Per-flow optimal service selection for Web services based processes

Danilo Ardagna *, Raffaela Mirandola

Politecnico di Milano, Dipartimento di Elettronica e Informazione, Via Golgi 40, 20133 Milano, Italy

A R T I C L E   I N F O

Article history:
Received 1 September 2009
Received in revised form 8 March 2010
Accepted 18 March 2010
Available online 31 March 2010

Keywords:
Composite Web services
Quality of service
Optimization

A B S T R A C T

With the development of the Service-Oriented Computing (SOC) paradigm, flexible business processes can be defined from independently developed services. Multiple services corresponding to the same functionality but characterized by different Quality of Service (QoS) attributes can be offered by different service providers and the best set of Web services can be selected at run-time in order to maximize the QoS for end users.

In the literature many approaches have been proposed for the optimal service selection which is usually performed on a per-request basis, i.e., considering a single process invocation. In this paper we propose a broker-based framework which solves the optimal service selection on a per-flow basis. Multiple applications, defined as different BPEL processes are considered at the same time and multiple requests to the same process are optimized concurrently.

Service selection is formulated as a constrained non-linear multi-criteria optimization problem and an heuristic algorithm is proposed to determine a scalable and efficient solution.

A comparison with top performing state-of-the-art approaches for a number of different scenarios of interest is also provided. Results show that the overall optimization time reduction which can be achieved by our solution is proportional to the system incoming workload. Order of magnitudes optimization time improvements can be obtained if compared to alternative methods. Moreover, our solution is robust to workload prediction uncertainty.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

The Service-Oriented Computing (SOC) paradigm foresees the creation of business applications from independently developed services. In this vision, Service Providers (SPs) offer similar competing services corresponding to a functional description and the best set of Web Services (WSs) can be selected at run-time in order to maximize the Quality of Service (QoS) for end users. In these systems, applications are specified as BPEL processes defined by means of abstract Web services and the component services, i.e., concrete services, are invoked at run-time by implementing a late binding mechanism (Papazoglou et al., 2008; Ardagna and Pernici, 2007).

In the literature, many approaches have been proposed for concrete service selection (see, e.g., Claro et al., 2005; Yu et al., 2007; Ardagna and Pernici, 2007; Canfora et al., 2008; Alrifai and Risse, 2009; Liang et al., 2009) which have been formalized as optimization problems. Usually, the service selection problem is NP-hard (e.g., Yu et al., 2007) and these approaches perform the optimization considering a single BPEL process invocation, i.e., on a per-request basis. This means that the service selection is independently performed for each incoming request adding a significant overhead in the Service Oriented Architecture (SOA) infrastructure. Furthermore, Internet application workloads can vary by orders of magnitude even within the same business day (Chase et al., 2001). Hence optimization has to be performed when the BPEL process execution starts and has to be iterated at run-time in order to take into account workload fluctuations.

Optimizing a single request leads to a (possibly) optimal solution for that single invocation according to the current conditions of the execution environment. This could incur in problems under a sustained traffic of requests addressed to a complex SOA system. Indeed, the single request execution could conflict with the “optimum” execution of the set of concurrent requests, leading to instability and management problems. For example, if multiple service requests are assigned to the cheapest available concrete service, that service could be overloaded and its performance service may degrade. Finally, traditional approaches (Zeng et al., 2004; Ardagna and Pernici, 2007; Canfora et al., 2008; Alrifai and Risse, 2009) provide hard QoS constraints which usually imply the under utilization of the service infrastructure.

To overcome these limitations, in this paper we exploit and refine the ideas first presented by Cardellini et al. (2006, 2007) and we propose a broker-based framework which allows the optimal
service selection satisfying a set of QoS constraints on a per-flow basis. Multiple applications, defined as different BPEL processes are considered and multiple requests to the same process are optimized concurrently. In order to take into account environment and requirement changes, the optimization is performed periodically according to a prediction of the requests flow for a given time horizon. This prediction is based on the continuous monitoring of the system which can also trigger the optimization if requests incur in QoS violation or in case of WS component failures.

Service selection is formulated as a constrained non-linear multi-criteria optimization problem and several QoS attributes, such as execution time, cost, and reputation are considered. The problem solution is obtained through SNOPT (a commercial solver for non-linear programming (Gill et al., 2002)). In order to find the solution efficiently, we have developed an heuristic algorithm to determine the initial solver solution and to speed up the optimization step.

We have compared our solution to top performing state-of-the-art techniques (Ardagna and Pernici, 2007; Alrifai and Risse, 2009) for a number of different scenarios of interest. Results show that the overall optimization time reduction which can be obtained by our approach is proportional to the SOA system incoming workload and other literature solutions (Ardagna and Pernici, 2007; Alrifai and Risse, 2009) are outperformed even under light load conditions. Moreover, our solution is also robust to workload prediction variations.

The remainder of the paper is organized as follows. Section 2 discusses other literature approaches. An overview of the main components of our brokering framework is reported in Section 3. The quality and BPEL process models are introduced in Section 4. The service selection optimization problem formulation and solution are discussed in Section 5. Experimental results in Section 6 demonstrate the effectiveness of our approach, while conclusions are finally drawn in Section 7.

2. Related work

In SOA systems, building applications through the composition of available services is a key point. This composition involves several activities: (1) the definition of an integration schema yielding to the target application, (2) the selection of concrete services offering the required functionalities, and (3) the fulfillment of 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 WSs 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., Papazoglou et al., 2008; Menascé and Dubey, 2007). Example of SOA systems which allow the execution of QoS-aware BPEL processes can be found in Ardagna et al. (2007), Menascé et al. (2010) and Secse EU Project (xxxx).

Literature approaches can be classified into two main categories: composition by planning and business process optimization (Srivastava and Koehler, 2003). The former approach, proposed by the Semantic Web and AI communities, investigates the problem of synthesizing a complex behaviour from an explicit goal and a set of candidate services which contribute to a partial solution of the complex problem (i.e., the aim is supporting step 1 of SOA applications development). In the latter case (Patil et al., 2004; Zeng et al., 2004; Alrifai and Risse, 2009), complex applications are specified as BPEL processes and the best set of services are dynamically selected at run-time by solving an optimization problem (i.e., the focus is on steps 2 and 3). The Semantic Web and AI approach is very flexible since a BPEL process is built automatically or semi-automatically from a high level specification of the required functionality (Lazovik et al., 2006; Marconi et al., 2008; da Costa et al., 2004; Agarwal et al., 2005). Planning is very flexible, but it is usually computation intensive and, from the QoS point of view, only sub-optimal solutions can be identified (Lazovik et al., 2006).

