Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

Service selection in web service compositions optimizing energy consumption and service response time

Yanik Ngoko[1*], Alfredo Goldman[2] and Dejan Milojicic[3]

Abstract

A challenging task in Web service composition is the runtime binding of a set of interconnected abstract services to concrete ones. This question, formulated as the service selection problem, has been studied in the area of service compositions implementing business processes. Despite the abundance of work on this topic, few of them match some practical needs that we are interested in. Indeed, while considering the business process implemented by service compositions, we can distinguish between two classes: compositions that correspond to single business process and those implementing multiple communicating processes. While most of the prior work focuses only on the first case, it is the latter that interests us in this paper. This paper contributes to the service selection by proposing a new algorithm that, in polynomial time, generates a mixed linear integer program for optimizing service compositions based on the service response time and the energy consumption. The novelty in this work is our focus on multi-process composition and energy consumption. The paper also proposes a new analysis of the service selection and an evaluation of the proposed algorithm.

Keywords: Web service composition; Service selection problem; Business Process Modeling Notation (BPMN)

1 Introduction

The prevalent techniques for building Web service compositions (WSCs) in middleware distinguish between two levels of service manipulation [1,2]. At the upper level, the middleware manipulates abstract services, defined through an interface of operations and a behavioral specification. This one might be expressed by using for instance OWL-S [3], WSDL-S [4] or DAML-S [5]. At this abstract level, we also have WSCs, here defined as a set of Web-based interactions over services operations. Underneath, there is a concrete level made of published Web services (WSs). In order to run the WSC, the middleware must at runtime associate each abstract operation with a concrete one. Our paper focuses on this aspect.

The automation of runtime binding in service composition has been addressed in previous work. The viewpoint that we adopt for its implementation is inspired by the work of Lee [2] and Ben Mokhtar et al. [1]. There are two successive tasks to be done for runtime binding. The first is the determination of the functional bindings of each abstract operation. This is done by comparing the specification defined for these operations, with published information, available on concrete WSs. For each abstract operation, the execution of this task returns a set of concrete operations that meets its specification. If this set is empty for an abstract operation, then the service composition is not realizable. Assuming that we have a realizable composition, the second task consists of finding the functional bindings that can result in an optimal composition, with respect to some QoS parameters (e.g. availability, service response time, price, energy consumption). Usually, we may have several abstract operations to bind, implying a combinatorial problem.

The work described in this paper is part of a middleware project [6] that aims at implementing both of these tasks (determining functional bindings and optimal binding choice) on ultra large scale compositions of WSs. Our study however only focuses on the second binding challenge. More precisely, given a realizable WSC, we seek the optimal concrete services for running it in order

*Correspondence: yanik.ngoko@lipn.univ-paris13.fr
[1] Laboratoire d'Informatique de Paris Nord, Villetaneuse, France
Full list of author information is available at the end of the article

to optimize the QoS of the composition. We focus, in this paper, on service response time (SRT) and energy consumption (EC).

The question of optimal binding regarding QoS has been addressed in previous studies under the service selection problem. In most of these work, the problem modeling considers two perspectives. From the middleware perspective, there is a global penalty function that on each WSC returns the QoS aggregate of its services constituents. The binding to find must minimize this function. From the perspective of the user, for each QoS dimension, a minimal performance must be guaranteed. This reduces the set of feasible solutions accepted for the problem. Both perspectives have a practical justification. While indeed the middleware optimizes its global performance, users set Service Level Agreement (SLA) for the minimal performance to be met. There are various studies for the service selection addressing SLAs [7-12]. However, few of them match our needs. This is made clear when considering the business process viewpoint of WSCs. Here, compositions belong to two classes: those that correspond to a single enterprise business process and those corresponding to multiple communicating enterprise processes [13]. While previous work on service selection addressed the first case, we are interested in the multi-process case.

In Figure 1, we illustrate a single and a multi-process. As stated by Goldman et al. [14], the multi-process case introduces other challenges such as the cost of inter-process communication (communication between E and F in Figure 1) and the distribution of the service composition graph. In addition, our work differs from existing ones in finding the optimal binding in regard to both SRT and the EC. While the SRT is a common QoS parameter used in service selection, it is not the case with EC.

In this paper, we consider the optimal binding between abstract and concrete operations in a WSC corresponding to multiple communicating business processes. The binding must be optimized for the SRT and EC. We address this problem by extending the linear programming-based solution introduced by Goldman et al. [14] to EC prediction. Given a service composition, our global contribution is a polynomial time algorithm for generating a Mixed Linear Integer Program (MILP) whose execution will return the optimal binding for the composition. In a detailed viewpoint, we contribute by: (1) extending our prior model [14] to service selection related to SRT and EC; (2) analyzing the complexity and the feasibility of our modeling; (3) providing an analysis of the service selection problem; (4) proposing an evaluation of our solution.

The remainder of this paper is organized as follows. In Section 2, we present existing work on service selection. In Section 3 we describe our modeling of the service selection problem. The solution that we propose for its resolution is given in Section 4, an application case is studied in Section 5 and we conclude in Section 6.

2 Related work

We considered two separate tasks to perform for binding an abstract WSC to concrete services. The first task consists of finding adequate concrete operations for abstract ones. Our paper does not focus on it. However, interesting work on the topic have been proposed by Lee [2] and Burstein et al. [5].

Given a realizable WSC, we are interested in finding an optimal binding in order to optimize the global service

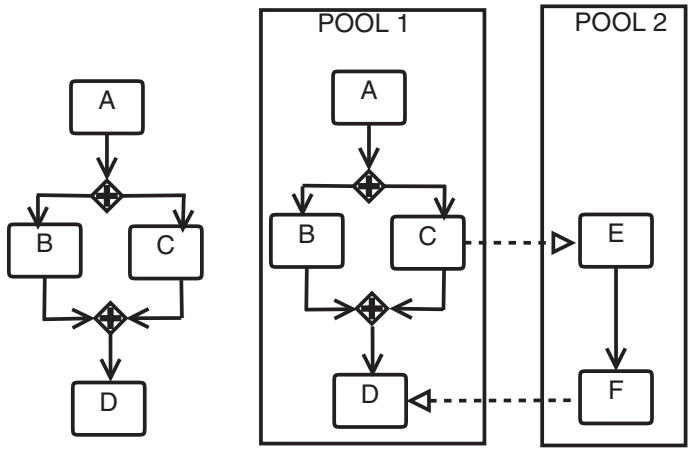

Figure 1 Example of services compositions. In the panel **a)**, we have a single process and in **b)** a multi-process. *A*, *B*, *C*, *D*, *E* and *F* correspond to WSs operations. The notations are based on the BPMN collaboration diagram.

composition. This problem has been addressed in various work with ILP.

