

Time-Dependent QoS Aware Best Service Combination Selection

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

▶ **To cite this version:**

Ikbel Guidara, Nawal Guermouche, Tarak Chaari, Mohamed Jmaiel, Saïd Tazi. Time-Dependent QoS Aware Best Service Combination Selection. International Journal of Web Services Research, 2015, 12 (2), pp.1. 10.4018/IJWSR.2015040101 . hal-01396995

HAL Id: hal-01396995
https://hal.science/hal-01396995

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.

Time-dependent QoS aware Best Service Combination Selection

Ikbel Guidara[1,2], Nawal Guermouche[3], Tarak Chaari[1], Mohamed Jmaiel[1], and Said Tazi[2]

[1]ReDCAD Laboratory, University of Sfax,
National School of Engineers of Sfax, B.P. 1173, 3038 Sfax, Tunisia
[2]Univ de Toulouse, UT1, CNRS, LAAS, 7 avenue du
colonel Roche, F-31400 Toulouse, France
[3]INSA, CNRS, LAAS, 7 avenue du colonel
Roche, F-31400 Toulouse, France

{iguidara, nguermou, tazi}@laas.fr
tarak.chaari@redcad.org
mohamed.jmaiel@enis.rnu.tn

ABSTRACT:

Service Oriented Architecture allows developing complex business applications from existing services. Given that many services are available with the same functionality and with different Quality of Service (QoS) attributes, one common challenge is to select the best service combination regarding user's requirements. Existing solutions often consider static QoS values for candidate services. Nevertheless, in real world applications, QoS values can change during time. In addition, besides structural constraints, several QoS and temporal constraints can also be specified at the business level. Considering time-dependent QoS values associated with business level constraints makes the selection process a very complex and time consuming decision problem given the large number of service combinations to be compared. To deal with this issue, in this paper, we propose a novel service selection approach based on QoS and temporal pruning techniques to reduce the number of candidate services. The proposed approach allows pruning uninteresting services based on a set of local thresholds. These latter are measured using constraint optimization models while dealing with general flow structures including sequential, parallel, choice and loop patterns and different types of QoS and temporal constraints. Experimental studies show the benefits of the proposed approach in particular in terms of computational time.

KEY WORDS:
Service Selection, Time-dependent QoS, Pruning, Constraint Optimization, Business Process

Introduction

Service Oriented Architecture (SOA) paradigm allows the integration of several service components to develop complex business applications. In advanced service-oriented computing, complex applications with QoS requirements are usually specified as abstract business processes with global QoS constraints. The execution of these applications requires the selection of a set of services to implement abstract business tasks while fulfilling the QoS constraints. With the growing number of potential candidate services of each business task that offer the same functionality but differ in their QoS attributes (e.g., response time, availability), the selection of the best combination of services that satisfies business process constraints and end-to-end user's requirements is a challenging task.

The selection of the best service combination with end-to-end QoS requirements has been widely treated in the literature (Hwang, Lim et al., 2008), (Ma, Bastani et al., 2013), (Ardagna and Pernici, 2007), (Zeng, Benatallah et al., 2004), (Yu, Zhang et al., 2007). Two strategies are generally adopted: *global*

optimization strategy which consists in searching candidate services that optimize the overall QoS of the business process and fulfill global QoS constraints while considering all possible combinations of services (Zeng, Benatallah et al, 2004), (Ardagna and Pernici, 2007) and *local optimization* strategy which aims to select the best service from each class of candidate services (i.e., candidate services for each task) independently from other classes. This strategy is especially used in distributed environments where a global QoS management is not required (Benatallah, Sheng et al., 2002). In recent years, some works proposed new service selection strategy adopting a top-down method. These approaches assume that global constraints can be considered as the aggregation of a set of local ones (Alrifai and Risse, 2009), (Qi, Tan et al., 2010), (Sun and Zhao, 2012), (Mardukhi, Nematbakhsh et al., 2013). Based on these latter, local optimization is applied to select the best service for each abstract task such that all local constraints are fulfilled. Despite active research in the context of service selection for abstract business processes, some issues still remain unsettled so far.

First, most selection approaches assumes that services are always available and that QoS values cannot change over time. However, within different time periods, QoS attributes of candidate services can have different values (Chen, Yang et al., 2011), (Wagner, Klein et al., 2012). For instance, the invocation of services during business hours can be more expensive than invoking them outside these hours. Thus, considering permanent availability of services and assuming static QoS values is very restrictive to effectively represent services and reflect the impact of time on the QoS attributes.