In process optimization, vice versa, the process schema, i.e., the sequence of activities, is given and the optimum mapping of activities to component WSs candidate for their execution is identified. Process optimization has its roots in workflow scheduling problems where the mapping of tasks to resources has to be identified such that some temporal or resource constraints (i.e., agents which can support tasks executions) are met (Senkul and Toroslu, 2005). The literature has provided three generations of solutions. First generation solutions implemented local approaches (Zeng et al., 2004; Yu et al., 2007; Ardagna and Pernici, 2007) which select WSs one at the time by associating the running abstract activity to the best candidate service which supports its execution. Local approaches are very simple (the optimum solution can be identified by greedy algorithms), but they can guarantee only local QoS constraints, i.e., candidate WSs are selected according to a desired characteristic, e.g., the price of a single WS is lower than a given threshold. An example of first generation technique can be found in Maamar et al. (2003), where Web agents can also migrate to invoke services locally in order to minimize the network bandwidth.

Second generation solutions proposed global approaches (Zeng et al., 2004; Patil et al., 2004; Claro et al., 2005; Jaeger et al., 2005; Canfora et al., 2008). The set of services which satisfy the process constraints and user preferences for the whole application are identified before executing the process. In this way, QoS constraints can predicate at a global level, i.e., constraints posing restrictions over the whole BPEL process execution or for sub-sets of abstract services can be introduced. Second generation techniques are based on the solution of NP-hard optimization problems. In Bonatti and Festa (2005) the complexity of some variants of the global process optimization problem is analyzed, while an overview of heuristic techniques can be found in Jaeger et al. (2005). Global approaches have been proposed for the first time in Zeng et al. (2004), where the process optimization problem has been formalized as a mixed integer linear programming problem and solved by integer linear programming solvers. Some recent proposals face the process optimization problem by implementing genetic algorithms (Canfora et al., 2008; Claro et al., 2005). In Canfora et al. (2008), the reduction formulas presented in Cardoso (2002) are adopted and global constraints are guaranteed statistically. In Claro et al. (2005), the multi-objective evolutionary approach NSCGA-II (Non-dominated Sorting Genetic Algorithm) is implemented, which identifies a set of Pareto optimal solutions without introducing a ranking among different quality dimensions. More recently, in Yu et al. (2007) process optimization has been modeled as a multiple choice multiple dimension knapsack problem and as a graph constrained optimum path problem and efficient heuristic techniques have been proposed for the per-request approach. Wan et al. (2008) have also proposed an efficient recursive branch and bound algorithm.

Second generation solutions, requiring the solution of NP-hard problems, introduce a significant overhead in the system. To reduce optimization complexity, a number of solution have been proposed which guarantee global constraints only for the critical path (Zeng et al., 2004) (i.e., the path which corresponds to the highest execution time), or reduce loops to a single task (Canfora et al., 2008). Another drawback of second generation
solutions is that, if the end user introduces severe QoS constraints for the BPEL process execution, i.e., limited resources which set the problem close to unfeasibility conditions (e.g., limited budget or stringent execution time limit), no solutions could be identified and the BPEL process execution fails (Canfora et al., 2008). Furthermore, Web services QoS is subject to uncertainty and considering the worst case performance could be too conservative.

Third generation techniques (Ardagna and Pernici, 2007; Alrifai and Risse, 2009; Liang et al., 2009), try to overcome the limits of the previous approaches. In particular, the work of Ardagna and Pernici (2007) focuses on the execution of BPEL processes under severe QoS constraints. Authors have introduced loops peeling, which significantly improves previous solutions based on loops unfolding (Zeng et al., 2004; Canfora et al., 2008). Furthermore, negotiation is exploited if a feasible solution cannot be identified, to bargain QoS parameters with SPs offering services, reducing process invocation failures. The proposed approach is based on mixed integer linear programming and has been proven particularly efficient for large process instances.

Alrifai and Risse (2009) have proposed a hybrid global/local approach with the aim of reducing optimization complexity and allowing also a decentralized implementation of the WS composition. The proposed solution consists of two steps: mixed integer linear programming is initially adopted to find an optimal decomposition of global QoS constraints into local constraints. In a second phase, distributed local selection of the WSs that satisfy these local constraints is performed.

Recently, in Liang et al. (2009) a novel approach based on local search with the aim to maximize the QoS under probabilistic constraints has been proposed. The goal is to provide the solution which returns the best quality level $q^*$, such that the probability that the actual quality received by the end user falls below $q$ is within a prescribed threshold. Negotiation is also triggered at run-time if the prediction of the final QoS is below $q$.

All of the above mentioned works considered the optimization on a per-request basis and focus on the execution of a single instance of the BPEL process with a constant and conservative QoS profile. Since the number of submitted requests to a service center could be large and the QoS is highly dynamic (Andreolini et al., 2008), the fast selection of component WSs is particularly important.

This paper is based on the work presented in Cardellini et al. (2006, 2007) which has posed the basis to solve the service selection problem on a per-flow basis, considering the execution of multiple BPEL processes instances. With respect to Cardellini et al. (2006, 2007) our work considers BPEL processes with different schema which make the problem particularly challenging since the objective function becomes neither concave nor convex and no optimality guarantees can be provided. Furthermore, we consider explicitly the performance of component WSs and do not incur in the problem of overloading WS components. We evaluate WS performance in terms of response time and reputation which are particularly relevant since influence the SP revenues on the long-term (Rana et al., 2008; Almeida et al., 2006).

3. Reference architecture

The conceptual architecture considered in this work is illustrated in Fig. 1. SOA users submit to a federation of front-end brokers abstract BPEL processes annotated with quality specifications and constraints. An user request could be assigned to a given broker according to its location as in Le et al. (2009), or on the basis of load balancing criteria as in Boonea et al. (2010). Brokers cooperate serving requests and act on behalf of a significant number of customers (Serhani et al., 2005; Yu and Lin, 2005). Each broker is in charge of determining the optimum plan, i.e., the optimum mapping between the abstract and the concrete services on the basis of the QoS constraints established in SLA contracts and specified as annotations.