One of the precursor works based on ILP was done by Lee [2]. This work innovated by showing that even with just two QoS parameters (price, SRT), the service selection problem can be reduced to the multi-choice Knapsack problem that is NP-hard. Consequently, polynomial time algorithms should not be expected for this problem, unless P = NP. In considering a more general setting, with a finite number of QoS factors that can be aggregated linearly, an extended formulation of the Lee model was proposed by Yu et al. [11]. Their work also showed that the service selection problem can be reduced to the multidimensional multi-choice Knapsack problem. Let us recall that this confirms the NP-completeness of the problem. Zeng et al. [12], proposed an ILP for service selection on: price, duration, reputation, reliability and availability. The main interest of this work is to state how to linearize constraints related to availability, expressed in an exponential form in aggregation rules. A similar ILP was proposed by Ardagna et al. [15]. Moreover, the authors noticed that the global model was not always feasible. They then proposed a global negotiation algorithm for having a feasible service selection solution. Our work will highlight the benefit of such an algorithm.

ILPs provide an exact solution; but, they are not always efficient when dealing with large problem sizes. This motivated the study of the service selection problem with heuristics.

Zeng et al. [12] proposed an algorithm that, for each service, locally performs service selection optimized for the penalty. As they pointed out, this solution can be suboptimal; however it is a good option for obtaining fast results for the service selection problem. Yu et al. [11] showed how to use the relaxation of their ILP for building a branch and bound algorithm on service selection. The runtime of the resulting algorithm however can be exponential. A similar but faster algorithm was described by Alrifai et al. [7]. Ben Mokhtar et al. [1] proposed a two-phases heuristic for service selection. In the first phase, the heuristic classifies the concrete services regarding their local penalty. The classification is then used for guiding the selection process in the second phase. The advantage of this approach is to propose near-exact solutions for the service selection problem. Finally, genetic programming approaches were also used for solving the service selection problem [8,10,16]. Let us remark that from the service selection problem we can easily derive the gene representation that consists of a string where each character is a service operation. One downside of genetic algorithms is that additional parameters such as the population size need to be tuned.

At this point, it is clear that there are works on service selection. However, as already said in the introduction, our work considers a specific case of WSCs based on multiple collaborating processes. Moreover, we focus on SRT and EC as QoS dimensions. To the best of our knowledge, there is no previous work in this context. In the CHOReOS project we are involved with, near-exact heuristics and genetic programming are desirable for the service selection in order to deal with ultra-large WSCs. In this work however, we propose to use MILP as a preliminary solution. MILP are adequate for having an exact bound that can further be used for assessing the quality of future heuristics proposed on the question.

The next section is devoted to the description of the computational setting in which we study the service selection.

3 Model

For modeling WSCs, we use the hierarchical service graph (HSG) model proposed by Goldman et al. [14].

As shown in Figure 2, an HSG is a graph with three layers, each encapsulating a particular abstraction of a service composition; these are the business processes, the WSs and the machines layers.

The business processes layer, which we will also refer to as the **operations graph**, presents the logic of the service composition in the form of several communicating business processes. We specify this layer by the means of the BPMN graph for collaboration processes. As illustrated by Figure 2b, an operations graph is made of operations (as e.g. $A-D$ in Figure 2b), interconnected within business process (we will also use the term pool) by the means of BPMN connectors (e.g. AND split connector in Figure 2b); between the pools, there are messages exchanges (e.g, between C and E).

At the upper layer of an HSG, there are interactions between abstract operations; the behavior of these operations is implemented in the WSs of the second layer. In the general case, any abstract operation can be implemented within various WSs; moreover, each WS can exist in multiple instances, deployed on various machines [17]. In our study however, we will assume that any WS only exist in one instance, deployed on a single machine.

The connectivity of an HSG is mainly captured by the operations graph (a subgraph of the HSG). In a formal manner, we describe it as a tuple $G_o = (P, O, C, E_{ocp}, E_{oc}, E_{oo})$ where: $P = \{P_1, \ldots, P_h\}$ is the set of pools (business processes of the HSG), O the set of WSs operations, and C the set of BPMN connectors (we consider the AND, XOR and OR connectors).

Here, E_{oc} describes precedence constraints between operations and connectors; E_{ocp} states for each operation and connector its pool and E_{oo} describes messages connections between operations of separated pools.

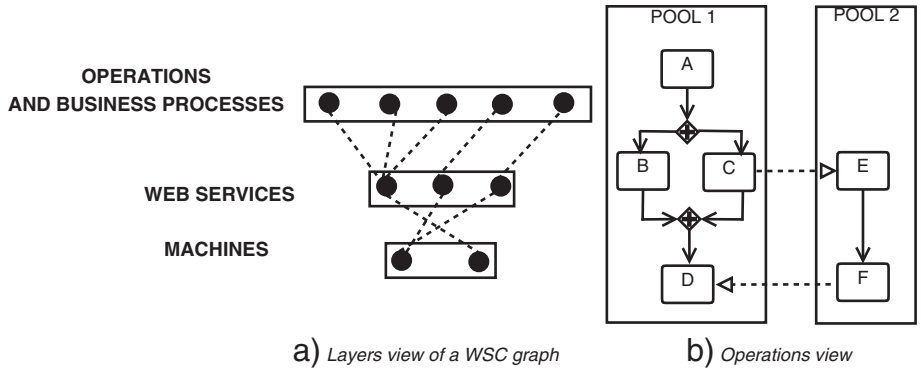

Figure 2 Views of a WSC.

Given an HSG, we will refer to the part of the operations graph that is contained in a pool as a pool operations graph. Any operation that can send data will be referred to as a sending operation; those that can receive will be referred to as receiving ones.