Second, in real world scenarios, several constraints can be specified at the business process level (e.g., structural, QoS and temporal constraints). Current selection approaches consider only structural constraints. Usually, temporal constraints are considered when modeling and verifying business processes (Lanz, Kolb et al., 2013), (Cheikhrouhou, Kallel et al., 2014) and neglected during the selection of the best combination of services. For example, a partner of electronics manufacturing organization can require in its business process that the manufacturing of peripheral parts has to finish no later than 20 time units after the starting of the process. Moreover, some QoS constraints can also be specified in the business process (Ardagna and Pernici, 2007). Considering QoS and temporal constraints when selecting the best service combination is a vital task since the violation of one or more constraints may affect the successful execution of the business process.

Third, existing approaches focus only on sequential flows between tasks. Business process models with other flow structures have to be transformed to sequential models (Cardoso, Sheth et al., 2004). However, merging several execution paths in one or multiple sequential flows may lead to lose some interesting dependencies between business tasks and thus violate some global constraints. Moreover, this transformation can be impossible to apply in very complex processes when several dependencies exist between abstract tasks mainly structural and temporal dependencies.

Fourth, when dealing with temporal properties, adopting a global optimization approach (Zeng, Benatallah et al, 2004), (Ardagna and Pernici, 2007) is not a practical solution and can lead to scalability issues because of the large number of constraints that should be considered comparing to existing approaches. Furthermore, although the decomposition of global constraints into local ones (Alrifai and Risse, 2009), (Qi, Tan et al., 2010) is a promising solution when selecting services based on static QoS values, this is not adequate to handle the problem we focus on. Indeed, selecting the optimal service of each task based only on local constraints without considering the temporal constraints and dependencies that can exist between services does not guarantee that the global collaboration of the selected services succeeds.

Consequently, a novel approach that allows selecting the optimal solution that satisfies all users constraints, while considering both *time-dependent QoS* and *complex constraint*s specified at business level and guaranteeing a good level of performance is still needed. To deal with this, we propose an hybrid approach that combines the use of local thresholds with global optimization to select the best solution in a

reasonable time. Our approach defines a rich representation of the service selection problem that allows the specification of structural, QoS and temporal constraints while considering time-dependent QoS values and complex composition structures including sequential, parallel, choice and loop patterns. In addition, it provides a *service pruning process* based on a set of computed *local thresholds* to eliminate non adequate services and thus, reduce the number of service combinations to be considered. The local thresholds are determined based on both QoS and temporal constraints while ensuring that only service combinations which are guaranteed to violate one or more constraints are not considered in the selection process. Based on the results of the pruning phase, we design and implement a selection algorithm that takes into consideration both time-dependent QoS attributes and business level constraints and ensures the selection of the best service combination. The evaluation results demonstrate the effectiveness of our pruning based selection algorithm especially in terms of computation time needed to select the best service combination.

The rest of this article is organized as follows. First, we present a motivating example followed by a formal description of the service selection problem. Then, we detail our pruning approach based on QoS and temporal constraints and we present our selection algorithm. Finally, we evaluate our approach through experimental results and illustrate some existing works.

MOTIVATING SCENARIO

In this section, we consider the case study of electronic devices manufacturing. Figure 1 shows the business process of the production of electronic devices in manufacture enterprise.

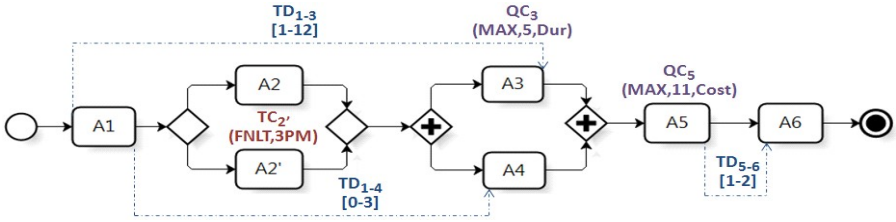


Fig. 1. The business process of the production of an electronic device

The production process has seven abstract tasks presented in Table 1. This process starts by receiving and writing technical reports (A_1). Then, the adequate model of the device has to be searched if it exists ($A_{2'}$) according to the requirements of the user or a new model has to be designed (A_2). After that, the manufacturing of the electronic and technical parts are executed in parallel (A_3 and A_4). After manufacturing the different parts of the product, these parts are assembled together to deliver the final product (A_5). Finally, the product has to be tested before delivering (A_6). For simplicity, in this example we do not consider loop structures.

