

Heuristic Based Time-Aware Service Selection Approach

Ikbel Guidara, Nawal Guermouche, Tarak Chaari, Saïd Tazi, Mohamed Jmaiel

▶ **To cite this version:**

Ikbel Guidara, Nawal Guermouche, Tarak Chaari, Saïd Tazi, Mohamed Jmaiel. Heuristic Based Time-Aware Service Selection Approach. International Conference on Web Services, Jun 2015, New York, United States. 10.1109/ICWS.2015.19 . hal-01396981

HAL Id: hal-01396981
https://hal.science/hal-01396981

Submitted on 8 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Heuristic based Time-aware Service Selection Approach

Ikbel Guidara[1,2,3], Nawal Guermouche[1,2]
[1]CNRS, LAAS, 7 avenue du colonel Roche
F-31400 Toulouse, France
[2]Univ de Toulouse, UT1, INSA, LAAS
F-31400 Toulouse, France
{iguidara, nguermou}@laas.fr

Tarak Chaari[3], Said Tazi[1,2] and Mohamed Jmaiel[3]
[3]ReDCAD Laboratory, University of Sfax
National Engineering School of Sfax
B.P. 1173, 3038 Sfax, Tunisia
tarak.chaari@redcad.org, tazi@laas.fr
mohamed.jmaiel@enis.rnu.tn

***Abstract*—QoS-based service selection is one of the important requirements in Service Oriented Computing (SOC). A challenging task towards this purpose is the selection of the best combination of services that fulfils user's requirements while meeting quality of service (QoS) constraints. This challenge becomes more complex when dealing with time-dependent QoS values and temporal properties. Indeed, during the selection, mutual dependencies between the different temporal constraints may arise so that the selection of each service may influence or be influenced by the selection of other services. On other side, to find the best solution, all potential combinations must be compared. However, the number of these combinations may be very high, which can present a barrier for enabling effective service selection. In this paper, we present a heuristic based time-aware service selection approach to efficiently select a *close-to-optimal* combination of services. First, pruning techniques are adopted to reduce the search space. Second, a novel heuristic approach is proposed based on service clustering, constraints decomposition and local selection while considering both QoS and temporal constraints. Finally, experiments which confirm the feasibility and effectiveness of the proposed approach in terms of its timeliness and optimality, are conducted.**

***Index Terms*—Service selection; Heuristic; Time-dependent QoS; Constraints decomposition; Clustering; Pruning;**

I. Introduction

Service-Oriented Computing (SOC) has emerged as a promising concept to integrate and compose atomic services for fulfilling complex tasks which can be specified as abstract processes. One of the key requirements in service composition is to ensure that the selected combination of services meets global quality of service (QoS) constraints. During the selection process, candidate services are evaluated in terms of both functional and QoS properties. As a large number of services can have similar functionality to realize the awaited abstract tasks, a specific issue that emerges is which services should be selected to form the optimal solution meeting end-user's global QoS constraints.

QoS-aware service selection becomes even more challenging when dealing with time-dependent QoS values (i.e., the offered QoS can change during time) and temporal constraints. In fact, in real world scenarios, QoS values of candidate services are not static and can change over time. For instance, the cost of a service can be higher in business hours. Moreover, temporal properties and dependencies between services can be specified at the abstract business level. For example, the business designer can require that the duration between the start times of two tasks should be less than a specific threshold.

Considering temporal properties when selecting the best solution brings two specific problems. First, as candidate services can have different QoS with respect to the execution time, each service can be considered as offering more than one instance. Hence, the number of service combinations that have to be compared becomes larger and then, the selection process requires more time to find the best solution. The second issue is that when dealing with temporal properties, the selection of one service can influence or be influenced by the selection of another service. These dependencies make the selection problem heavily constrained and thus, more complex to resolve.

Several selection approaches have been proposed in the literature. Global selection which enumerates and compares all the possible solutions is proven to be NP-hard [1], [2]. This is not practical in real-world applications when a solution has to be selected in a reasonable time. Some works adopt heuristic strategies to enhance the performance of the selection process [3], [4], [5]. These approaches are usually based on the decomposition of global QoS constraints into local ones. However, they are not appropriate when handling time-dependent QoS values. In fact, selecting the best service of each task based on local QoS constraints can not guarantee that temporal constraints will be satisfied. Despite several works dealing with the service selection problem, most of the selection approaches consider static QoS values and do not deal with temporal properties.

In this paper, we propose a novel heuristic approach to select a near-to-optimal solution while considering QoS and temporal constraints. First, pruning techniques are applied to reduce the number of candidate services that have to be considered in the selection process. After that, we detail our heuristic approach that allows selecting a close-to-optimal solution (i.e., combination of services) that satisfies global constraints and maximizes overall utility. The proposed approach is based on clustering and constraints decomposition techniques and local selection while dealing with temporal properties. Our

experiments revealed that our heuristic approach performs better than existing selection approaches and reaches solutions at least as good as the optimal solution.

The paper is structured as follows. In Section II, we specify the selection problem. In Section III, we give an overview of our pruning approach based on QoS and temporal constraints. In Section IV, we detail our heuristic selection approach. In Section V, we evaluate the proposed approach through experimental results. Finally, Section VI illustrates some existing works and Section VII concludes the paper.