Concrete services candidate for the execution are retrieved from an extended UDDI registry (Plebani and Pernici, 2009), which stores service components’ WSDL descriptions and their QoS profile. The architecture includes also QoS Monitors which are responsible for collecting information about the service usage. For example perfor-
formance and availability of concrete services are periodically detected in order to update the end-to-end performance profiles in the QoS registry (Baresi et al., 2008).

At run-time, when BPEL processes are executed the invocations to concrete services are implemented through wrappers which can be dynamically generated as in Modafferi et al. (2006) and Ardagna et al. (2007).

Each broker identifies the optimum plan periodically (usually every half an hour (Pacifici et al., 2005; Ardagna et al., 2007)) according to a prediction of the frequency of execution of BPEL processes which are provided by Workload Analysers. Predictions are determined by analysing historical data as in Andreolini et al. (2008). The optimum plan is used for the next control time horizon to drive the incoming service invocations to the concrete services. However, a new optimum plan is triggered in case of failures of WS components or workload variations (e.g., the current workload differs from the prediction more than a given threshold) detected by QoS monitors and/or workload analysers.

In this paper we provide novel and efficient techniques to devise the optimum plan on a per-flow basis guaranteeing statistically global constraints.

4. System model

In this section the quality attributes considered in our framework and the BPEL process model are discussed.

4.1. Quality model

Several quality criteria can be associated with WSs execution. Furthermore, if the same WS is accessible from the same SP, but with different quality characteristics, then multiple copies of the same WS are stored in the registry, each copy being characterized by its quality profile.

In this paper the following subset of quality dimensions, which have been the basis for QoS consideration also in other approaches (Chandrasekaran et al., 2003; Zeng et al., 2004; Ouzzani and Bouguetaya, 2004), will be considered:

- **execution time** $e_i$: the expected delay, between the time instant when a request is sent to a Web service ($ws_j$ is invoked) and the time when the result is obtained. We assume that Provider publishes in the extended UDDI registry the maximum execution time $e_i$ for $ws_j$ invocation;
- **cost** $c_i$: the fee that a service requester has to pay to the Service Provider for the invocation of service $ws_j$;
- **reputation** $r_i$: defined as the probability that the execution time of $ws_j$ invocation is lower than the threshold $e_i$.

This quality model can be easily extended in order to include other quality dimensions. As discussed above, the quality profiles are stored in the extended UDDI registry and are updated by the QoS Monitor. Furthermore, as in Menascé and Dubey (2007), we assume that the SPs store the maximum service rate $\mu_i$, i.e., the maximum incoming workload which can be accepted by the SP. As in grid environments (Ardagna and Pernici, 2007; Le et al., 2009), we assume that each SP pre-allocates some resources to a given broker in order to provide QoS guarantees. In the following we will model each service $ws_j$ as an M/G/1 queue (Bolch et al., 1998) as Menascé and Dubey (2007) and Liu et al. (2001), and we assume that requests are served according to the processor sharing scheduling discipline which is common among Web services containers.

We adopt analytical models in order to obtain an indication of system performance, as in Pacifici et al. (2005) and Urgaonkar et al. (2007). There is a trade off between the accuracy of the model and the time required to estimate system performance. More accurate performance models have been provided in the literature for Web systems (see e.g., Cunha et al., 2007; Risika et al., 2002), but due to the analysis complexity only small size models based on a limited number of queues can be dealt with and cannot be adopted here.

Reputation values are periodically updated by the QoS Monitor (see Section 3). As it will be discussed in the following, each broker evaluates the average execution times according to workload predictions and the abstract to concrete service assignment with the aim to provide run-time performance guarantees.

In the following we will denote with $q_{in}$ the $n$th quality parameter, with $1 \leq n \leq N$.

4.2. Multi-class business process model

In our framework, BPEL specifications are annotated in order to provide statistics on processes executions. Each BPEL process is transformed in a Directed Acyclic Graph (DAG). Without loss of generality, we assume that BPEL processes have single starting and ending points, and the loops are peeled or unfolded before the analysis is computed as in Zeng et al. (2004), Ardagna and Pernici (2007) and Canfora et al. (2008). Hence, each DAG has a source and a sink node. An execution of the BPEL process consists of the invocation of the services on a path from the source to the sink.

The BPEL process model adopted in this paper is driven by Cardellini et al. (2006) and Cardellini et al. (2007). Each BPEL process defines a QoS class (indicated also simply with class in the following). We denote by $K$ the set of QoS classes, by $\gamma^i$ class $k$ requests incoming workload ($k \in K$), and by $\gamma^i = (\gamma^1, \ldots, \gamma^K)$ the overall user requests arrival rate to the broker. The main notation adopted in this paper is summarized in Table 7 of Appendix A.

For each class $k$ request, brokers assign to each abstract service $i$ a set of concrete services $ws_j$ such that the QoS for the flow of requests is maximized, while given global QoS constraints (i.e., constraints over the whole BPEL process execution) are guaranteed.

In Fig. 2, each macro-node depicted as a rectangular box represents an abstract service $i < r^k$ in the DAG. The directed edge from the macro-node $r$ to the macro-node $s$ represents a sequencing constraint; that is, it indicates that abstract service $r$ must complete before abstract service $s$ may begin.

![Fig. 2. Example of DAG for class $k$ process.](image-url)
Multiple edges exiting from a macro-node \( r \) are weighted by a probability, which provides statistical information about the next abstract service required by a client of the BPEL process. Probabilities can be provided and updated by the monitoring component or should be estimated by BPEL process designer for new applications. For the sake of simplicity, the parallel workflow execution pattern (van der Aalst et al., 2003) is not considered here but can be easily included in the optimization problem formulation, e.g., recurring to bounds for the evaluation of fork join models (Xia et al., 2007).

In the following we denote with \( p_i^s \) the probability expressing the frequency with which service \( s \) is executed after completion of service \( r \) in process \( k \). The probabilities have to be provided at design time on the basis of an initial guess (Smith and Williams, 2002) and can be refined at run time by using the historical data gathered by the QoS Monitor.

For each macro-node \( r \), \( \sum_{s:\text{service}} p_i^s = 1 \). If only one edge exits node \( r \), the probability is equal to 1 and we omit its value in the graph. Different QoS classes are characterized by a different BPEL process schema (and hence DAG) and/or probabilities.