Table 1. Description of the business process's activities

Activity	Description
A_1	Receive the order and write the technical reports
A_2	Design the model of the device
$A_{2'}$	Search for the adequate model
A_3	Manufacture the electronic parts of the device
A_4	Manufacture the peripheral parts of the device
A_5	Assemble all parts
A_6	Test the final product

In order to increase its market share and profitability, the enterprise sets some temporal constraints to get control over the production time of the product. In the example presented in Figure 1, three temporal dependencies are specified between tasks:

- $TD_{1,3}$: The manufacturing of electronic parts can be fulfilled at least one time unit and at most 12 time units after the reception of the order.
- $TD_{1,4}$: The manufacturing of peripheral parts starts no later than 3 time units after the writing of the technical reports.
- $TD_{5,6}$: The test of the final product can start no earlier than one time unit and no later than 2 time units before the assembly of all parts considering the necessary time of the transportation of the product after the assembly of its parts.

Moreover, given that some tasks can be released internally in the enterprise (i.e. expressed in its internal business view), this latter may have some local temporal constraints related to the availabilities of its internal departments. In our example, we consider the following temporal constraint related to the task A_2:

- TC_2 : The design of the product model has to finish no later than 3PM since it should be checked by the designer who is available every day from 8AM to 4PM.

Business designers can also require local QoS constraints. For instance, we consider the following QoS constraints:

- QC_3 : The execution duration of the task A_3 should not exceed 5 time units.
- QC_5 : The price of the task A_5 should be less than 11 Euros.

We suppose that three global constraints are associated to the process: (1) the price must not exceed 65 Euros, (2) the duration of the execution must be lesser than 13 units of time, and (3) the finish time of the process must not exceed 11 PM (i.e., 23 units of time in our example).

To be implemented, each abstract activity has a set of concrete functionally similar candidate services. Some of these services have several instances in which they offer different QoS values according to their temporal properties. We note by S_{ijk} the k^{th} instance of the j^{th} candidate service of the abstract activity A_i. For instance, as we can see in Figure 2, the 3rd service of the first activity has two instances: (S_{131}) the service offers 3 time units for the execution duration with a cost of 23 Euros from 9 to 13 units of time and (S_{132}) during 14 to 20 units of time, the service offers a duration of 4 time units and a cost of 19 Euros.

	Avail	Dur	Cost		Avail	Dur	Cost		Avail	Dur	Cost
S_{111}	[8, 15]	4	12	S_{211}	[12, 17]	1	10	S_{311}	[15, 22]	4	12
S_{121}	[17, 21]	2	8	S_{221}	[7, 13]	4	12	S_{321}	[8, 13]	6	12
S_{131} S_{132}	[9, 13] [14, 20]	3 4	23 19	$S_{2'11}$ $S_{2'12}$	[9, 12] [14, 21]	2 1	17 15	S_{331} S_{332}	[10, 14] [15, 20]	4 5	20 23
	Avail	**Dur**	**Cost**		**Avail**	**Dur**	**Cost**		**Avail**	**Dur**	**Cost**
S_{411} S_{412}	[8, 15] [16, 23]	7 5	22 18	S_{511} S_{512}	[7, 12] [14, 18]	2 3	8 18	S_{611}	[19, 23]	1	7
S_{421}	[17, 24]	6	15	S_{521}	[8, 16]	3	11	S_{621} S_{622}	[9, 13] [14, 18]	2 4	9 10
S_{431}	[14, 20]	4	12	S_{531}	[14, 22]	2	12	S_{631}	[8, 22]	1	6

Fig. 2. Candidate services of each abstract task

Let us now search the best service combination to implement the business process without taking into account global user's constraints. First, we suppose that all QoS values are static (i.e., they do not change during the time). Thus, the best combination of services that satisfies the user requirements is $C = (S_{121}, S_{211}, S_{311}, S_{431}, S_{511}, S_{631})$. This combination can be selected based on the notion of dominance (Alrifai and Risse, 2009), (Barakat, Miles et al., 2011). In other words, each selected service chosen to implement an abstract activity offers the best QoS (i.e., it dominates all the other candidate services).

As stated previously, QoS parameters can change according to time. When considering time-dependent QoS, the combination C is no more valid even if the selected services are the best for their corresponding activities. Indeed, despite that the task A_2 should be executed after the task A_1, the service S_{211} is available in a time span before that of the service S_{121} and thus, these two services cannot be parts of the same solution. The combination $C' = (S_{111}, S_{211}, S_{311}, S_{431}, S_{531}, S_{631})$ where availability intervals are respectively [8,12], [12,14], [14,18], [14,19], [19,21] and [21,22] is a satisfactory solution.