II. Service Selection Problem

Service selection problem we are interested in can be described as follows: given an abstract business process in which complex structural and temporal constraints are specified, and a set of candidate services which offers time-dependent QoS, our goal is to select a close-to-optimal combination of services that meets global user's requirements while satisfying business constraints.

1) Constraints Specification: A composed service is usually specified as an abstract business process. Each business process is composed by a set of activities or abstract tasks $\mathcal{A} = \{A_1, ..., A_n\}$ which depend on each other by a set of structural dependencies that can follow different patterns including sequential, parallel, choice and loop patterns [6]. In addition to *structural constraints*, business designers can also require supplementary *intra* and *inter-task temporal constraints* at the business level in order to increase their market share and profitability. *Temporal constraints* define constraints related to the start and the finish time of business tasks. They can be associated to a given business task (e.g., a task that must start no earlier than a time point T) or to express dependencies between two directly or indirectly succeeding tasks (e.g., a task *A* that has to start no earlier than *v* time units after the start of a task *B*).

Besides business constraints, *global constraints* are required by users in order to select the best combination of services. A global constraint for a QoS attribute q_y is denoted by $Q(q_y)$. Hereafter, we denote by $\mathcal{QS}$ the set of QoS attributes. These latter can be classified into: negative attributes (which values need to be minimized) and positive attributes (which values need to be maximized). For simplicity, in this paper, we consider only negative QoS attributes since positive attributes can be transformed into negative ones by multiplying their values by -1.

2) Timed Service Instances: Each component abstract business task $A_i \in \mathcal{A}$, has a set of concrete candidate services that can implement it (i.e., a service class). The set of candidate services, denoted by $\mathcal{S}_i$, of each business task A_i contains services with the same functionality but with different QoS values. As stated previously, in this paper, we consider time-dependent QoS values. Thus, each candidate service $S_{ij} \in \mathcal{S}_i$ can present several timed instances with respect to the different time span during which it offers different QoS values. Each service instance S_{ijk} is characterized by a time span T_{ijk} and a quality value $Q(S_{ijk}, q_y)$ for each quality attribute $q_y \in \mathcal{QS}$. We denote by $\mathcal{T}_{ij}$ the set of time intervals of the service S_{ij}. The start and the finish times of each time span $T_{ijk} \in \mathcal{T}_{ij}$ are denoted by t_{ijk}^{min} and t_{ijk}^{max} respectively. Table I presents some extra notations used through this paper.

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER.

Notation	Description
m	The number of quality attributes
q_y	The y^{th} quality attribute with $1 \leq y \leq m$
W_y	The weight of q_y given by the user
$Q(A_i, q_y)^{min}$ $Q(A_i, q_y)^{max}$	Minimum and maximum values of q_y of the task A_i respectively
$Q(q_y)^{min}, Q(q_y)^{max}$	Minimum and maximum aggregated values of q_y respectively
$Q(A_i, q_y)$	The value of q_y of the selected service of A_i
Agg	The aggregation function that derives the quality values of the composite service from the quality values of its component services
$Q(CS, q_y)$	The value of q_y of the composite service CS with $Q(CS, q) = Agg_{A_i \in \mathcal{A}}(Q(A_i, q))$
$\mathcal{P}d(A_i)$	The set of predecessors of the task A_i

III. Overview of Our Pruning Approach

The aim of the pruning process is to select the most adequate services while guaranteeing that the optimal solution can still be found. This is based on a set of *local thresholds* such that the unsatisfaction of one of these thresholds by a candidate service guarantees the unsatisfaction of the global constraints. In other words, if a service instance violates at least one local threshold, all possible combinations of services that include it will violate global constraints. Consequently, it can be pruned from the set of candidate services to narrow the search space.

To better illustrate our idea, we consider the example in Figure 1. In this example, we suppose that the business process has two sequential abstract tasks A_1 and A_2 and that there are two QoS attributes: the cost and the execution time. Global QoS constraints are defined as follows: $Q(cost) \leq 5$ and $Q(time) \leq 6$. To compute local thresholds, first, we identify minimum values of the QoS attributes for both tasks. In this example: $Q(A_1, cost)^{min} = 2$, $Q(A_2, cost)^{min} = 1$, $Q(A_1, time)^{min} = 1$ and $Q(A_2, time)^{min} = 2$. Based on these values and on the global QoS constraints, we can compute local QoS thresholds for each quality attribute of both tasks (presented by the dashed lines in Figure 1). For instance, as the cost is an additive attribute and since the minimum cost value of the task A_2 is equal to 1 in the best case, all services with a cost greater than 4 of the task A_1 can not be part of a feasible solution. In fact, if one of these services is selected, the global cost constraint will be violated even if the service with the minimum cost value of the second task is selected. Thus, all services in the grey area will be eliminated. By the same way, we can define the remainder thresholds.