Let \( \lambda_k^r \) be the rate of class \( k \) requests that arrive at the abstract service \( i \in \gamma^r \). Using well-known flow conservation arguments (Bolch et al., 1998), we get the following set of linear equations for the request rates, that can be used to calculate \( \lambda_k^r \):

\[
\lambda_k^r = P^T \lambda^r + \gamma_k^r e_1 \quad \forall k \in K,
\]

where \( \lambda^r = (\lambda_1^r, \ldots, \lambda_n^r) \) and \( e_1 = (1,0,\ldots,0) \) are column vectors and \( P^k \) is the \( |\gamma^r| \times |\gamma^r| \) routing probability matrix for class \( k \) requests. In the following we will denote by \( \lambda_k^r \) the flow of requests for the abstract service \( i \in \gamma^r \) given by the solution of Eq. (1).

Each DAG macro-node contains the concrete services \( ws_j, j \in J_i \) (shown in Fig. 2 as circles inside the rectangular box representing the abstract service), that correspond to specific implementations of a given abstract service \( i \in \gamma^r \). Let \( J = \cup_{k \in K} J_i^k \) be the set of indexes of all the concrete WS components managed by the broker. Finally, each QoS class is associated with:

- a set of normalized weights \( \{w^k_1, w^k_2, w^k_3\} \), \( w^k_1 + w^k_2 + w^k_3 = 1 \), indicating a relative priority among the set of quality dimensions for the BPEL process \( k \) end users;
- the global constraints, i.e., the maximum (minimum) values of QoS required for the BPEL process invocation (maximum execution time \( t_{\text{max}} \), maximum cost \( c_{\text{max}} \), and minimum reputation \( r_{\text{min}} \));
- QoS class \( k \) weight \( Q^k \), which denotes the BPEL process \( k \) relative priority, \( \sum_{k \in K} Q^k = 1 \).

As an example, Fig. 3 introduces two BPEL processes including seven abstract services which can be supported by eleven concrete WSs. Requests for the first and second BPEL process are classified as silver and gold classes, respectively.

The first BPEL process includes a simple sequence, while the second one introduces a switch. Each abstract service can be supported by two different candidate services while, in both cases, the last service is supported only by the concrete service \( ws_5 \).

The routing probability matrices for the two processes are

\[
P^1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad P^2 = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

Users in the gold class accept to pay a higher cost (the global constraint is set to \( c_{\text{max}} = 4 \)) and are interested mainly in the maximization of the process reputation and are characterized by \( w^2_1 = 0.8 \), \( w^2_2 = 0.1 \), \( w^2_3 = 0.1 \). Users in the silver class introduce stringent execution costs (the global constraint is set to \( c_{\text{max}} = 3 \)) and are interested mainly in the minimization of the execution time and are characterized by \( w^1_1 = 0.7 \), \( w^1_2 = 0.2 \), \( w^1_3 = 0.1 \).

Table 1 summarizes the system parameters.

5. Optimal service selection

In this section, the optimal service selection problem addressed by each broker will be presented. Section 5.1 formulates the optimal service selection as a non-linear optimization problem. Section 5.2 provides the optimization problem analysis and outlines the implemented solution technique.

5.1. Optimization problem formulation

The goal of individual brokers is to select, for each QoS class \( k \), the set of concrete services \( ws_j, j \in J_i^k \) that must be used to fulfill the abstract service \( i \) invocations in order to maximize the QoS perceived by the overall flow of requests, while guaranteeing global constraints.

In our approach, service selection is performed probabilistically and constraints are guaranteed statistically. The decision variables of the problem are \( x_i^k \) which denote the probability that the concrete service \( ws_j, j \in J_i^k \) will be invoked by the QoS class \( k \) when the workflow reaches the stage indicated by the macro-node \( i \). Given a flow of requests \( \lambda_i^k \), the abstract service \( i \) requests are splitted among the corresponding concrete services \( j \in J_i^k \) according to the \( x_i^k \) probabilities. Hence, \( x_i^k \lambda_i^k \) is the incoming workload for the concrete service \( ws_j \) generated by clients belonging to the QoS class \( k \).

The QoS levels experienced by class \( k \) users depend on both the total request flow \( \lambda_i^k x_i^k \) addressed to each concrete service, and by the value of the concrete service quality attributes.

Under the M/G/1 assumption, the average execution time of each concrete service \( j \) can be computed as:

\[
e_j = \frac{1}{\mu_j} \frac{1}{1 - \sum_{k \in K} \sum_{i \in J_i^k} x_i^k \frac{\mu_j}{\mu_j}},
\]

Note that, given a matrix \( x \), the term \( \sum_{k \in K} \sum_{i \in J_i^k} x_i^k \frac{\mu_j}{\mu_j} \) indicates the overall utilization \( U_j \) of the M/G/1 queue modelling \( ws_j \) due to the execution of the abstract services of all BPEL processes in \( K \).

Differently from Cardellini et al. (2006) and Cardellini et al. (2007), we explicitly define the reputation \( r_j \) of a concrete service \( j \) as the probability that the execution time exceeds a given threshold \( e_j \), which is given by:
Table 1  Parameters of the simple case study.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\gamma_1, \gamma_2))</td>
<td>(0.1, 0.01)</td>
</tr>
<tr>
<td>((\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7))</td>
<td>(0.2, 0.4, 0.8, 0.5, 0.3, 0.3, 0.4, 0.8, 0.9, 0.9, 0.3)</td>
</tr>
<tr>
<td>((c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8))</td>
<td>(0.9, 1.3, 1.2, 0.3, 0.8, 1.1, 0.5, 0.7, 1.2, 1.8, 2.6)</td>
</tr>
<tr>
<td>((r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8))</td>
<td>(0.99, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999)</td>
</tr>
<tr>
<td>((\omega_1, \omega_2))</td>
<td>(0.3, 0.7)</td>
</tr>
<tr>
<td>((c_{\text{min}} - c_{\text{max}}))</td>
<td>(3.4)</td>
</tr>
</tbody>
</table>

The reputation (see Section 4.1) is very relevant and may vary over time depending on the SP’s ability to satisfy incoming requests. If \(\theta_i\) threshold is violated, the reputation of the SP decreases, and this could lower the workload and SP’s revenue in the long-term (Almeida et al., 2006).

According to Eq. (2), the average execution time of an abstract service \(i \in \mathcal{A}\) of class \(k\) can be computed as:

\[
\text{exTime}_i = \sum_{j \in \mathcal{J}} \frac{N_k}{j_k^i} \sum_{j \in \mathcal{J}} \left( x_{ij}^k / j_k^i \right) = \sum_{j \in \mathcal{J}} \frac{x_{ij}^k}{j_k^i} - \sum_{j \in \mathcal{J}} \left( x_{ij}^k / j_k^i \right).
\]

Note that, \(x_{ij}^k / j_k^i\) is the mean number of class \(k\) invocations to the \(i\)th abstract service.

