1$.

The basic idea relies on the fact that assuming an infinite number of resources, i.e. relaxing constraints (5) in the MI-WSC problem, every instance will have the same global plan which is equal to the global plan obtained optimizing a single instance. We exploit this idea in order to instantaneously optimize a number C of successive instances saturating critical resources. For the sake of simplicity, in the following we assume that the SI-WSC probem is always feasible, otherwise an adimission control scheme has to be introduced which rejects some incoming requests.

Let $T^n_{j,o,u}$ be the number of tasks of instance n executed by operation o of Web service j at time interval u.

The heuristic is shown in Algorithm 1. Our greedy approach exploits the fact that if the SI-WSC solution for instance n assigns task i on Web service j during time interval u then the solution of the $(n+1)$-th instance will have the same mapping if there exists a sufficient number of resources, i.e. if each $\mathcal{N}_{j,o,u}$ is still greater than or equal to $T^n_{j,o,u}$. Thus, we can instantly assign the maximum number of successive instances, C, which keeps each $\mathcal{N}_{j,o,u}$ greater than or equal to zero. Then, we iteratively re-run the SI-WSC problem for the $(n+C+1)$-th instance and compute again C until the task to Web service assignment is computed for all the instances.

Algorithm 1 Heuristic for the MI-WSC problem

1: $n \leftarrow 1$;
2: **while** $n \leq N$ **do**
3: Solve the SI-WSC problem for instance n;
4: $C \leftarrow \min_{j,o,u} \left\lfloor \frac{\mathcal{N}_{j,o,u}}{T^n_{j,o,u}} \right\rfloor$;
5: Adopt the task to Web services assignment for instances $n+1, \ldots, n+\min\{C, N-n\}$;
6: **for all** i, j, o, u **do**
7: **if** $y_{i,j,o,u}(n) == 1$ **then**
8: $\mathcal{N}_{j,o,u} \leftarrow \mathcal{N}_{j,o,u} - \min\{C, N-n\} T^n_{j,o,u}$;
9: **end if**
10: **end for**
11: $n \leftarrow n + C$;
12: **end while**

With Algorithm 1 we reduce brokers computation overheads because, whenever a request arrives to a broker, its optimal global plan has already been computed.

5. EXPERIMENTAL RESULTS

We evaluate the effectiveness of our approach measuring both the accuracy of Algorithm 1 and computation times savings with respect to existing approaches based on the classic solution of a single instance. Experimental analyses have been performed implementing a software prototype based on the ILOG *CPLEX* optimization library. The approach has been tested on about 300 randomly generated instances. The analyses have been performed on a 3 GHz Intel Pentium-D Workstation.

In Section 5.1 we provide parameters used for the evaluation. In Section 5.2 we evaluate the performance of Algorithm 1 computing the gap with respect to a bound of the global optimum of the problem and measuring computation times savings for real world workloads.

5.1 Parameters

The problem instances have been randomly generated as follows. N has been varied from 100 to 180. I has been varied between 10 and 90. We have varied the number of candidate Web services operations per task from 5 to 20 with step 5, while $N_{j,o,u}$ has been randomly generated between 20 and 40 with a uniform distribution. The execution time global constraint has been set to 24 hours and Δ has been set equal to one hour. Quality of service values of candidate Web services have been randomly generated according to the values reported in the literature (see [4]). Availability values were randomly generated assuming a uniform distribution in the interval 0.95 and 0.99999. Reputation was determined in the same way but considering the range $[0.8, 0.99]$. As in [9], we assume that the execution time has a gaussian distribution. We assumed that the price of each service invocation was proportional to service reputation and availability and inversely proportional to the execution time (i.e., the higher is the execution time of a Web service operation invocation, the lower is the price). Finally, the set of weights ω_r has been randomly generated and weights have been adjusted to sum 1.

5.2 Heuristic Solution Evaluation

As discussed in Section 4.2, the optimum of the MI-WSC problem cannot be computed for real world workloads and composed Web services. Therefore, we first evaluate the accuracy with respect to a bound. Since we formulated the MI-WSC problem as a *maximization* problem, the feasible solution computed by Algorithm 1 is less than or equal to the optimum, thus we measure the gap with respect to an *upper* bound on the optimum which can be easily obtained relaxing finite resource constraints (5), i.e. assuming $\mathcal{N}_{j,o,u} = +\infty$. The bound is computed running the SI-WSC problem for the first instance and adopting the same task to Web service mapping for the remaining $N-1$ instances.