Besides QoS thresholds, temporal thresholds can also be defined to prune services according to their time availability based on the deadline of the overall process. Nevertheless, computing these thresholds is a very hard task when handling complex business structures including several structural

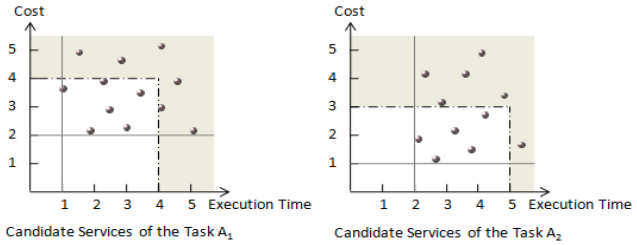


Fig. 1. Preselected services based on our pruning approach

patterns and dealing with different QoS categories such as additional, multiplicative, average and min-max operators. In addition, intra and inter-task temporal constraints have to be considered when measuring local thresholds. This is not trivial since some constraints may be overlapped or included in each others. To overcome these issues, in [6] and [7], we have proposed a set of constraint optimization models to compute local thresholds while dealing with complex business structures and several local and inter-task constraints. To avoid discarding any service that might be part of a feasible solution, local thresholds of each task are relaxed as much as possible. For more details we refer reader to [6] and [7].

IV. Heuristic Selection Approach

In this section, we present our heuristic approach while considering time-dependent QoS values. The proposed approach proceeds through four phases: A. *service clustering*, B. *selection of the best centroids*, C. *deadline decomposition*, and D. *local selection*. Hereafter, we present each step.

A. Service Clustering

In the literature, some works proposed decomposition techniques to decompose global constraints to local ones [3], [4], [5]. Nevertheless, these approaches deal with QoS values of each candidate service independently and do not consider correlations between them. To cope with this issue, we propose a clustering based approach to compute local QoS constraints while dealing with dependencies between QoS values of each candidate service.

The clustering phase is performed locally for each abstract task in the business process. It aims to classify candidate services of each abstract business task into a set of clusters (i.e., QoS levels) according to their QoS values. Each cluster contains services that have approximately similar QoS values. The purpose of this classification is to define the most important cluster for each task with respect to the number of its candidate services and their QoS values. To do so, we use clustering techniques in particular the K-means algorithm [8].

1) K-means algorithm overview: The K-means algorithm is commonly used to automatically partition a data set into a fixed number of groups (i.e., clusters). The main idea of this algorithm is to define a centroid for each group and then, associate each data point to the appropriate centroid (i.e., the centroid that has the shortest distance with the data point according to multiple parameters). For instance, suppose a data point x_i which is characterized by a set of values defined by the vector $\langle Q(x_i,q_1), Q(x_i,q_2), ..., Q(x_i,q_m)\rangle$ and a centroid $c = \langle Q(c,q_1), Q(c,q_2), ..., Q(c,q_m)\rangle$, thus, the Euclidean distance between x_i and c can be defined as follows:

$$D(x_i,c) = \sqrt{\sum_{y=1}^{m}(Q(c,q_y) - Q(x_i,q_y))^2} \quad (1)$$

The values of the centroids are then updated by computing the average of the values of all their associated data points for each parameter. The clustering and the updating steps can be repeated until there is no changes in the values of the centroids or until a stop criteria is reached (e.g., convergence threshold, maximum number of iterations).

2) Classification of services: In our approach, we use K-means clustering to associate services into a set of QoS levels. Each candidate service is considered as a data point which is characterized by its QoS values denoted by $S_{ijk} = \langle Q(S_{ijk},q_1), Q(S_{ijk},q_2), ..., Q(S_{ijk},q_m)\rangle$. Furthermore, each vector of QoS levels is considered as a centroid. In this step, the range of each quality attribute $q_y \in \mathcal{QS}$ is partitioned into a set of K discrete quality levels for each abstract task where K is a constant number strictly greater than 1. We suppose that the number of levels (i.e., K) is fixed by domain experts and can be different from one task to another according to the values of its candidate services. In the following, we denote by QL_{iy}^z the QoS value of the attribute q_y of the z^{th} level for the task A_i with $1 \leq z \leq K$ and $1 \leq y \leq m$. To speed up the classification algorithm, we compute the initial values of QoS levels as follows:

$$QL_{iy}^z = Q(A_i,q_y)^{min} + \frac{z-1}{K-1} * (Q(A_i,q_y)^{max} - Q(A_i,q_y)^{min}) \quad (2)$$

Hence, based on the values of QoS levels, the initial set of centroids can be defined. Let's denote by $QL_i = \langle QL_i^1, QL_i^2, ..., QL_i^K\rangle$ the set of centroids of the task A_i with $QL_i^z = \langle QL_{i1}^z, QL_{i2}^z, ..., QL_{im}^z\rangle$ denotes the z^{th} centroid of A_i for each $1 \leq z \leq K$. Once all centroids are defined, we assign each candidate service to the closest centroid using the Euclidean distance (as defined in 1).

B. Local QoS constraints specification

This phase aims to compute local QoS constraints for each business task based on the set of centroids. It is based on two main steps: centroid utilities and the selection of the best centroids.

1) Centroid utilities: To compute local QoS constraints, first, we assign each centroid QL_i^z an utility value (i.e., $U(QL_i^z)$) between 0 and 1 which estimates the benefit of using the QoS values of this centroid as local QoS constraints. The utility value of each centroid is computed as follows:

$$U(QL_i^z) = U_q(QL_i^z) * \frac{r(QL_i^z)}{l}, \forall A_i \in \mathcal{A}, \forall 1 \leq z \leq K \quad (3)$$

Where:

$$U_q(QL_i^z) = \sum_{y=1}^{m} W_y * \frac{Q(A_i,q_y)^{max} - QL_{iy}^z}{Q(A_i,q_y)^{max} - Q(A_i,q_y)^{min}} \quad (4)$$

The first part (i.e., $U_q(QL_i^z)$) specifies the utility of the centroid based on its QoS values. The second (i.e., $\frac{r(QL_i^z)}{l}$) allows giving better utility value to the centroid that has more candidate services where $r(QL_i^z)$ is the number of candidate services of the centroid QL_i^z and l is the total number of services of the task A_i.