As stated above, the HSG upper layer is made of multiple communicating business processes. Since the BPMN language is vast, there are multiple possible structures for the operations graph. For the sake of simplicity, we will reduce the potential pool operations graph to the ones derived by composing the patterns of Figure 3. Let us notice that many related works did similar considerations [12,15]. In these patterns, each G_i refers either to a subgraph built from a composition of the patterns, or to a single operation. $G_1*, \ldots G_n*$ refer to subgraphs of G_o, having a unique node without predecessors and a unique one without successors. Finally, let us remark that in the proposed patterns, for each subgraph having a split

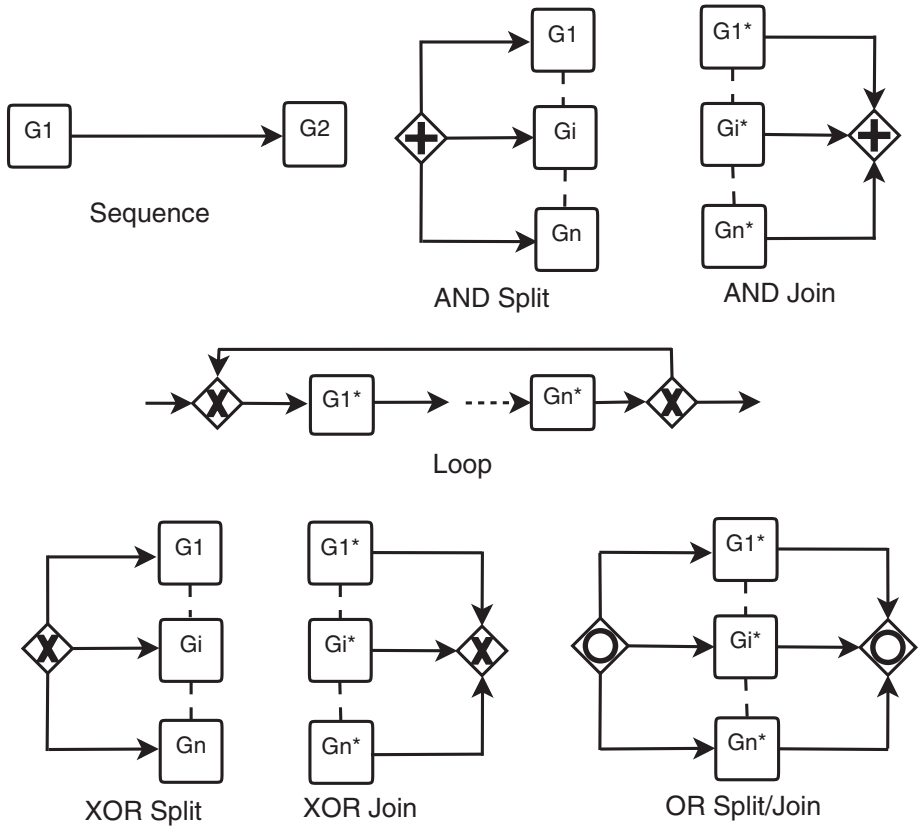

Figure 3 Set of generic subgraph patterns in the operations layer of an HSG. Each G_i is again a subgraph obtained from these patterns or an operation.

OR connector, there must be a corresponding join OR connector.

We described the HSG model that we will use for WSC representation. In the next section, we present our modeling of the service selection problem.

3.1 The service selection problem with HSG

In HSGs, abstract operations of the upper layer are associated with WSs implementations. We will use the term concrete operations for referring to these implementations. In the service selection problem the objective is to look for the best binding to operate between abstract and concrete operations in order to optimize the quality of the service composition. Below, we provide a formal description of the problem.

3.1.1 Problem inputs

Given an HSG, we have the set O of its operations. For each operation $u \in O$, there is a set of concrete implementations $Co(u) = \{u_1, \dots u_{m_u}\}$. For each concrete implementation u_v, we have the mean SRT $S(u_v)$ and the energy consumption $E(u_v)$. Finally, we have two positive upper bounds $MaxS$ and $MaxE$ on the service response time and a weighted value $w \in [0, 1]$ tuned in the middleware for giving more priority to the SRT or the EC in the problem optimization goal.

3.1.2 Problem objective

We are looking for an assignment of concrete operations for O that fulfills the following constraints:

- C_1: each operation must be associated with a unique concrete implementation;
- C_2: the QoS of the resulting composition must not exceed $MaxS$ on SRT and $MaxE$ on EC;
- C_3: if S is the SRT of the resulting composition and E its energy consumption, then the assignment must minimize the global penalty $w.S + (1 - w)E$.

This formulation is a classical one on service selection [1,2]. The constraint C_2 is used to include SLAs defined by the user on SRT and EC. C_3 defines the global penalty function to be optimized in the middleware perspective.

In our problem formulation, we assume that if we bind each operation with a concrete implementation, then we can compute the mean SRT and EC of the resulting WSC. In the following, we explain how we intend to compute these means by providing an execution semantics for HSGs.

3.2 Execution semantics of HSG

Our execution semantics is based on the idea that it is possible to have a good approximation of the mean value capturing the SRT and EC led by computations and communication on concrete operations. These estimates are then used throughout aggregation rules to infer the SRT and EC of WSCs. This section has two parts. In the first, we present the aggregation rules that we use. In the second, we discuss the computation of a good approximation for the mean SRT and EC.

3.2.1 Aggregation rules for SRT and EC in a pool

The objective of these rules is to state how to infer the SRT and the EC from the generic subgraphs of Figure 3. For the SRT, we will use the aggregation rules proposed in our previous work [14,17]. In our context, we did not find a previous work on aggregation rules for EC. However, we propose to deduce them (as previously done for SRT) from a mean analysis of the possible routing occurrences, in an HSG where the upper layer is reduced to a subgraph pattern. In applying this analysis for energy consumption, we obtain the aggregation rules of Table 1. Here, $E(G_i)$ denotes the SRT that can be expected from a subgraph G_i. On a generic XOR split graph, we assume a probability p_i for a request to be routed towards one subgraph $G_1, \dots, G_n$. For a loop subgraph, we assume that there is a maximal number of loops n_l and a probability p_{li} to loop i times. Finally, for an OR split/join, we assume for simplification that the request can only be routed to the subgraph G_1, G_2 or simultaneously, to both. Each routing occurrence, has a known probability p_{or1}, p_{or2} and $p_{or||}$.

The probability values that we consider are the same as the ones used for SRT aggregation as stated in our previous work [14]. For computing these values a training stage where the composition behavior is observed on multiple requests might be necessary. This is not however within the scope of our paper. Aggregation rules for EC differ from the SRT cases on AND and OR subgraphs. Indeed, the SRT for a request that traverses an AND subgraph is the maximal one obtained from the execution of all branches. However since all branches participate in the computation, the EC is obtained by accumulating the local consumptions.

The given rules state how to aggregate SRT and EC in a pool operations graph. For a complete aggregation, communication must be included. This is done in the following.