Again, the combination C' cannot be a satisfactory solution if we deal with further constraints expressed in the business process. Consider for example the temporal dependency $TD_{5,6}$. This constraint cannot be satisfied by the combination C' and thus, another service combination should be selected such as $C'' = (S_{111}, S_{211}, S_{311}, S_{431}, S_{531}, S_{611})$ with the following availability intervals [8,12], [12,14], [14,18], [14,19], [19,21] and [22,23].

To summarize, in this section, we intuitively show that considering time-dependent QoS attributes associated with business constraints is not a trivial task and makes the selection problem very complex. The existing selection approaches cannot be applied since most of them consider only static QoS constraints. To overcome these limitations, we present a time-aware service selection approach.

SPECIFICATION OF THE SELECTION PROBLEM

Service selection problem we are interested in consists in finding the adequate services so that constraints at business and service level and global user constraints are satisfied. Hereafter, we present the different constraints we consider and the utility function.

Global User Constraints

In order to select the best composite service CS (i.e., the best combination of services), the user specifies in his request a set of *global constraints* on QoS attributes. Let QS denotes the set of QoS attributes specified in the user requirements. In this paper, we consider four categories of QoS attributes that are widely used in the literature: Additive, Average, Multiplicative and Max-Operator (Li, Li et al., 2011), (Jaeger, Rojec-Goldmann et al., 2004), (Sun and Zhao, 2012).

Each QoS attribute $q \in QS$ has either an *increasing better value direction* (i.e., the quality is better when the attribute value increases) or a *decreasing better value direction* (i.e., the quality is better when the attribute value decreases). On the other hand, QoS attributes can be divided into two classes: *quantitative attributes* that can be measured using metrics (e.g., availability, response time) and *qualitative attributes* that cannot be measured and they are generally evaluated based on boolean values (e.g., privacy, security). For the sake of simplicity, henceforth, we do not consider QoS attributes with increasing value direction since they can be easily transformed to decreasing value direction based attributes by multiplying their values by -1. In addition, we consider only quantitative attributes since qualitative attributes can be considered as quantitative attributes that can be measured based on two boolean metrics (i.e., 0 and 1).

In the following, $Q(q)$ denotes the global constraint value for the q^{th} QoS attribute of the composite service specified by the user (e.g., $Q(cost) = 65$ indicates that the cost of the required service has to be less than 65 cost units). Note that since we consider only attributes with decreasing value direction, only upper bound QoS constraints are taken into account when dealing with global user constraints. In addition, the user may specify a weight for each QoS attribute q denoted by W_q to represent its preferences, s.t. $\sum_{q \in QS} W_q = 1$. Otherwise, all quality attributes are considered at the same level of preferences.

Business Level Constraints

A business process is usually defined by a set of activities (or abstract tasks) $A = \{A_1, \ldots, A_n\}$. In the following, we specify the constraints considered at the business level.

Structural Constraints

Usually, activities of each business process have structural dependencies between them. We assume that each business process is characterized by a single initial task and a single end task. For simplicity, only basic structures are considered. We denote by $Pd(A_i)$ the set of immediate predecessors of the activity $A_i \in A$. In this paper, we consider four structural patterns which are commonly adopted by existing composition approaches since they cover most of the structures specified by composition languages:

- *Sequence*: In this pattern, activities are executed in sequence. We denote by S the set of activities that belong to a sequence structure.
- *Parallel (AND)*: This pattern indicates that activities are executed in parallel. We denote by P the set of activities that belong to a parallel structure and SP the set of parallel structures. $P_l \in SP$ indicates the parallel structure number l.
- *Choice (OR)*: Choice pattern indicates that only one activity can be executed in each choice structure. We denote by C the set of activities that belong to a choice structure. In the following, SC indicates the set of choice structures and $C_l \in SC$ indicates the choice structure number l. For each choice structure $C_l \in SC$, p_{li} is the probability to execute the branch i s.t. $\sum_{i=1}^{nC_l} p_{li} = 1$ and nC_l is the number of disjoint branches in the choice structure C_l.
- *Loop*: Loop pattern indicates that activities can be executed in an iterative manner. In the following, we denote by L the set of activities that will be executed iteratively. We suppose that there is an upper bound for the number of iterations of each loop. The maximum number of iterations of each activity $A_i \in L$ is denoted by α_i. This number can be determined through history based estimation and from previous task executions.