2) The selection of the best centroids: The second step allows the identification of the best centroid of each business task. QoS values of the selected centroids will be considered as local QoS constraints in the selection process. We propose a constraint optimization model to find the best centroids such that all global QoS constraints are satisfied. To select only one centroid for each task, we use a binary decision variable x_i^z for each centroid such that $x_i^z = 1$ if the centroid QL_i^z is selected for the abstract task A_i and $x_i^z = 0$ otherwise which is expressed in Constraint (5).

$$\sum_{z=1}^{K} x_i^z = 1, \forall A_i \in \mathcal{A}, x_i^z \in \{0,1\} \quad (5)$$

The goal of the objective function (6) is to maximize the utility value of the set of the selected centroids in order to reduce the number of discarded services.

$$\text{maximize} \quad \sum_{A_i \in \mathcal{A}} \sum_{z=1}^{K} U(QL_i^z) * x_i^z \quad (6)$$

To guarantee that the QoS values of the selected centroids ensure that the global QoS constraints will be satisfied, we add Constraint (7). In this paper, we do not give any details about the aggregation function. More details about this function while considering several QoS categories and complex business structures can be found in our previous work [6].

$$Agg_{A_i \in \mathcal{A}} \ (\sum_{A_i \in \mathcal{A}} \sum_{z=1}^{K} QL_{iy}^z * x_i^z) \leq Q(q_y), \forall 1 \leq y \leq m \quad (7)$$

Additionally, the selected centroids must ensure that the overall deadline is fulfilled. To do so, we add Constraint (8) which guarantees that the sum of the minimum start time of the first task and the aggregated duration value of the selected centroids satisfies the required deadline. In this constraint, st_0 indicates the minimum start time of the first business task where $st_0 = min\{t_{0jk}^{min}\}, \forall S_{0j} \in \mathcal{S}_0$ and y refers to the execution duration attribute.

$$st_0 + Agg_{A_i \in \mathcal{A}} \ (\sum_{A_i \in \mathcal{A}} \sum_{z=1}^{K} QL_{iy}^z * x_i^z) \leq deadline \quad (8)$$

C. Deadline Decomposition

Unlike existing works [3], [5] which handle static QoS values, specifying local QoS constraints does not guarantee that all services which satisfy these constraints can collaborate when considering time-dependent QoS values. For instance, suppose that the business process has two abstract tasks A_1 and A_2 which will be executed in sequence with A_1 precedes A_2. Let's denote by S_1 and S_2 the best services that satisfy all local QoS constraints of the two tasks A_1 and A_2 respectively. Suppose now that the service S_2 is available in a time span before that of the service S_1. In this case, the two services can not be part of a feasible solution even though they satisfy all local QoS constraints.

To this end, temporal properties have to be considered also to identify local temporal constraints that have to be satisfied by the selected services to guarantee that all selected services can collaborate together. To do so, we identify four variables for each task: earliest start time est^m, earliest finish time eft^m, latest start time lst^M and latest finish time lft^M. The values of these variables are defined based on the minimum and the maximum duration values of each task and should guarantee that all intra and inter-task temporal constraints are satisfied and the overall deadline is respected.

To better illustrate our idea, let's take the example presented in Figure 2. In this example, we consider two abstract business tasks. For each task, we compute the largest time intervals based on both minimum and maximum duration values as well as the minimum start time and maximum finish time of its candidate services. In this example, we suppose that the deadline is equal to 38 time units. The local temporal values of each task are depicted in Figure 2.

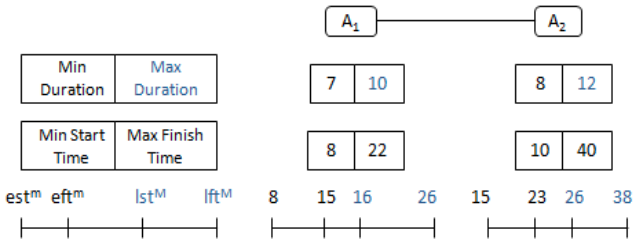


Fig. 2. Example of deadline decomposition

The specification of the local temporal constraints of each activity is not a trivial task when handling several temporal constraints and complex business structures. To deal with this, time intervals are identified based on constraint optimization model. This model can be applied in parallel for each task $A_i \in \mathcal{A}$. The objective function (9) allows maximizing the value of lft_i^M while minimizing the value of est_i^m.

$$\text{maximize} \quad lft_i^M - est_i^m \quad (9)$$

To specify the dependencies between the start and the finish time of each task, we add Constraints (10) and (11). Here, we consider that the difference between est_i^m and eft_i^m is equal to the minimum duration value for the task A_i (i.e., $Q(A_i, dur)^{min}$) whereas the difference between lst_i^M and lft_i^M is equal to the duration of the selected centroid for the task A_i (i.e., QL_{iy}^z) which is considered as the maximum duration value of A_i, with z refers to the selected centroid of the task A_i and y refers to the duration attribute.