Table 1 Energy consumption aggregation on subgraphs patterns

Sequence	**AND split**	**AND Join**
$E(G_1) + E(G_2)$	$E(G_1) + \dots + E(G_n)$	$E(G_1^*) + \dots + E(G_n^*)$
XOR Split	**XOR Join**	**LOOP**
$\sum_{i=1}^{n} p_i.E(G_i)$	$\sum_{i=1}^{n} p_i.E(G_i^*)$	$\sum_{i=1}^{n_l} p_{li}.\sum E(G_i^*)$
OR Split/Join	$p_{or1}.E(G_1) + p_{or2}.E(G_2) + p_{or\|\|}.(E(G_1) + E(G_2))$	

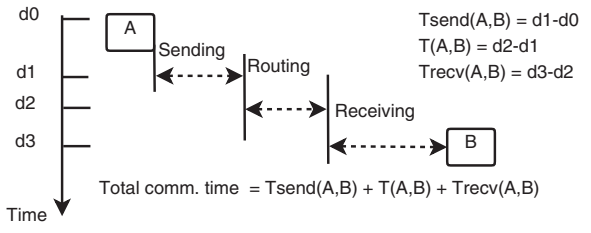

Figure 4 Exchange of information between *A* and *B*, starting at the date d_0 and ending at d_3.

3.2.2 SRT and EC aggregation for communication

For communication, we propose to decompose any data transfer between two operations A and B in three parts: a sending part where the sending operation is busy, a transit part where the data are routed and a receiving part, where the receiving operation is busy. Figure 4 presents this decomposition. For each of these parts, we have a mean duration time.

This decomposition is the one used in the port model [18]. Since this model has been proved efficient in many practical settings, our assumption on mean SRT for communication holds.

Finally, for estimating the EC for data transport, we refer to the work by Baliga et al. [19]. In their study, they estimate the EC required for sending a bit from a computer to a private or public data center. Their modeling does not contradict the port model. The main idea behind is that the resulting energy can be deduced by aggregating the power consumption of the switches that are part of the communication. Based on this result, we can assume that for any sending between A and B, we have an estimation of the resulting EC: $E_{A,B}$.

The aggregations rules can be used for computing the SRT and EC of a composition only if it is reasonable to expect a mean SRT and EC at the operation level. This assumption will be discussed in what follows.

3.2.3 Estimates of SRT and EC for computation

The SRT of a concrete operation is the mean time for which it returns a result. For computation we believe that such an estimation is possible if: (1) there is a low variation on the number of FLOPS performed in concrete operation runs; (2) there is a low variation on the mean frequency at which the computer performing the operation runs. With these two assumptions, a good approximation of the SRT can be computed from the mean number of operations performed by each machine in a time unit.

We would like to stress that for a precise estimation, it might be preferable to consider that the runtime of concrete operations are input sensitive. This has implications on modeling the mean time of a concrete operation as a function of the input data size that it processes. However, we did not make these considerations here for the sake of simplicity. Other works also use similar SRT modeling [12,17,20].

For the EC of a WS operation, we adopt the definition of Bartalos et al. [21]. The EC of an operation is the total power consumed by hardware resources. We formalize this definition as follows. In our HSG model, u is executed on a unique machine $m(u)$. If at each time instant t during this run, a power $P_{m(u),u}(t)$ is consumed, then the energy consumption of this operation is $E_u(t) = \int_t^{t+t_u} P_{m(u),u}(t).dt$. In the same vein of our assumption (for the SRT case) of a low variation on machine frequency and number of FLOPS, we also consider that we have a low variation on the power $P_{m(u),u}(t)$ between distinct time instants. Therefore, the EC caused by o_u will be $E_u = P_{m(u),u}(0).t_u$. Here, t_u is the mean response time of the operation o_u.

At this stage, the service selection problem that we are addressing is clarified. In the next section, we will analyze this further.

3.3 Analysis of the service selection problem

Lee [2] established the NP-hardness of the service selection problem. The NP-hardness proof considers the service selection problem on a sequence of operations. However, we believe that by relaxing some constraints, the problem could be simpler. The aim of this section is to propose an analysis of the service selection in the specific case where $MaxS = MaxE = +\infty$. We will refer to this relaxed version as the SLAs free service selection problem. The SLA free problem has a practical implication on the negotiation stage when building a service composition. At the beginning, a user who wants to compose services might not have an idea of the SRT and EC that he expects. In such a context, we can globally minimize the penalty by deferring to the middleware to choose the best service composition. Now we present some results for the SLA free problem.

Property 1. In the case where we have (1) a sequence of operations; (2) an XOR split tree whose branches comprise sequence of operations; (3) a loop on a sequence of operations; the optimal solution for the SLA free problem can be obtained in polynomial time in the maximal number of abstract and concrete operations.

Proof. Let us consider indeed a sequence of l abstract operations $o^1, \ldots o^l$. If for any operation o^i, we chose a concrete one o^i_{is} then the total sequence penalty is $\sum_{i=1}^{l}[w.S(o^i_{is}) + (1-w).E(o^i_{is})]$. This is the sum of the penalties of chosen operations. Therefore, the SLA free problem here consists of choosing a set of services such as to minimize the sum of their penalties. Since the penalty is always positive, we can then obtain an optimal solution by local optimization of the penalty. The same construction

can be applied in the case of XOR trees and loops made of sequences. Finally if m is the maximal number of concrete services for each $o^i, i \in \{1,\ldots,l\}$, then the described process can be performed in O($m.l$). □

An interesting question then is whether or not the local optimization approach used in the previous proof gives an optimal solution on any type of process graphs. The answer is no.

Property 2. If the SRT and EC values can be drawn from any arbitrary distribution, then the local optimization approach on the SLA free problem is not always optimal if we have AND split patterns.

For instance, let us consider an AND split tree with two branches. The first tree branch has an operation u and the second an operation v. Here, $w = 0.1$ and each operation can be associated to two concrete ones according to the SRT and EC given in Table 2.

While local optimization will bind u with u_1 and v with v_1 (resulting penalty: $(0.1 * \max\{5,8\} + (12 + 12.6) * 0.9 = 22.94$), the best solution consists of binding u with u_2 and v with v_1 (resulting penalty: $0.1 * \max\{8,8\} + (11.8 + 12.6) * 0.9 = 22.76$).

Our proposed counter-example exploits the fact that it is only after choosing an assignment on all branches that we can deduce the final resulting SRT. Implicitly, this suggests that an exploration of possible solutions that consider all possible SRT values might give optimal results. Let us for example consider the specific case of a sequence of *elementary* Fork/Join. Each fork here comprises at most D branches made of one operation. Let us assume that the operations SRT are all positive integers and that each operation can be associated with at most m concrete ones. Let us also assume that $S^+ = \sum_{u \in O} \max_{x \in Co(u)} S(x)$. We have the following result.