In the following, we indicate with \(\bar{E}_i\), \(\bar{C}_i\), and \(\bar{R}_i\) respectively the average value of execution time, cost, and reputation for the class \(k\) BPEL process which can be evaluated as follows (Ardagna and Pernici, 2007; Zeng et al., 2004):

\[
\bar{E}_i(x) = \sum_{i \in \mathcal{A}} \text{exTime}_i,
\]

\[
\bar{C}_i(x) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{J}} \frac{x_{ij}^k}{j_k^i} - \sum_{j \in \mathcal{J}} \left( x_{ij}^k / j_k^i \right),
\]

\[
\bar{R}_i(x) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{J}} \frac{x_{ij}^k}{j_k^i}.
\]

\(\bar{C}_i\) is computed as the sum of the average cost of the invoked services. Similarly, \(\bar{E}_i\) and \(\bar{R}_i\) are computed as the sum of the average execution time and reputation of the invoked concrete services and are evaluated under the hypothesis of the BCMP theorem (Bolch et al., 1998) as in Menascé and Dubey (2007) and Cardellini et al. (2006). Note that, if the conditions of the BCMP theorem do not hold, (2) can still be adopted as a measure of the congestion at the concrete service \(w_{ij}\) and can be used to avoid highly congested nodes (Bertsekas and Gallager, 1991).

In general, each broker has to mediate among multiple QoS attributes, which can be either mutually independent or possibly conflicting (e.g., usually the lower the execution time and the higher the cost). Therefore, the optimal service selection results in a multi-objective optimization.

Some literature proposals (see, e.g., Claro et al., 2005) address the problem as a multi-objective problem identifying the Pareto frontier of multiple solutions. In our approach as in Zeng et al. (2004), Cardellini et al. (2006) and Ardagna and Pernici (2007), the multi-objective problem is transformed into a single objective problem by applying the Simple Additive Weighting (SAW) technique (Hwang and Yoon, 1981), one of the most widely used techniques to obtain a score from a list of dimensions. In this way, a single (possibly) global optimum solution can be computed.

Given a matrix \(x\), we denote by \(F(x)\) the score (i.e., the aggregated value of QoS for the multiple quality attributes) for class \(k\). Since, quality dimensions \(q_k\) have different units of measure, the SAW method first normalizes the raw values for each quality dimension. Each quality dimension \(q_k\) is also associated with a weight \(w_k\) in the BPEL process model (see Section 4.2). Let us denote with \(q_k^{\text{min}}\) the value of the nth quality dimension for the BPEL process \(k\). The score \(F(x)\) is then calculated as a weighted sum of the normalized values of quality dimensions.

In the normalization phase, positive quality criteria (i.e., attributes such that the higher the value the higher the quality) are scaled as follows:

\[
F_k(x) = \begin{cases} q_k^{\text{max}} - q_k & \text{if } q_k^{\text{min}} = q_k^{\text{max}} - q_k^{\text{min}}, \\ q_k^{\text{max}} - q_k^{\text{min}} & \text{if } q_k^{\text{max}} = q_k^{\text{min}}. \end{cases}
\]

\[F_k(x) = \sum_{i \in \mathcal{A}} \left( q_k^{\text{min}} - q_k^{\text{max}} \right) w_k \text{exTime}_i(x).
\]
The objective function of the optimization problem is obtained by considering the weighted average of functions $F_i^k(x)$, where each function is weighted by the relative importance of class $k$, $\ell^k$ multiplied by the incoming workload $\gamma^k$:

$$F(x) = \frac{\sum_{k=1}^{K} \ell^k F_k^k(x)}{\sum_{k=1}^{K} \ell^k}.$$ (11)

In this way, the BPEL processes characterized by the higher number of completions and/or weighted $\ell^k$ are favoured. The service selection problem can be modelled by the following non-linear problem:

$$(P1) \quad \max \: F(x) = \sum_{k=1}^{K} \ell^k F^k(x),$$

$$\sum_{j \in \theta} x^k_j = 1; \quad \forall \: k \in K, \, i \in \tau^{-k},$$ (12)

$$\sum_{k \in \mathcal{K}} \sum_{j \in \tau^{-k}} x^k_j \mu_j < 1; \quad \forall \: j \notin \theta,$$ (13)

$$E^k(x) \leq e^k_{\text{max}}; \quad \forall \: k \in K,$$ (14)

$$C^k(x) \leq c^k_{\text{max}}; \quad \forall \: k \in K,$$ (15)

$$R^k(x) \geq r^k_{\text{min}}; \quad \forall \: k \in K,$$ (16)

$$0 \leq x^k_j \leq 1; \quad \forall \: k \in K, \, i \in \tau^{-k}, \, j \in J^k.$$ Constraints family (12) guarantees that all abstract service invocations are served by concrete services. Constraints family (13) entails the equilibrium condition for M/G/1 queues, while constraints families (14)–(16) are the global QoS constraints.

If we consider again the example reported in Fig. 3, in the optimum solution the abstract services of the gold class are executed by the candidate services with highest performance and reputation ($w_{s_6}, w_{s_5}, w_{s_3}$, and $w_{s_1}$), i.e., the assignment is deterministic and the total cost is 3.85$. For the silver class, the first abstract service is executed by $w_{s_2}$ which is the fastest concrete service available, the node $i = 3$ is executed by $w_{s_5}$, while the macro-node $i = 2$ is executed by a probability $x^S_{42} = 0.7$ by the fastest concrete service available ($w_{s_6}$) and with a probability $x^S_{42} = 0.3$ by service $w_{s_5}$, with a total cost equal to the global constraint 3.85$.

5.2. Optimization problem analysis and solution

Problem (P1) is a non-linear optimization problem in the continuous decision variables $x^k_j$. The following proposition holds:

**Proposition 1.** The objective function of problem (P1) is neither concave nor convex.

**Proof 1.** See Appendix B. □

According to Proposition 1, only sub-optimal solutions can be determined for (P1) with no optimality guarantees. Our approach is based on SNOPT, a general-purpose solver for continuous constrained non-linear programming, which is particularly efficient for sparse problems.