$$eft_i^m = est_i^m + Q(A_i, dur)^{min}, \forall A_i \in \mathcal{A} \quad (10)$$

$$lft_i^M = lst_i^M + QL_{iy}^z, \forall A_i \in \mathcal{A} \quad (11)$$

To satisfy precedence dependencies, we add Constraints (12) and (13). For simplicity, here we consider basic business structures.

$$eft_i^m \leq est_j^m, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \quad (12)$$

$$lft_i^M \leq lst_j^M, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \quad (13)$$

Besides, to guarantee that the overall deadline is fulfilled, we add Constraint (14).

$$lft_n^M \leq deadline \quad (14)$$

Finally, Constraints (15) and (16) deal with temporal constraints. For simplicity, in this model, we only consider end-to-start inter-task temporal constraints. Here, we denote by $td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max})$ the temporal dependency between the end of the task A_i and the start of the task A_j (i.e., ES), with D_{ij}^{min} and D_{ij}^{max} denote the minimum and the maximum durations between the start and the finish times respectively. Thus, $\forall\ td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD}$ with $\mathcal{TD}$ is the set of inter-task temporal dependencies, the following constraints have to be satisfied:

$$eft_i^m + D_{ij}^{min} \leq est_j^m \leq eft_i^m + D_{ij}^{max} \quad (15)$$

$$lft_i^M + D_{ij}^{min} \leq lst_j^M \leq lft_i^M + D_{ij}^{max} \quad (16)$$

The start and finish times of each task will be considered as local temporal constraints and thus, used in the next step to find a candidate service for each abstract task.

D. Local Selection

After defining local QoS and temporal constraints, the last step of our approach is to select a close-to-optimal solution. The local selection is to find the best service of each business task such that all local QoS and temporal constraints are fulfilled. The best service is the service that has the best utility value amongst all the candidate services of the corresponding service class. This phase is made through a simple selection algorithm which can be applied in parallel for each task. The selection process is presented in Algorithm 1.

Algorithm 1 Service Selection Algorithm

```
1:  for each S_ij ∈ S_i do
2:      for each T_ijk ∈ T_ij do
3:          ComputeUtility(S_ijk)
4:  RankServices(A_i)
5:  for each S_ij ∈ S_i do
6:      for each T_ijk ∈ T_ij do
7:          for each q ∈ QS do
8:              if Q(S_ijk, q) ≥ Q_L(A_i, q) then
9:                  break
10:         if t_ijk^min ≤ lst_i^M and t_ijk^max ≥ eft_i^m then
11:             SelectService(S_ijk)
12:             DefineTimeIntervals(S_ijk)
13:             break
```

First, we rank candidate services of each business task according to their utilities (lines 1 to 4). The utility of each service is quantified by the utility function declared in (17).

$$U(S_{ijk}) = \sum_{y=1}^{m} W_y * \frac{Q(A_i, q_y)^{max} - Q(S_{ijk}, q_y)}{Q(q_y)^{max} - Q(q_y)^{min}} \quad (17)$$

For each task, we select the candidate service with the best utility and that meets all local QoS (lines 5 to 9) and temporal (lines 10 to 11) constraints of its corresponding task. Local QoS constraints are considered as upper bounds for QoS values of the different candidate services. To deal with temporal constraints, a candidate service is selected if it can start before the lst^M and finish after the eft^m of its corresponding task. Here, we note that the time span of each service instance covers its execution duration (i.e., $t_{ijk}^{max} - t_{ijk}^{min} \geq Q(S_{ijk}, dur)$). For each selected service, we identify the time spans in which it can start and finish so that it can collaborate with other selected services (line 12). To do so, we add the following constraints:

$$max(t_{ijk}^{min}, est_i^m) \leq st_i \leq lst_i^M \quad (18)$$

$$eft_i^m \leq ft_i \leq min(t_{ijk}^{max}, lft_i^M) \quad (19)$$

With st_i and ft_i are the start and the finish time of the selected service of the task A_i. Hence, if these two constraints are satisfied, we can guarantee that the selected services can form a feasible solution. The start time of each service is then defined at run time with respect to the Constraints (18) and (19) and the start and the finish times of its precedent services. In the following, we demonstrate that the selected services based on local QoS and temporal selection can collaborate with each other while satisfying all constraints.

Lemma: If all the selected services satisfy local QoS constraints and time spans (identified by the deadline decomposition step) and guarantee Constraints (18) and (19), these services can collaborate with each other and all local and global constraints are fulfilled.