QoS Constraints

Besides structural constraints, business designers can also specify a set of *local QoS constraints* at the business level denoted by QC. Local constraints define quality for a given task in the process. A QoS constraint $qc_i(TP, V, q) \in QC$ is characterized by the activity $A_i \in A$ concerned by the QoS constraint, the type of the constraint which can be either MIN to denote a lower bound constraint or MAX to denote an upper bound constraint (i.e., $TP \in$ {MIN, MAX}), a value V and a QoS parameter $q \in QS$. For example, $qc_i(MAX, vc, cost)$ indicates that the cost of the task A_i has to be less than vc cost units.

Temporal Constraints

In addition, temporal constraints can be associated to business processes. We distinguish between local and inter-task temporal constraints (Lanz, Kolb et al., 2013), (Cheikhrouhou, Kallel et al., 2013).

- *Local temporal constraints* relate to the start and the finish time of each task. We denote by *TC* the set of local temporal constraints. A temporal constraint $tc_i\ (TP,\ T) \in TC$ is characterized by the activity concerned by the temporal constraint (i.e., $A_i \in A$), a type $TP \in$ {must start on (MSO), must finish on (MFO), start no earlier than (SNET), finish no earlier than (FNET), start no later than (SNLT), finish no later than (FNLT)} and a time point *T*. For example, $tc_i\ (SNET,\ T)$ indicates that the task A_i must start no earlier than the time point *T*.
- *Inter-task temporal constraints* relate to temporal dependencies between tasks. They specify time lags between two directly or indirectly succeeding tasks to restrict the time span between them. The set of inter-task temporal constraints is denoted by *TD*.
  Each temporal dependency $td_{ij}\ (TP,\ D_{ij}^{\ min},\ D_{ij}^{\ max}) \in TD$ is characterized by a source and a destination tasks $A_i \in A$ and $A_j \in A$, a type $TP \in$ {start-to-start (SS), start-to-finish (SF), finish-to-start (FS), finish-to- finish (FF)} and a minimum and a maximum duration between the source and the destination tasks (i.e., A_i and A_j) denoted by $D_{ij}^{\ min}$ and $D_{ij}^{\ max}$ respectively. For instance, $td_{ij}\ (SS,\ vt_1,\ vt_2)$ indicates that the task A_j has to start no earlier than vt_1 time units and no later than vt_2 time units after the start of the task A_i.

This paper assumes that business processes are well-structured and that all constraints are verified (i.e., there are no conflicts between them).

Service Level Constraints

Apart from constraints specified at the business level, other constraints can also be defined at service level. Each activity A_i of the business process has a set S_i of potential candidate services. The potential services of an activity A_i are functionally equivalent and can be distinguished by their QoS values. In this paper, we assume that all services have the same QoS attributes.

As presented in previously, services have temporal constraints related to their availabilities (e.g., a service can be unavailable each day from 8PM to 11PM). Moreover, a service can associate temporal constraints to QoS attributes (i.e., time-dependent QoS). For example, a service guarantees a smaller response time from 7PM to 10PM. Each service $S_{ij} \in S_i$ is characterized by a set T_{ij} disjoint intervals during which it offers different QoS values. To capture QoS variations related to these time-dependent QoS, we introduce the notion of *timed instance* of candidate services. Each timed instance is associated to a time interval which specifies its start and finish times. We denote by S_{ijk} the k^{th} timed instance of the service S_{ij} corresponding to the time interval $T_{ijk} \in T_{ij}$. The boundaries of each time span T_{ijk} are denoted by $t_{ijk}^{\ min}$ *and* $t_{ijk}^{\ max}$. We denote by $Q(S_{ijk},\ q)$ the value of the q^{th} attribute offered by the service S_{ij} at the time span T_{ijk}.

In this paper, we do not consider any special scheme for QoS values and we suppose that time-dependent QoS models are defined by service providers. The specification of these models can be achieved using existing QoS prediction methods (Chen, Yang et al., 2011).

Utility Function

The value of a particular QoS attribute *q* for the composite service *CS* denoted by $Q(CS,\ q)$ is computed by the aggregation of the corresponding quality values of its components services. The aggregation function *Agg* depends on the categories of quality attributes (i.e., Additive, Average, Multiplicative and Max-Operator) and the structure of the business process (i.e., the involved structural patterns) (Sun and Zhao, 2012). Thus, $Q(CS,\ q) = Agg_{Ai \in A}\ (Q(A_i\ ,\ q))$ with $Q(A_i,\ q)$ denotes the value of the quality attribute *q* of the component service corresponding to the task A_i.