SNOPT uses an iterative Sequential Quadratic Programming (SQP) algorithm (Gill et al., 2002) where an augmented Lagrangian function is reduced along each search direction to ensure convergence. The basic structure of the SQP algorithm starts from an initial solution that satisfies only the linear constraints. The SQP procedure converges to a point that satisfies the non-linear constraints and the first-order conditions for optimality (i.e., identifies a feasible solution which is also a local optima).

In order to speed up the optimization process, we have developed a greedy heuristic which provides to SNOPT a promising initial solution better than the one obtained with the standard SQP approach. The `initialSolution`() procedure is reported in Algorithm 1.

The BPEL processes are initially ordered according to non-decreasing value of the product $\ell^k \gamma^k$. In this way the BPEL processes with the higher priority and/or higher incoming workload are favoured. Abstract services of BPEL processes are then considered in lexical order.

In the following, the `initialSolution()` procedure iteratively selects an abstract service $i$ which is assigned to a concrete service $w_j$, $j \in J^k$. Concrete services are considered according to non-decreasing value of the aggregated QoS metric:

$$qos_j = w^k_j \max_{j \in J^k} \frac{1}{\mu_j} - \frac{1}{\mu_j},$$

$$+ w^k_j \max_{j \in J^k} \frac{1}{\mu_j} - \frac{\min_{j \in J^k}}{\mu_j} + C_j,$$

$$+ w^k_j \max_{j \in J^k} C_j - \min_{j \in J^k} C_j,$$

$$+ \frac{1 - e^{-\mu_j} t_j}{\min_{j \in J^k} (1 - e^{-\mu_j} t_j)} - \min_{j \in J^k} \frac{1 - e^{-\mu_j} t_j}{\min_{j \in J^k} (1 - e^{-\mu_j} t_j)}.$$ (17)

Eq. (17) evaluates the aggregated value of QoS of each concrete service $w_j$, and corresponds to Eq. (10), where the SAW technique is applied locally at each component service instead of the global BPEL process.

The process is repeated until $x^k_i$ is completely allocated, guaranteeing that $w_j$ utilization $U_j$ is at most $\Omega$ (steps 12–15). If $x^k_i$ cannot be allocated completely to the single concrete service $w_j$ (steps 16–19), then the workload is partially assigned to $w_j$, $w_i$ is saturated (i.e., $U_i = \Omega$), and the remaining workload $A^{res}_i$ computed in step 18 will be allocated on subsequent concrete services.

### Algorithm 1: Evaluation of SNOPT Initial Solution

1. Order the set $K$ according to non-decreasing $\Omega^k$ value;
2. $x^0_i \leftarrow 0 \forall k \in K, i \in \theta, j \in J^k$;
3. $U_j \leftarrow 0 \forall j \notin \theta$;
4. for $k \in K$ do
5.   for $i \in \theta$ do
6.     $A^{res}_i \leftarrow x^0_i$;
7.     while $A^{res}_i > 0$ do
8.       $j = \arg \max_k x^i_k \in \Omega$, $j \in J^k$;
9.       if $j = \text{null}$ then
10.          exit ()
11.       end
12.       if $(U_j + \frac{\mu_j}{\mu_j} w_{s_j}^k < U_j)$ then
13.         $x^k_j \leftarrow \frac{A^{res}_i}{\mu_j}$;
14.         $U_j \leftarrow U_j + x^i_k w_{s_k}^k$;
15.         $A^{res}_i \leftarrow 0$;
16.       else
17.         $x^k_j = \frac{A^{res}_i}{\mu_j}$;
18.         $A^{res}_i \leftarrow A^{res}_i - x^k_j w_{s_j}^k$;
19.         $U_j \leftarrow \Omega$;
20.       end
21.   end
22. end
23 end
If the overall incoming workload is such that the initialSolution () cannot find an assignment, then the procedure terminates (step 10) and the standard SNOPT SQP algorithm is used.

The overall complexity of the initialSolution () procedure is $O\left(\max\left\{ |K| \cdot \log( |K| ) , |K| \cdot \text{max}_{x_k \in \mathcal{E}} |V^k| \cdot \text{max}_{x_k \in \mathcal{E}} |J_k| \cdot |J_k^*|^2 \right\} \right)$, where the first term is the sorting time complexity of BPEL processes, while the second term is given by $|K| \cdot \text{max}_{x_k \in \mathcal{E}} |V^k| \cdot \text{max}_{x_k \in \mathcal{E}} |J_k| \cdot |J_k^*|$, which is the maximum number of iterations of the while loop, times max$_{x_k \in \mathcal{E}} |J_k^*|$, which is step 8 complexity.

6. Experimental analysis

Our approach has been tested on a wide set of randomly generated problem instances. Section 6.1 reports the results of experiments performed to evaluate the performance properties and scalability of our algorithm. Section 6.2 compares the per-flow approach proposed in this paper to alternatives (Ardagna and Pernici, 2007) and B (Alrifai and Risse, 2009), which are top performing per-request state-of-the-art solutions. Unfortunately in the other third generation solution (Liang et al., 2009), the performance of the optimization algorithm have not been investigated in depth (optimization time is reported only for instances of size $|J| = 30$, $|J| = 150$) and hence it will not be considered in our quantitative evaluation. Finally, we discuss the sensitivity of our approach with respect to the prediction of the incoming workload.

6.1. Optimization algorithm performance

To evaluate the performance of our algorithm we have performed a set of tests with the aim to evaluate:

- how the execution time varies with the size of the problem;
- the speed up which can be obtained by providing to SNOPT the initial solution obtained through Algorithm 1 with respect to the standard SNOPT SQP approach;
- the quality of our final solution with respect to the standard SNOPT SQP approach.

The scalability analysis has been performed considering a large set of randomly generated problem instances. The optimization problem parameters have been selected according to the values usually adopted in the literature (Ardagna and Pernici, 2007; Alrifai and Risse, 2009; Canfora et al., 2008). In particular parameters have been randomly generated using an uniform distribution and the ranges are reported in Table 2.