Proof. First, since all selected services fulfil all local QoS constraints, global QoS constraints will be satisfied according to (7). In other hand, suppose that the selected service of the first task (i.e., $A_i \in \mathcal{A}$) meets Constraints (18) and (19), consequently, three cases arise: (a) If $st_i = est_i^m$, then $ft_i = eft_i^m$ if the selected service has the minimum duration value and $ft_i \leq lft_i^M$ otherwise since $st_i \leq lst_i^M$. Therefore, the service that will implement the successor task (i.e., A_j with $A_i \in \mathcal{P}d(A_j)$) can verify $est_j^m \leq st_j \leq lst_j^M$ according to (12) and (13) and hence, it also satisfies its local temporal constraints. (b) If $st_i = lst_i^M$, then, $ft_i \leq lft_i^M$ if the selected service has the minimum duration value and $ft_i = lft_i^M$ otherwise. Hence, in the worst case (i.e., the duration of the selected service is equal to the duration of the selected centroid for the task A_i), the successor service can start in its allowed time span. (c) If $est_i^m < st_i < lst_i^M$, then, $eft_i^m < ft_i < lft_i^M$ which also guarantee that the remaining selected services will meet their local temporal constraints. We note that in all cases, intra and inter-task temporal constraints and the overall deadline are satisfied since they are considered

in the deadline decomposition phase (Constraints from (14) to (16)). Moreover, the maximum durations of the selected services guarantee the overall deadline according to (8). □

V. Performance Evaluation

In this section, we evaluate the performance of the proposed approach by studying its time complexity and analyzing experimental results based on simulation studies.

A. Complexity evaluation

In this subsection, we analyze the complexity of the different phases of the proposed approach.

1. Service clustering: This phase includes the specification of the centroids and the classification of services. First, for each task, the number of computed levels is K for each QoS attribute. Hence, the complexity of the first step is $O(n*K*m)$ with n is the number of business tasks, m is the number of QoS attributes and K is the number of centroids. Since the classification of services can be done in parallel for all tasks, the complexity of this step is $O(l*K*m*it)$ with l is the number of services per task and it is the number of iterations. Our experiments show that the maximum number of required iterations is 2. Hence, the complexity of the service clustering phase is $O(n*K*m+l*K*m*2)$.

2. Local QoS constraints specification: The number of decision variables of the constraint optimization model to find the best local QoS constraints (i.e., the best centroids) is $n*K$. Consequently, the complexity of this phase is $O(2^{n*K})$.

3. Deadline decomposition: This step relies on the use of four variables (earliest start, earliest finish, latest start, and latest finish times) and it can be applied in parallel for all tasks. The complexity of this step is $O(4*no)$ with no is the operation number of equation resolution.

4. Local selection: The best service of each task can be selected locally with a complexity of $O(l)$.

Based on the above analysis, the overall complexity of our approach is: $O(n*K*m+l*K*m*2+2^{n*K}+4*no+l) = O(2^{n*K})$. Therefore, the time complexity of the proposed approach is dominated by the local QoS constraints specification phase whose complexity does not depend on the number of candidate services which enhances its scalability. Let us now compare our approach with existing works [9], [3], [2], [1]. If $K < h$ with h is the number of promising services per task in [9], our approach achieves better performance than this work which does not deal with time-dependent QoS and temporal constraints. Furthermore, if $K << m*d$ with d is the number of QoS levels in [3] and $K << l$, we can ensure that the size of our model is much smaller than those of the models proposed in [3], [2], [1] even though these works consider only static QoS values.

B. Experimental Results

Experiments have been performed on a laptop with a 32 bit Intel Core 2.20 GHz CPU and 4GB RAM and ubuntu 14.04 as operating system. To implement our approach, we used the constraint solver Choco[1]. To evaluate the effectiveness of our approach, we study its performance in terms of the computation time and the optimality comparing to the global optimization approach (global approach for short) in which the optimal solution is selected. We compare the results of the different tests in four cases: the global approach that searches the optimal solution without pruning (GWO), the global approach with pruning (GW), our heuristic approach without pruning (HWO) and the heuristic approach with pruning (HW). In all cases, the number of temporal constraints is fixed to 3.

First, we analyze the computation time of the different test cases in terms of candidate services per task and the number of business tasks (Figure 3).

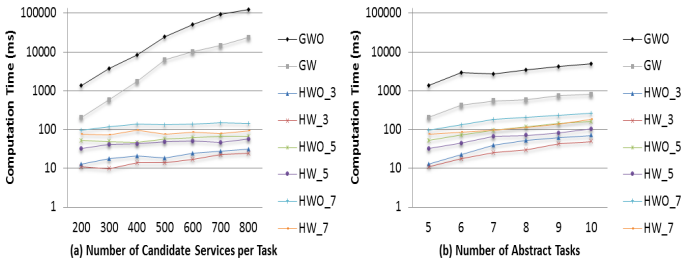


Fig. 3. Performance of the selection approach in terms of the computation time

Figure 3(a) shows the computation time of the different approaches. The number of business tasks and global constraints is fixed to 5 and 4 respectively. The number of centroids varies between 3 and 7. QoS attributes service instances were generated for a time horizon with 150 time points and distributed in the range between 1 and 100. The results indicate that the computation time of the heuristic approach is very small comparing to the global approach with and without pruning. In addition, the computation time is smaller when applying our pruning techniques before the selection process. This is explained by the fact that the number of services is reduced when applying our pruning approach. In addition, the domains of the QoS values of the centroids are more smaller due to the consideration of local thresholds. Hence, a near-to-optimal solution can be found more quickly. In our previous work [6], we demonstrated that the computation time of the pruning process is equal to 23 ms in the worst case. This is an expected result since the computation of the different QoS and temporal local thresholds can be applied in parallel. Experiments also show that although the computation time increases exponentially when the number of services rises in the global approaches, it is relatively stable in our approach. However, the computation time increases slowly when the number of centroids increases. This is obvious since the time of the local QoS constraints specification phase increases along with the number of centroids. Figure 3(b) shows the computation time of the selection approaches when the number of tasks varies between 5 and 10 and the number of global constraints is fixed to 4 with 200 candidate services for each

[1]http://www.emn.fr/z-info/choco-solver/

task. Again, experiments show that our approach outperforms the other approaches. In fact, the computation time of our approach increases very slowly along with the number of tasks.