Let X_M be the aggregated QoS value obtained by running Algorithm 1 and X_B be the QoS aggregated value computed by the upper bound. We measure the percentage gap as

$$\%\text{gap} = \frac{X_B - X_M}{X_M} \cdot 100\%. \tag{11}$$

The optimization execution time varies with the size of the problem. Since the broker has to solve the MI-WSC problem efficiently, for each instance the CPLEX execution time has been limited to one minute. CPLEX can find the optimum for a single instance within one minute for small problem size instances. Table 1 reports, as an example, the percentage gap between the approximate solution of Algorithm 1 and the problem upper bound for 20 randomly generated problem instances of different size. In this case the maximum gap is 18.3% while on average the gap is 7.8%. Note that in the instance with $I = 100$, $N = 158$ and $|WS_i| = 5$ the problem was unfeasible because the global constraints were too stringent.

I	$\lvert WS_i \rvert$	N	% gap
50	5	158	8.19
	10	170	10.75
	15	135	7.97
	20	172	9.73
60	5	161	2.66
	10	104	1.34
	15	116	1.37
	20	108	1.27
70	5	128	2.32
	10	140	2.28
	15	160	8.68
	20	131	7.18
100	5	158	unfeasible
	10	168	18.13
	15	123	16.33
	20	166	18.30

Table 1: Percentage gap between the approximate and global optimum solutions

Computation time savings have been computed by measuring the overhead of the approach we presented in [3] which performs the optimization on the basis of a single request. We denote by

$\mathcal{T}(N)$ the CPU time required by Algorithm 1 to optimize N instances and by $\mathcal{T}'(N) = N\mathcal{T}(1)$ the related CPU time required by the single instance based technique. We also denote by $\mathcal{S}(N) = \mathcal{T}'(N)/\mathcal{T}(N)$ the speed-up. Results are reported in Table 2, where we consider a business process with $I = 100$ and show that the overhead is significantly reduced up to a factor of 5. We retain that a loss of accuracy of about 8% could be acceptable considering the significant reduction of the computation time.

N	$\mathcal{T}$	$\mathcal{T}'$	$\mathcal{S}$
100	100	21	4.76
110	110	21	5.24
120	120	25	4.8
130	130	24	5.42
140	140	30	4.67
150	150	36	4.17
160	160	45	3.56
170	170	52	3.27
180	180	60	3

Table 2: CPU time (in minutes) and speed-up.

6. RELATED WORK

Recently, dynamic Web service composition have attracted great interest in the research community. The best set of services are selected by solving an optimization problem, and are invoked at run time by implementing a *dynamic/late binding* mechanism.

The work in [20] presents a middleware platform for Web services selection and execution, where local and global optimization constraints are considered.

In [6] the complexity of some variants of the SI-WSC problem is analyzed, while an overview of heuristic techniques can be found in [14]. In [19] the SI-WSC problem is modeled as a multiple choice multiple dimension knapsack problem and as a graph constrained optimum path problem. Ad-hoc efficient techniques are proposed to identify sub-optimal solutions of the problem, but composed processes which include only a single execution path are considered.

The work presented in [7] proposes a genetic algorithm for the solution of the SI-WSC problem. The work is based on the reduction formulas presented in [8]. In [11], the multi-objective evolutionary approach NSGA-II (Non-dominated Sorting Genetic Algorithm, see [12]) is implemented, which identifies a set of solutions Pareto optimal without introducing a ranking among different quality dimensions. Every identified solution is characterized by the fact that no other plans exist such that a quality dimension is improved without worsening the other ones.

The above mentioned works considered the optimization of a single instance of the composed process with a constant Quality of Service Profile. In [3], we considered varying QoS profiles and long lived execution of business processes. In this paper we extended [3] by considering the optimization of several instances of a business process as a single optimization problem.