Theorem 1. *Given an elementary Fork/join sequence. If D is the maximal number of branches to which a Fork can lead to, then the optimal solution for the SLA free problem can be computed in $O(h.D.(S^+)^2.(m \log m))$ where h is the depth of the sequence.*

Table 2 Counter-example on the optimality of local optimization

(a) Costs on u			
SRT	EC	Penalty	Concrete operation
5	12	11.3	u_1
8	11.8	11.42	u_2
(b) Costs on v			
SRT	EC	Penalty	Concrete operation
8	12.6	12.14	v_1
9	12.7	12.33	v_2

Proof. For elementary Fork/join sequences, we propose to make a bi-dimensional exploration on SRT and depth. Our exploration is also based on dynamic programming and work by exploring first the SRT dimension. Any point (e,f) is a partial assignment made for all services that are under the depth e and that leads to a global SRT equal to f. We associate each point (e,f) with an EC denoted $d(e,f)$. $d(e,f)$ is the minimal EC that can be obtained from any assignment for operations at depths $1,\ldots e-1$ and whose SRT is f.

The computation of this weight obeys to a sub-optimality rule. Let $Z(e',f')$ be the minimal cumulated EC obtained from an assignment made on abstract operations of the depth v' and whose maximal SRT is e'. Then, we have the following equation:

$$d(e,f) = \min_{f' \in \{0,\ldots f-1\}} \{d(e-1,f') + Z(e,f-f')\}.$$

The computation of $Z(e',f')$ can be done as follows. The operations whose SRT exceeds f' are firstly eliminated from all set of concrete services at the depth e'. If this leads to an empty set or if there is not an operation with SRT equal to f' then we return $Z(e',f') = +\infty$. Otherwise, we sort each set $Co(u)$ on the EC and we return for each operation, the minimal EC while ensuring that at least one operation of a branch has an SRT equal to f'. The sum of these minimal ECs is returned in $Z(e',f')$. Naturally, at each point (e,f), we have two cases: either an assignment is possible and then we keep the one leading to the minimization of $d(e,f)$ at depth f, or there is not a possible assignment (we only have infinite values when running Z). Then, there will not be any assignment associated with this point.

When we end the computations of values $d(h,f), f = 0,\ldots,S^+$, we then compute the possible penalty costs at height h. These can be obtained from the following relation: $u(h,f) = a.f + (1-a).d(h,f)$. We return as optimal solution the one that leads to the minimal penalty at depth h.

The proposed Bellman equation assumes that we can decompose the optimal solution on a Fork/Join sequence into a set of solutions optimized for EC for any pair $(SRT, depth)$. Such a characterization holds on Fork/Join sequences. □

This dynamic programming can be used with local optimization to obtain an optimal solution on other cases such as an elementary AND split combined with a sequence, nested AND tree etc.

We showed that fast solutions can be obtained on some cases of the SLA free problem. If however we include SLAs, even on sequence case, we already have an NP-hard problem to solve. Next, we will consider a more general approach for the resolution of the service selection problem in HSGs.

4 Solving the service selection problem

Our resolution of the service selection problem uses our prior algorithm proposed for QoS prediction [14]. Let us refer to it as **SRT_LP_gen**. Given an operations graph $G_o = (P, O, C, E_{ocp}, E_{oc}, E_{oo})$ where we already have a concrete service associated with each abstract one, **SRT_LP_gen** states how to generate a linear program that when solved will return the SRT of the resulting WSC. For solving the selection problem, we propose to modify **SRT_LP_gen** in order to: (1) introduce the choice of concrete services; (2) compute the EC of the composition; (3) introduce constraints related to SLAs and penalty. In doing so, instead of a Linear Program, we will obtain an MILP. Below, we give a description of these different stages.

4.1 Introducing a choice among concrete operations

For each operation $u \in O$, we associate a bi-dimensional $0-1$ variable $y_{u,j}$ such that

$$y_{u,j} = \begin{cases} 1 \text{ if the concrete service } u_j \in Co(u) \text{ is bound with } u \\ 0 \text{ otherwise.} \end{cases}$$

In **SRT_LP_gen**, any operation u has an SRT value defined as a constant $T(u)$. For the binding purpose, we will change it into a variable. We also introduce a variable $D(u)$ giving the EC of the operation u. On these variables, we propose to generate the following equations:

$$T(u) = \sum_{j=1}^{|Co(u)|} y_{u,j}.S(u_j) \qquad \forall u \qquad (1)$$

$$D(u) = \sum_{j=1}^{|Co(u)|} y_{u,j}.E(u_j) \qquad \forall u \qquad (2)$$

$$y_{u,j} \in \{0,1\} \qquad \forall u, j \qquad (3)$$

$$\sum_{j=1}^{|Co(u)|} y_{u,j} = 1 \qquad \forall u \qquad (4)$$

Let us remark that in defining $T(u)$ as variables, we still keep linear equations on SRT constraints. For each abstract operation u, the equations $1-4$ set in the variables $T(u)$ and $D(u)$, the SRT and EC that they will have, depending on the concrete service to which they are bound (the service u_j for which $y_{u,j} = 1$).

4.2 Computing the EC of a composition

We associate each operation $x \in O$ with a real variable e_x. **SRT_LP_gen** successively explores the sets E_{oc}, and E_{oo}. During this exploration, multiple interpretations are associated with every arc in order to generate the mean SRT. These interpretations will lead to the generation of an equation that we denote Eq. Our goal is to extend **SRT_LP_gen** to generate EC constraints. For this, we revisit the different arc interpretations in