Furthermore, we studied the optimality of the proposed approach with respect to the number of services and business tasks (Figure 4).

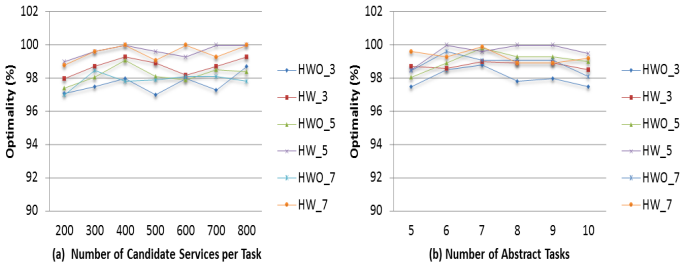


Fig. 4. Evaluation of the optimality of the selection approach

To evaluate the optimality of our approach, we compare the utility value of the obtained solution with the utility value of the optimal solution of the global approach. The utility value of the composite service is computed as follows:

$$U(CS) = \sum_{y=1}^{m} W_y * \frac{Q(q_y)^{max} - Q(CS, q_y)}{Q(q_y)^{max} - Q(q_y)^{min}} \qquad (20)$$

Figure 4(a) shows that by increasing the number of centroids, the optimality of the selected solution increases as well. However, if the number of centroids increases more than necessary, the optimality does not change in most cases and can decrease in some cases. This is due to the fact that when the number of centroids is very high, the classification of services is no more efficient since candidate services will be distributed in a very large set of QoS levels which can affect the utility values of the selected centroids. Moreover, the optimality of the heuristic approach with pruning is better than the optimality of the approach without pruning. This is due to the fact that the computed centroids are more accurate after the pruning process. Additionally, Figure 4(b) also shows that our approach can produce a satisfying optimality (i.e., more than 97 %) in most cases. Again, applying our pruning techniques before the selection process allows reaching better results since the selection algorithm is applied on the set of the most significant services. Moreover, in contrast to the selection approach without pruning which can reach a reasonable optimality when increasing the number of centroids, the heuristic approach with pruning can obtain better optimality even when the number of centroids is very small. This is because when considering all candidate services, several inadequate instances can affect the values and the utilities of the centroids.

We can conclude that in all test cases, our approach can achieve a satisfactory optimality with a very small computation time. However, there is a significant trade-off between the number of the centroids and the computation time and the optimality of the proposed approach. In fact, when the number of centroids increases, the computation time of the selection process increases and the optimality increases as well and inversely. Hence, the choice of the number of centroids is of great importance. It should not be very high to reduce the computation time and find an efficient classification of services. Additionally, it should not be very small so that we can find the most adequate centroids and thus, a close-to-optimal solution. Results also show the advantage of the pruning process before the selection phase. In fact, although the pruning step can increase the computation time, this latter is still very negligible compared to the time of the selection approach without pruning. This is mainly due to the fact that local thresholds of the pruning process can be computed in parallel for all tasks and for each QoS attribute [6] in contrast to the local QoS constraints that can not be computed in parallel and that requires more computation time when no pruning techniques are applied. Thus, the pruning process allows: (i) reducing the computation time of the selection process and (ii) achieving more accurate results with a very small number of centroids.

VI. Related Work

Several selection algorithms have been proposed in the literature to select services for abstract business tasks. Some works aim at searching for the optimal combination of services using exhaustive algorithms such as linear programming techniques, constraint satisfaction models and global planning [1], [2], [10]. These methods have a high computation time, thus, they are only suitable for small problems. Other approaches are based on evolutionary algorithms to select a near-to-optimal solution more efficiently than exact methods such as immune and genetic algorithms [11], [12]. In these strategies, services are usually selected randomly unlike our approach in which services are selected based on their utility values which can enhance the performance of the selection process. All these approaches do not provide techniques to reduce the search space and can not deal with temporal properties.

Other alternatives propose pruning strategies to reduce the number of candidate services to be considered. Some works prune services while guaranteeing that the optimal solution still be found [6], [7], [13]. Although they enhance the efficiency of the selection algorithm, they are limited in complex selection problems where the number of services is very huge. Other proposals propose pruning techniques to select a near-to-optimal solution. The most adopted technique is the decomposition of global constraints to local ones [3], [4], [5]. Local optimization is then applied to select the best service for each abstract task.

Several methods have been proposed to compute local QoS constraints based on global ones. Alrifai et al. [3] use a mixed integer programming model to compute local constraints of each task based on a set of QoS levels for each QoS attribute. In [4], Sun et al., introduce a QoS decomposition approach based on the mean and the standard deviation of each QoS attribute while considering several composition structures. Another approach is proposed in [5] to define local constraints using genetic algorithm. The main drawback of these works is that they extract local QoS levels of static quality values while

dealing with QoS attributes independently. Hence, possible dependencies and correlations between QoS attributes of every candidate service are not considered. This can not effectively represent the quality levels of candidate services and may lead to a very restrictive decomposition strategy that can not be satisfied by any combination of candidate services even though a solution to the selection problem does exist. To cope with this limitation, our approach adopts clustering algorithm to select local constraints while considering correlations between quality attributes of each service. In addition, in contrast to our approach, these works do not provide pruning strategies before the decomposition phase which can negatively affect the values of local constraints.