Algorithm 1 performance depends on the utilization threshold $U$ adopted. The best value obtained experimentally has been 0.7 according also to the best practice which suggests to limit the utilization of physical systems between 0.6 and 0.8 (Cherkasova and Phaal, 2002; Rolia et al., 2006). Overall the number of candidate concrete services has been varied between 2,500 and 100,000 as in Ardagna and Pernici (2007), Emmerich et al. (2006) and Alrifai and Risse (2009). The maximum service rate $\mu_j$ has been varied between 0.125 and 10 (Menascé and Dubey, 2007; Abrahao et al., 2006), the concrete services response time threshold have been set equal to $\epsilon_i = 10/\mu_i$ (Ardagna et al., 2007), while, as in Ardagna and Pernici (2007), concrete services cost $c_j$ has been set proportional to the service rate $c_j = a_j \times \mu_j$, $j \in J_k$. The constant factor $a_j$ has been randomly generated assuming an uniform distribution in the range [0.1,1] in order to differentiate abstract service costs (otherwise costs depend only on component service performance and not on the application logic implemented). Finally, the set of weights $w_i$ was randomly generated and weights were adjusted to sum 1.

Analyses have been performed on a 2 GHz Intel Pentium-D Workstation with 2 GB of RAM. The execution time of our overall approach includes the time required to Algorithm 1 to compute the initial solution and the SNOPT optimization time. If it will not be differently stated, in the following the execution time reported are obtained as the average execution time of the solution of 25 randomly generated instances characterized by the same size.

The execution time is primarily a function of the number of decision variables $x_k^i$, which can be evaluated as:

$$x_k^i = |y^k| \times |r^k| \times (1 - \text{AbsShare}),$$

where $\text{AbsShare}$ denotes the percentage of abstract services shared among different processes. The overall execution time for the solution of problem instances with different value of $|x_k^i|$ is reported in Table 3. As the results show, our algorithm can solve within the autonomic computing half an hour time constraint (see Section 3 (Pacifici et al., 2005; Ardagna et al., 2007)) problem instances up to 110,000 variables. Such instances correspond, for example, to problems with $|K| = 100$ BPEL processes, $\sum_{j \in J_k} |r^k| = 10,000$ abstract services (100 per BPEL process), 22 candidate concrete services per abstract service with $\text{AbsShare} = 50\%$.

By varying the node out degree NodeOutDeg (i.e., the number of edges exiting on average from DAG nodes), we have verified that

Table 3

| $|x_k^i|$ | Algorithm 1 | SNOPT | Overall time |
|--------|-------------|-------|-------------|
| 2717   | 0.3         | 9.8   | 10.1        |
| 5405   | 1.6         | 11.5  | 13.1        |
| 12,821 | 4.5         | 16.9  | 21.4        |
| 27,027 | 18.9        | 133.9 | 152.8       |
| 67,568 | 122.1       | 330.8 | 452.9       |
| 110,000| 370.4       | 650.1 | 1,020.5     |
| 160,000| 600.2       | 2,110.1 | 2,710.3    |

Fig. 4. Algorithm 1 execution time as a function of AbsShare.
the performance of our solution does not depend on the structure of the processes schema, e.g., the number of execution path of the BPEL processes. Furthermore, the overall execution time does not depend on the overall incoming workload but primarily depends only on \( |x^0_j| \).

The plot depicted in Fig. 4 shows the overall time required to compute a solution as a function of AbsShare in a test case characterized by \( |x^0| = 1,000 \), \( |f^0| = 4 \), while AbsShare is varied between 5 and 50%. As the results show, the execution time increases as could be expected since the competition to access the shared resources, i.e., the concrete WSs, increases. Note that, if AbsShare = 0%, each process can be optimized separately, and in that case the objective function becomes convex and the global optimum solution can be found by example by the approach proposed in Cardelli et al. (2006).

As a final analysis, we have evaluated the impact of the value of the global constraints on the optimization time. In particular we have considered the problem of minimization of execution cost \((w, = 1)\) by reducing the constraint on the global execution time \( E_{\text{max}} \) for a single process. Table 4 shows the average value of 10 randomly generated instances with the same value of \( |x^0| \). The first column corresponds to the lower bound of \( E_{\text{max}} \) which allows to determine a feasible solution and hence corresponds to a problem instance characterized by very stringent global constraints. The second column corresponds to the solution of the same instances where the global constraint has been set \( E_{\text{max}} = 2 \times E_{\text{max}} \). As the results show, the time increases if the constraints are more stringent but remains of the same order of magnitude without increasing exponentially.

| \( |x^0| \) | Stringent constraints | Soft constraints |
|--------|----------------------|----------------|
| 2703   | 10.69                | 5.34           |
| 5435   | 20.67                | 14.2           |
| 14,085 | 64.00                | 46.03          |
| 42,254 | 69.00                | 27.65          |
| 58,824 | 240.00               | 91.00          |
| 81,967 | 1969.00              | 941.00         |

In order to perform a comparison with the standard SNOPT SQP approach we have considered 40 randomly generated instances of different sizes and we have evaluated the overall execution time and final objective function value. Our approach speed up has been evaluated as

\[
speedup = \frac{\text{SNOPT SQP Execution Time}}{\text{Our Solution Execution Time}}
\]

while the objective function percentage variation is give by:

\[
\Delta f_{\text{obj}}\% = \frac{f_{\text{obj}} - \text{SNOPT SQP} f_{\text{obj}}}{\text{SNOPT SQP} f_{\text{obj}}}.
\]

Results are reported in Figs. 5 and 6, respectively. As the plots show, our approach obtains a speed up which is on average 2.92 while the percentage variation of the final objective function value is lower than 0.5% in the worst case. Hence, we can conclude that our approach allows obtaining a significant performance improvement with respect to SNOPT standard SQP without worsening the quality of the final local optimal solution.

### 6.2. Per-flow vs. per-request performance comparison

The per-request optimization approach is intrinsically more simple than the per-flow based solution proposed in this paper. Hence, one could expect that under light load conditions the overall optimization time required for optimizing subsequent BPEL process invocations could be significantly lower than the optimization time required by the per-flow approach. In this section we assume to reiterate periodically the process optimization with a time period equal to one hour.

In order to perform a quantitative comparison between the two approaches, alternative A (Ardagna and Pernici, 2007) and B (Alrifai and Risse, 2009) will be considered as reference implementations of the per-request approach. These works are among the most representative contributions currently available in the literature.

In particular, the test comparison with alternative A is performed on the same physical machine. The comparison with alternative B is performed considering the data provided for the random data set published in Alrifai and Risse (2009) (see the paper for further details). Alternative B analyses are performed on a slightly different machine which anyway is characterized by similar performance of ours according to CINT and CFP SpecCPU2006 benchmark results.1 Hence, the execution times reported in Alrifai and Risse (2009) are compared directly with the execution time obtained in our experimental setting.