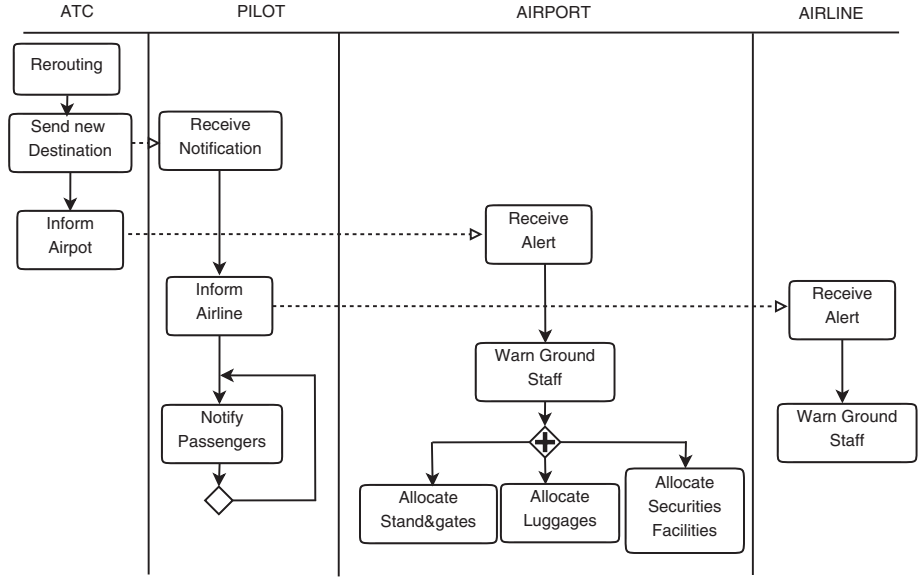

Figure 5 Crisis management in airport transportation.

SRT_LP_gen and include equations related to the EC. For any arc $(u, v) \in E_{oc}$ during the exploration, we thus have the following additional equations that we generate:

- C_1: [u does not have a predecessor]: $Eq \longleftarrow e_u \geq D(u)$;
- C_2: $[u, v \in O] : Eq \longleftarrow e_v \geq e_u + p_a[(u, v)] .D(v)$; $[u \in O \cup C, v \in C]$
- C_3: If v is an XOR join that closes a loop (there is $(v, s) \in E_{oc}$ and a path $(s, y_1), \ldots, (y_n, u)$ of E_{oc}), then $Eq \longleftarrow e_v \geq \sum_{i=1}^{nl[v]} p_{li}(e_u - e_s) + e_s$; Here, $nl[v]$ is the maximal index of looping stage;
- C_4: If v is a split connector then $Eq \longleftarrow e_v \geq e_u$; $[u \in C, v \in O \cup C]$
- C_5: If u is a join connector then $Eq \longleftarrow e_v \geq e_u$;
- C_6: If u is a split then $Eq \longleftarrow e_v \geq p_r[(u, v)] .e_u + p_a[(u, v)] .D(v)$; [$v$ is a join connector whose predecessors are $u_1, \ldots, u_n$]
- C_7: If v is an AND JOIN then $Eq \longleftarrow e_v \geq \sum_{i=1}^{n} e_{u_i}$;
- C_8: If v is an XOR JOIN but does not close a loop then $Eq \longleftarrow e_v \geq \sum_{i=1}^{n} e_{u_i}$;
- C_9: If v is a OR JOIN then $Eq \longleftarrow e_v \geq \sum_{i=1}^{2} e_{u_i} + p_{or||}(v).em(u_1, u_2)$ and $Eq \longleftarrow em(u_1, u_2) \geq \frac{e_{u_1}}{p_r[(u_1,v)]} + \frac{e_{u_1}}{p_r[(u_1,v)]}$;

$pr_a[(u, v)]$ and $p_a[(u, v)]$ are conditional and reachability probabilities. They are defined by Goldman et al. [14]. These constraints differ from the SRT ones in the interpretation of parallelism. While for SRT, we must take the SRT of the longest path, for EC, we must take the sum from all paths. Using the same philosophy for interpreting parallelism, we can easily extend on EC the constraints on SRT defined in **SRT_LP_gen** for aggregating EC in the resulting WSC.

4.3 SLAs and penalty constraints

With the proposed modifications, this stage we will have at two variables: ZS that comprises the SRT of the service composition and ZE that comprises its EC. We then add the following equations for computing SLAs and penalty:

$$ZS \leq MaxS \tag{5}$$

$$ZE \leq MaxE \tag{6}$$

$$Z = w.ZS + (1 - w).ZE \tag{7}$$

Finally, we set that the objective function of the MILP is the minimization of the variable Z.

4.4 Analysis

It might seem that our algorithm generates too many variables or constraints. However, this number is polynomial as stated by the following result.

Theorem 2. *Given a graph $G_o = (P, O, C, E_{ocp}, E_{oc}, E_{oo})$, the generated MILP has at most $O(n.m)$ variables and $O(n.m)$ equations where $n = |O \cup C|$ and $m = \max_{u \in O} |Co(u)|$.*

Proof. The execution of **SRT_LP_gen** in our approach will generate at most O(n) variables and O(n) equations [14]. By adding equations related to EC, we keep the same order of complexity. However, variables and constraints related to the choice of a concrete service will lead to O($n.m$) and equations. □

Given the value *MaxS* and *MaxE*, the MILP might not find any solution because the problem is infeasible. In that case, it might be interesting to assist the user in the generation of a good compromise. We propose to approximate the intervals of SRT and EC values with which the problem is feasible.

For determining the interval $[S_{min}, \ldots, S_{max}]$ of *feasible values* for MaxS, we use two policies: the *SRT_Min_First* policy and the *SRT_Max_First*. In *SRT_Min_First*, we run the MILP with the variables $y_{u,j}$ tuned as follows:

$$y_{u,j} = \begin{cases} 1 \text{ if the concrete operation } u_j \text{ has the minimal SRT on } u \\ 0 \text{ otherwise} \end{cases}$$

The result that contains ZS after this run is S_{min}. In the *SRT_Max_First*, we proceed in the same way; however, we choose the concrete operations with a maximal SRT preference. At the end, we have an approximation of S_{max}. In the same way, we can develop *EC_Min_First* and *EC_Max_First* policies for computing the interval of possible EC values.

Finally, a user might want to know what is the minimal EC that he can expect given a maximal SRT value. In such cases, we propose to remove the equation $ZE \leq MaxE$ in the MILP, to set $w = 0$ and then run the MILP. ZE will return the minimal EC that can be expected for this SRT value. A similar transformation can be applied for

Table 3 Experimental settings

#Series	1	2	3	4	5	6	7	8	9	10
MaxE	1600	1600	1700	1700	1800	1800	1900	1900	2000	2000
MaxS	1600	1650	1650	1700	1700	1750	1750	1800	1800	1850

Given a series (e.g. 1) and a maximal number of concrete services (e.g. 20), we performed 100 experiments. We did not change the value of w in the experiments. We always have $w = 0.5$.

choosing the minimal SRT that can be expected given a maximal EC.