K-means algorithms have been applied in several application domains such as data mining and image segmentation. Recently, some works use K-means clustering in service selection problems. In [14], Ben Mabrouk et al. use K-means algorithm to classify services according to their QoS values. Clusters are used to compute the utility of each candidate service in order to rank services according to their utilities. A search tree is then applied to select the best combination of services by checking services in an ordered way. The proposed solution is computationally expensive since it does not consider local constraints and local selection of services. Furthermore, this work can not deal with time-dependent QoS values.

Some works deal with temporal properties in the service selection problem. In [15], Liang et al. use genetic algorithm to select services under temporal constraints while dealing with static QoS values. In [16], Wagner et al. propose a multi-objective optimization based approach while considering time-dependent QoS values. The proposed approach selects the best combination of services as well as the start and finish time of each service according to the QoS values at each time period. In [17], Klopper et al. take into consideration time-dependent QoS values when selecting the best service instances. In this work, authors suppose that all QoS attributes are monotonically decreasing. Moreover, both works [16] and [17] do not consider temporal constraints at the business level and do not provide any search space reduction techniques prior to the selection process. Time decomposition has been considered in [18] to deal with scheduling problems. In this work, Yu et al. propose a deadline decomposition approach to decompose the global deadline into sub-deadlines based on the minimum duration values. However, in this work, authors rely on a greedy decomposition method that may discard several interesting solutions. To overcome this limitation, in our approach, we compute the largest time interval in which each selected service can collaborate with other services.

VII. Conclusion

In this paper, a heuristic time-aware service selection approach has been proposed. Unlike existing works, the proposed approach deals with time-dependent QoS values and temporal properties. First, inadequate services are discarded based on QoS and time pruning techniques to reduce the search space. After that, we define a novel approach that combines the use of local QoS and temporal constraints with local selection. Local QoS constraints are identified based on K-means clustering algorithm and constraint optimization models. To deal with temporal properties, we proposed a deadline decomposition approach. Finally, a local selection is applied to select a close-to-optimal solution with a linear complexity algorithm. The results of the experimental evaluation show a significant performance gain with our approach in comparison to global optimization-based approaches while still obtaining a very close-to-optimal solutions. In the future, we plan to extend the proposed approach to deal with run time service selection and reselection while considering run-time violations and environment changes. We also aim to consider dependencies between QoS values of the different candidate services in the pruning and the selection processes and deal with real-world data and different distributions of services.

References

[1] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Software Eng.*, vol. 33, pp. 369–384, 2007.

[3] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *WWW*, 2009, pp. 881–890.

[4] S. X. Sun and J. Zhao, "A decomposition-based approach for service composition with global qos guarantees," *Inf. Sci.*, vol. 199, pp. 138–153, 2012.

[5] F. Mardukhi, N. Nematbakhsh, K. Zamanifar, and A. Barati, "Qos decomposition for service composition using genetic algorithm," *Appl. Soft Comput.*, vol. 13, pp. 3409–3421, 2013.

[6] I. Guidara, N. Guermouche, T. Chaari, M. Jmaiel, and S. Tazi, "Time-dependent qos aware best service combination selection," *Int. J. Web Service Res.*, vol. 12, 2015. (to appear).

[7] I. Guidara, N. Guermouche, T. Chaari, S. Tazi, and M. Jmaiel, "Pruning based service selection approach under qos and temporal constraints," in *ICWS*, 2014, pp. 9–16.

[8] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[9] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for qos-aware web service composition," in *ICWS*, 2010, pp. 34–41.

[10] A. B. Hassine, S. Matsubara, and T. Ishida, "A constraint-based approach to horizontal web service composition," in *International Semantic Web Conference*, 2006, pp. 130–143.

[11] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *GECCO*, 2005, pp. 1069–1075.

[12] J. Xu and S. Reiff-Marganiec, "Towards heuristic web services composition using immune algorithm." in *ICWS*. IEEE Computer Society, 2008, pp. 238–245.

[13] L. Barakat, S. Miles, I. Poernomo, and M. Luck, "Efficient multi-granularity service composition," in *ICWS*, 2011, pp. 227–234.

[14] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "Qos-aware service composition in dynamic service oriented environments," in *Middleware*, 2009, pp. 123–142.

[15] H. Liang, Y. Du, and S. Li, "An improved genetic algorithm for service selection under temporal constraints in cloud computing," in *WISE (2)*, 2013, pp. 309–318.

[16] F. Wagner, A. Klein, B. Klöpper, F. Ishikawa, and S. Honiden, "Multi-objective service composition with time- and input-dependent qos," in *ICWS*, 2012, pp. 234–241.

[17] B. Klöpper, F. Ishikawa, and S. Honiden, "Service composition with pareto-optimality of time-dependent qos attributes," in *ICSOC*, 2010, pp. 635–640.

[18] J. Yu, R. Buyya, and C. Tham, "Cost-based scheduling of scientific workflow application on utility grids," in *e-Science 2005*, 2005, pp. 140–147.