7. CONCLUSIONS

This paper presents a novel approach for the solution of the well known WSC problem. Since successive composed Web services execution requests can have the same task to Web service assignment, we address the MI-WSC problem optimizing simultaneously the set of requests which will be submitted to the system in the successive time interval instead of independently computing a solution for each incoming request. Our optimization is built on an autonomic grid computing infrastructure, where the service selection is performed in order to guarantee quality metrics perceived by end-users. The number of instances submitted in the successive time interval is based on a prediction provided by the grid infrastructure. Since, within this approach the plan of execution of an incoming request is already pre-determined, the computation overheads needed to perform the optimization can be dramatically reduced. The MILP formulation of the MI-WSC problem requires a strong computational effort and we proposed a greedy heuristic which quickly computes a sub-optimum solution. Experimental results show that the heuristic provides good quality solutions significantly reducing the computational effort (up to a factor of five).

We leave as future work the more general case in which users can access various types of business applications which are described by different BPEL specifications.

Acknowledgments

The work reported in this paper has been partially supported by the DISCORSO FAR Italian Project. Thanks are expressed to Prof. Barbara Pernici for many fruitful discussions. We are grateful also to Simone Oliverio and Claudio Terraneo for development activities.

8. REFERENCES

[1] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian. Resource management in the autonomic service-oriented architecture. In *ICAC 2006 Proc.*, 2006. In Press.

[2] D. Ardagna, C. Cappiello, P. Plebani, and B. Pernici. A Framework for Describing and Supporting Adaptive Context-aware Web Services. Politecnico di Milano Technical Report 2006.48 http://www.elet.polimi.it/upload/ardagna/Tech2006-48.pdf, June 2006.

[3] D. Ardagna, G. Giunta, N. Ingraffia, R. Mirandola, and B. Pernici. Qos-driven web services selection in autonomic grid environments. In *OTM Conferences (2)*, pages 1273–1289, 2006.

[4] D. Ardagna and B. Pernici. Global and Local QoS Guarantee in Web Service Selection. In *BPM 2005 Workshops Proc.*, pages 32–46, 2005. Nancy.

[5] M. N. Bennani and D. A. Menasce. Assessing the robustness of self-managing computer systems under highly variable workloads. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing (ICAC'04)*, pages 62–69, Washington, DC, USA, 2004. IEEE Computer Society.

[6] P. A. Bonatti and P. Festa. On optimal service selection. In *WWW 2005 Proc.*, pages 530–538, 2005. Chiba.

[7] G. Canfora, M. Penta, R. Esposito, and M. L. Villani. QoS-Aware Replanning of Composite Web Services. In *ICWS 2005 Proc.*, 2005.

[8] J. Cardoso. Quality of Service and Semantic Composition of Workflows. PhD. Thesis, Univ. of Georgia, 2002.

[9] S. Chandrasekaran, J. A. Miller, G. Silver, I. B. Arpinar, and A. P. Sheth. Performance Analysis and Simulation of Composite Web Services. *Electronic Market: The Intl. Journal of Electronic Commerce and Business Media*, 13(2):120–132, 2003.

[10] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *SOSP 2001 Proc.*, pages 103–116, 2001. Banff.

[11] D. B. Claro, P. Albers, and J. K. Hao. Selecting Web Services for Optimal Composition. In *ICWS 2005 Workshop Proc.*, 2005. Orlando.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Comp.*, 6(2):182–197, 2002.

[13] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Intl. J. of Supercomp. App.*, 2001.

[14] M. C. Jaeger, G. Muhl, and S. Golze. QoS-aware composition of web services: An evaluation of selection algorithms. In *COOPIS 2005 Proc.*, 2005. Cyprus.

[15] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 1(31):41–50, 2003.

[16] D. Menasce and E. Casalicchio. QoS in Grid Computing. *IEEE Internet Computing*, July–Aug 2004.

[17] M. Ouzzani and A. Bouguettaya. Efficient Access to Web Services. *IEEE Internet Comp.*, 37(3):34–44, 2004.

[18] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. A comparative analysis between egee and griday workload management systems. In *OTM Conferences (2)*, pages 1143–1151, 2006.

[19] T. Yu and K. J. Lin. Service selection algorithms for composing complex services with multiple QoS constraints. In *ICSOC 2005 Proc.*, 2005. Amsterdam.

[20] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnamam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. on Soft. Eng.*, May 2004.