5 Application

The CHOReOS project is interested in optimizing the interactions among various processes collaborating on airport transportation. A collaboration use case is the crisis management when facing bad weather condition during a flight. This use case involves an air traffic control (ATC), a pilot (more precisely an internal plane system), an airport and an airline. We can describe the collaboration as follows. Due to bad weather conditions, the ATC decides to cancel all flights. It checks for scheduled flights, reroutes them and notifies all concerned pilots. The pilots inform their airlines and passengers. The ATC also informs the airport at which the flight is rerouted about its decision. This airport then prepares its ground staff for the new situation.

A complete description of the use case has been done by Chatel et al. [22]. In our work, we consider only the sub-view presented in Figure 5.

There are multiple possible Web implementations of the process given in Figure 5. In particular, we can either use a sub-composition for implementing an activity or a WS operation. In our experiment we will consider the latter option for all activities.

5.1 Experimental setting

For the crisis management example, we performed 10 series of 1000 experiments. These experiments had two objectives: the first was to show that the proposed MILP approach has a real qualitative advantage over local optimization. The second was to show that although we use MILP, reasonable time can be expected in practice.

Each series is determined by the parameters *MaxE* and *MaxS* used for SLAs. The ones that we used are set in Table 3. For each experiment, we created multiple concrete operations to be associated with abstract ones. The number of concrete operations belongs to the set {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. For any experiment, we always chose the same number of concrete services for all abstract operations. This means that in any experiment, we have $\max_{u \in O} |Co(u)| = \min_{u \in O} |Co(u)|, \forall u$.

The concrete operations SRT values are drawn from the uniform distribution within [100, 1000] (in ms). Assuming that an operation has an SRT equal to S and leads to a mean power consumption equal to P during its execution, we used the formula $E = P.S$ for computing its EC. We draw P (in watt) from the uniform distribution in the interval [100, 200]. Finally, we chose the number of loops for passenger notification from the uniform distribution between 1 and 10, the communication times between 10 and 40 (ms) and we normalized E in order to have a metric in watt-second.

We performed our experiments on an Intel Core i7 processor, 2.7 Ghz, 8 GB. The implementations were done in C language with the GLPK solver [23].

5.2 Experimental results

In Figure 6, we show the mean aggregated penalties obtained from our experiments with our MILP approach and the *EC_Min_First* and *SRT_Min_First* policies. The aggregates are made on the number of concrete operations used in the experiment. These results are the optimal ones that the GLPK solver computed in each case. The results describe an increase of the penalty on the maximal number of concrete operations. This is due to the fact that in increasing the number of concrete services, we increase the diversity (or the standard deviation) in SRT and EC per operations. We also did a comparison of the penalties between the local optimization policies and MILP. The results are reported in Table 4 and confirm the

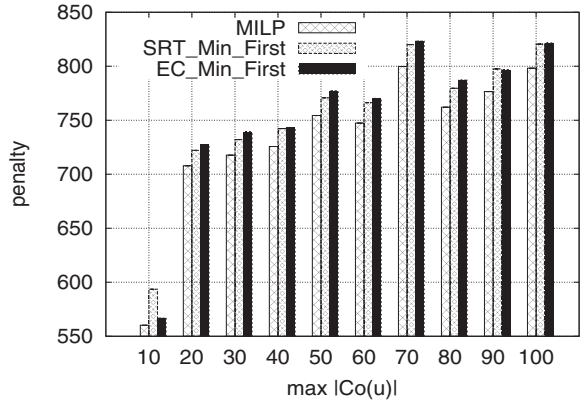

Figure 6 Mean penalty per approaches depending on the maximal number of concrete operations.

Table 4 Ratio of mean aggregated penalties between the local approaches and the MILP

$\max_{u \in O} \lvert Co(u) \rvert$	SRT_Min_First	EC_Min_First
10	1.059	1.011
20	1.020	1.027
30	1.022	1.030
40	1.021	1.024
50	1.025	1.029
60	1.025	1.030
70	1.022	1.029
80	1.026	1.032
90	1.027	1.025
100	1.028	1.029

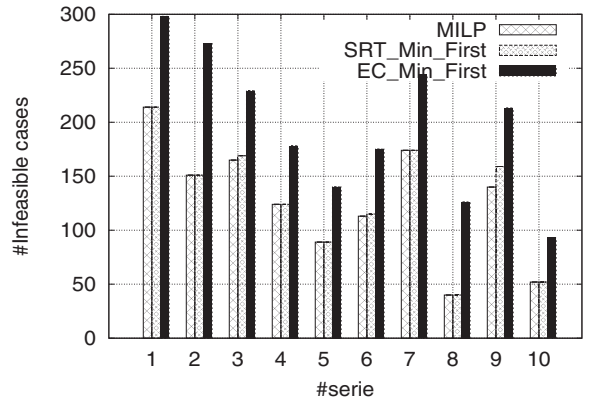

Figure 7 Number of infeasible cases per approaches.

superiority of MILP over local optimization as suggested by our previous analysis.

For the sake of fairness, we aggregated the penalties values in the experiments only when there were no approaches leading to an infeasible solution. In many cases indeed, the problem was infeasible. For each approach, Figure 7 reports the number of infeasible solutions that were detected. These results globally confirm the need to establish a negotiation procedure with the users for preventing infeasible problems. As shown by the results, it does not suffice to increase the maximal service response time or the energy consumption to have less infeasible cases. This number depends on the combination of these two parameters but also on the QoS of the concrete services. To conclude on infeasibility, we compared MILP and local optimization. The results presented in the Table 5 state that in all cases where a solution was found by local optimization, we found one with the MILP.

Finally in Table 6, we present the MILP runtime. The results are encouraging they show that despite

Table 5 Ratio of the number of infeasible cases between the local approaches and the MILP

#series	SRT_Min_First	EC_Min_First
1	1.000	1.392
2	1.000	1.807
3	1.024	1.387
4	1.000	1.435
5	1.000	1.573
6	1.017	1.548
7	1.000	1.402
8	1.000	3.15
9	1.136	1.521
10	1.000	1.788

Table 6 MILP Runtime (in 10^{-2} seconds)

#Series	1	2	3	4	5	6	7	8	9	10
Runtime	5.0	5.2	6.3	6.8	6.7	8.0	10.5	8.6	9.1	8.9

the NP-hardness of the service selection problem, we can expect to solve it in a reasonable runtime even with more than 1300 concrete operations. Let us however notice that we do not expect such results if we increase both the number of abstract and concrete services.