The results of the comparison with alternative A are reported in Table 5. Different values of \( I \) and \( |f^0| \) for the optimization of a single process have been considered. Each table row reports the average value obtained for the solution of 25 randomly generated problem instances. The first column in the table specifies the process instance size, the second column reports the optimization time in seconds of the approach presented in this paper, the third reports the optimization time in seconds required for the solution of a single instance of alternative A, the fourth expresses the break-

1 See http://www.spec.org/cpu2006/
The results of the comparison with alternative B are similar and (Ardagna et al., 2007; Urgaonkar et al., 2007; Zhu et al., 2009). Large service centers environments are characterized by hundreds of requests per second (Urgaonkar et al., 2007). Note that, large service center environments are characterized by hundreds of requests per minute. As the plot shows, the Opt. Time Ratio increases linearly with the incoming workload and even an order of magnitude improvement can be obtained for light loads outperforming per-request approaches. For example, if we consider a process characterized by \( |\gamma^k| = 10 \) and \( |J_k^i| = 500 \) for all \( i \) for alternative A solution with a request rate \( \gamma_k = 3.3 \text{req/min} \) and we observe the system for one hour, overall 200 process instances have to be optimized. The optimization time requested by the alternative approach is equal to 37.8 s, while our solution requires only 19.3 s. Hence, the optimization time ratio is equal to 1.95 and overall our approach reduces the system overhead by almost 100%.

6.2.1. Sensitivity to workload prediction

Results presented so far consider the overall requests incoming workload as given. \( \gamma \) is the result of the prediction of the exogenous arrival rates for the next optimization time interval (see Section 3). As in Kusic et al. (2008), in order to evaluate the performance of our framework, we compared the results which can be obtained by our approach with the results obtained by an oracle that has a perfect knowledge of the future. Results show that the delta between the objective function value determined by the oracle and the one obtained by our solution under uncertainty, ranges between 1% and 2% (1.6% in average) to a 10% prediction error of the workload for the whole set of QoS classes. Hence, the solution is robust to workload prediction variations.

7. Conclusions

In this paper we have presented a broker-based framework for the optimal service selection of multiple BPEL processes. The optimization is performed on a per-flow basis and an efficient and scalable algorithm able to identify a local optimum solution for multiple process instances executed concurrently is provided. A comparison with top performing state-of-the-art approaches have shown that our solution allows reducing the overall optimization time required by the SOA infrastructure significantly, up to orders of magnitude. Our solution is also robust to workload prediction variations.

This paper is part of an ongoing research. Future work concerns the evaluation of the framework on real case studies and the analysis of a multi-broker scenario, where the brokers compete instead of cooperating in the use of the same concrete services.

**Acknowledgements**

Particular thanks are expressed to Dr. Mohammad Alrifai and Dr. Thomass Risse for providing Alternative B data and allow per-

---

**Table 5**

Performance comparison with alternative A per-request approach.

| Test case \((|\gamma^k|, |J_k^i|)\) | Opt. time | Alternative A opt. time | Break-even rate (n/min) |
|-------------------------------|-----------|------------------------|-------------------------|
| 10-10                         | 16.10     | 0.17                   | 47.65                   |
| 10-20                         | 6.60      | 0.17                   | 38.57                   |
| 10-25                         | 5.80      | 0.17                   | 17.21                   |
| 10-50                         | 14.30     | 0.29                   | 15.14                   |
| 10-100                        | 19.60     | 2.10                   | 9.33                    |
| 10-200                        | 144.30    | 5.38                   | 26.82                   |
| 10-250                        | 149.60    | 4.54                   | 32.95                   |
| 10-500                        | 451.30    | 19.88                  | 22.70                   |
| 500-10                        | 444.90    | 4.54                   | 98.00                   |
| 500-20                        | 1000.05   | 35.06                  | 82.52                   |
| 10000-10                      | 570.15    | 113.92                 | 6.52                    |

**Table 6**

Performance comparison with alternative B per-request approach.

| Test case \((|\gamma^k|, |J_k^i|)\) | Opt. time | Alternative B opt. time | Break-even rate (n/min) |
|-------------------------------|-----------|------------------------|-------------------------|
| 10-10                         | 16.10     | 0.17                   | 47.65                   |
| 10-20                         | 6.60      | 0.17                   | 38.57                   |
| 10-25                         | 5.80      | 0.17                   | 17.21                   |
| 10-50                         | 14.30     | 0.29                   | 15.14                   |
| 10-100                        | 19.60     | 2.10                   | 9.33                    |
| 10-200                        | 144.30    | 5.38                   | 26.82                   |
| 10-250                        | 149.60    | 4.54                   | 32.95                   |
| 10-500                        | 451.30    | 19.88                  | 22.70                   |
| 500-10                        | 444.90    | 4.54                   | 98.00                   |
| 500-20                        | 1000.05   | 35.06                  | 82.52                   |
| 10000-10                      | 570.15    | 113.92                 | 6.52                    |

---

Fig. 7. Overall optimization time ratio as a function of the incoming workload.
Theorem 6. Problem (P1) is concave if and only if

\[ E(x) = \frac{x_1^2}{\gamma_1} + \frac{x_2^2}{\gamma_2} \]

is strictly positive. The determinant of the Hessian \( H \) is given by:

\[ |H| = (\lambda_1^2)(\lambda_2^2) \frac{\lambda_1^2 - \lambda_2^2}{\lambda_{12}} \frac{\lambda_1^2 - \lambda_2^2}{\lambda_{12}} \]

The scalar term is strictly greater than zero and will be omitted in the Hessian analysis. Indeed \( \lambda_1^2 > 0, (\lambda_2^2) > 0 \) and \( (1 - \frac{\lambda_1^2}{\lambda_{12}} - \frac{\lambda_2^2}{\lambda_{12}}) > 0 \) for the equilibrium condition (13). With the same arguments the first element of \( H_1, 2(\lambda_1^2)(1 - \frac{\lambda_1^2}{\lambda_{12}}) \) is strictly positive. The determinant of \( H_1 \) is given by:

\[ |H_1| = (\lambda_1^2)(\lambda_2^2) \frac{\lambda_1^2 - \lambda_2^2}{\lambda_{12}} \frac{\lambda_1^2 - \lambda_2^2}{\lambda_{12}} \]

Hence, for the Sylvester criterion (Horn and Johnson, 1985), \( E(x) \) is neither concave nor convex as in general the objective function of Problem (P1). \( \text{q.e.d.} \)

### References