6 Conclusion

In this paper, we proposed new solutions for optimizing the selection of services on service response time and energy consumption on Web service compositions. We proposed two cases in the optimization: the SLAs free case where there is no constraint on the maximal service response time and energy consumption and the general case where such constraints are included. In the SLA free case, we showed that in some settings, the problem can be solved by the means of local optimization and dynamic programming. In the general case, we proposed an algorithm that generates an MILP solving the problem. We tested our solutions with multiple simulations that have globally shown that our MILP gives better results than local optimization.

We have multiple opportunities for continuing this work. The first is to extend the MILP generation on other QoS parameters such as the availability and the reputation. The main challenge will consist of translating the aggregation rules into adequate equations. Our second opportunity is to develop a global solution including negotiation for the SLAs. As our experiments showed, indeed there are many cases where we have infeasible problems. A negotiation stage might also be considered for another feature that we did not include: it is possible to re-select services in the case where our estimates of the service response time and energy consumption are wrong. Our third opportunity is to reconsider the modeling of the penalty function. One weak point of our formulation is that we aggregate values of different units (for instance ms and watt). An investigation of the best formulation to adopt certainly promising.

Finally, let us remark that for cloud architectures, we need to take into account virtualization. This implies modifications of the notion of HSG in other to add a virtualization layer. An immediate consequence of this modeling is that we must include potential QoS fluctuation due to the migration of virtual machines. For this, we envision two tasks to perform. One is to make a sensitivity analysis of our proposal. The other is to extend our approach to include this dynamicity in the modeling of SRT and EC.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
YN, AG and DM proposed an MILP-based algorithm for the service selection problem in web services compositions, implementing multiple business processes. All authors read and approved the final manuscript.

Acknowledgements
This research was funded by HP Brasil under the Baile Project and from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS-Large Scale Choreographies for the Future Internet). Yanik Ngoko was partially supported by the FAPESP foundation of the State of São Paulo.

Author details
[1]Laboratoire d'Informatique de Paris Nord, Villetaneuse, France. [2]IME-USP, São Paulo, Brasil. [3]HP Labs, Palo Alto, USA.

Received: 6 May 2013 Accepted: 25 October 2013
Published: 26 November 2013

References

1. Ben Mokhtar S, Kaul A, Georgantas N, Issarny V (2006) Efficient semantic service discovery in pervasive computing environments In: Proceedings of the ACM/IFIP/USENIX 2006 international conference on Middleware, Middleware '06. Springer-Verlag New York, Inc., Melbourne, pp 240–259
2. Lee J (2003) Matching algorithms for composing business process solutions with web services. In: Bauknecht K, Tjoa AM, Quirchmayr G (eds) Proceedings of the 4th international conference on E-Commerce and web technologies, Lecture Notes in Computer Science. Springer, Prague, pp 393–402
3. OWL-S: Semantic Markup for Web Services. www.w3.org/Submission/OWL-S
4. Web Service Semantics – WSDL-S, Technical Note. lsdis.cs.uga.edu/projects/meteor-s/wsdl-s
5. Burstein MH, Hobbs JR, Lassila O, Martin D, McDermott DV, McIlraith SA, Narayanan S, Paolucci M, Payne TR, Sycara KP (2002) Daml-s: Web service description for the semantic web In: Proceedings of the first international semantic web conference on the semantic web, ISWC '02. Springer-Verlag, London, pp 348–363
6. The choreos project. www.choreos.eu
7. Alrifai M, Risse T, Dolog P, Nejdl W (2009) A scalable approach for qos-based web service selection. In: Feuerlicht G, Lamersdorf W (eds) Service-oriented computing — ICSOC 2008 workshops. Springer-Verlag, Berlin, Heidelberg, pp 190–199
8. Canfora G, Di Penta M, Esposito R, Villani ML (2005) An approach for qos-aware service composition based on genetic algorithms In: Proceedings of the 2005 conference on genetic and evolutionary computation, GECCO '05. ACM, Washington DC, pp 1069–1075
9. Issarny V, Georgantas N, Hachem S, Zarras A, Vassiliadis P, Autili M, Gerosa MA, Hamida AB (2011) Service-oriented middleware for the future internet: state of the art and research directions. J Internet Serv Appl 2(1): 23–45
10. Jaeger M, Rojec-Goldmann G, Muhl G (2004) Qos aggregation for web service composition using workflow patterns In: Proceeding EDOC '04 Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International. IEEE Computer Society, Washington, DC, USA, pp 149–159
11. Yu T, Zhang Y, Lin KJ (2007) Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web 1(1). doi:10.1145/1232722.1232728
12. Zeng L, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H (2004) Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30(5): 311–327
13. Weske M (2007) Business process management: concepts, languages, architectures. Springer
14. Goldman A, Ngoko Y, Milojicic D (2012) An analytical approach for predicting qos of web services choreographies In: Proceedings of the 10th international workshop on Middleware for grids, clouds and e-Science, MGC '12. ACM, Montreal, Canada, pp 4:1–4:6
15. Ardagna D, Pernici B (2007) Adaptive service composition in flexible processes. IEEE Trans Softw Eng 33(6): 369–384
16. Cao L, Li M, Cao J (2007) Using genetic algorithm to implement cost-driven web service selection. Multiagent Grid Syst 3(1): 9–17
17. Goldman A, Ngoko Y (2012) On graph reduction for qos prediction of very large web service compositions In: International conference on service oriented computing (SCC). IEEE Press, Hawai, pp 258–265
18. Banikazemi M, Sampathkumar J, Prabhu S, Panda DK, Sadayappan P (1999) Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations In: Proceedings of the eighth Heterogeneous Computing Workshop, HCW '99. IEEE Computer Society, Washington, DC, pp 125–133
19. Baliga J, Ayre R, Hinton K, Tucker RS (2011) Green cloud computing: balancing energy in processing, storage, and transport. Proc IEEE 99(1): 149–167
20. Cardoso J, Miller J, Sheth A, Arnold J (2002) Modeling quality of service for workflows and web service processes. J Web Semantics 1: 281–308
21. Bartalos P, Blake MB (2012) Green Web Services: Modeling and Estimating Power Consumption of Web Services In: International Conference on Web Services. IEEE Computer Society, Honolulu, USA, pp 178–185
22. Châtel P, Léger A, Lockerbie J (2011) Choreos requirements and scenarios for the "passenger-friendly airport" (d6.1). http://hal.inria.fr/hal-00664313. Hal_id = hal-00664313
23. The GNU Linear Programming Kit. http://www.gnu.org/software/glpk/

doi:10.1186/1869-0238-4-19
Cite this article as: Ngoko *et al.*: Service selection in web service compositions optimizing energy consumption and service response time. *Journal of Internet Services and Applications* 2013 **4**:19.