To evaluate the quality of the composite service based on the user preferences, we define a *utility function*. This latter is a normalized function whose values range over [0,1]. It enables the aggregation of the quality values of the service into a single value while considering user preferences in order to select the best services. Therefore, the utility of a composite service *CS* is computed as follows using Simple Additive Weighting method (SAW) (Yoon and Hwang, 1995):

$$U(CS) = \sum_{q \in QS} W_q * \frac{Q(q)^{max} - Q(CS,q)}{Q(q)^{max} - Q(q)^{min}} \quad (1)$$

Where $Q(q)^{max} = Agg_{Ai \in A}\ (Q(A_i\ ,\ q)^{max})$ and $Q(q)^{min} = Agg_{Ai \in A}\ (Q(A_i\ ,\ q)^{min})$ denote respectively the minimum and maximum aggregated values of the q^{th} quality attribute of *CS* with $Q(A_i\ ,\ q)^{max} = max\{Q(\ S_{ijk}\ ,\ q)\ |\ S_{ij} \in S_i\ ,\ T_{ijk} \in T_{ij}\}$ and $Q(A_i\ ,\ q)^{min} = min\{Q(\ S_{ijk}\ ,\ q)\ |\ S_{ij} \in S_i\ ,\ T_{ijk} \in T_{ij}\}$.

A solution to the selection problem is then a combination of concrete services (each service implements one abstract business task) that complies with business and service constraints and satisfies all global user's constraints while optimizing the overall utility.

SERVICE PRUNING

Intuitively, to select the best combination of services, all candidate services can be taken into account. However, this is impractical when the number of services increases since the time needed to solve the service selection problem becomes exponential. Moreover, while dealing with time-dependent QoS and QoS and temporal constraints, the number of possible solutions and decision variables increases even further. For instance, if we consider that there are 6 tasks in a business process, 500 candidate services for each task and two timed instances for each service, the number of possible combinations of services is $(2 * 500)^6$. However, not all services are pertinent candidates for the feasible solution. To overcome this problem, we propose a pruning approach to reduce the number of candidate services of each task and thus reducing the number of possible uninteresting combinations of services which are not relevant to the selection problem so that the optimal solution still be found. The aim is to avoid discarding any candidate service that might be part of a feasible solution.

To do so, we propose two space reduction techniques to compute *local thresholds* of each task: (1) *QoS constraints based pruning* and (2) *temporal constraints based pruning*. In the following, we detail how we measure thresholds using these two techniques.

QoS Constraints based Pruning

QoS based pruning strategy aims to compute local QoS thresholds for individual tasks for each QoS attribute $q \in QS$ that will serve as local upper bound constraints (since we consider only QoS attributes with decreasing value direction) such that the unsatisfaction of one of these constraints by a candidate service guarantees the unsatisfaction of the global constraints. In other words, if a service has at least one QoS value that does not satisfy a local threshold, all possible combinations of services that include it will violate global user constraints, and thus it is not worth considering and it can be pruned from the set of candidate services to narrow the search space.

In a previous work (Guidara, Chaari et al., 2014), we proposed a set of formulas to measure local thresholds of QoS attributes. However, the proposed formulas consider only sequence and parallel patterns and do not cater for local QoS constraints. To deal with these limitations, in this subsection, we propose to compute local thresholds based on constraint optimization model.

A local threshold $Q_{LT}(A_i, q)$ for the q^{th} attribute of the task A_i depends on the value required in the global user constraint $Q(q)$ and the minimum and maximum values of this QoS attribute (i.e., $Q(A_i, q)^{min}$ and $Q(A_i, q)^{max}$). The main idea is to compute for each task its maximum value (i.e., the worst case) considering the minimum quality values of all other tasks (i.e., their best cases) such that the global and local QoS constraints are satisfied. Computing these thresholds needs to consider both the structure of the business process and the distinctive characteristics of each QoS attribute.

In the following, we present our constraint optimization model to compute local thresholds for each business task. This model can be applied for generic business process structures that can contain sequential, parallel, choice and loop patterns taking into account the categories of quality attributes presented in the previous section. To deal with complex business structures, we propose to compute local thresholds in a recursive manner such that local thresholds of each complex structure will be considered as global constraints for the atomic structures it contains. The proposed model can then be applied recursively on each complex structure until each atomic task is associated to a set of local thresholds. It is presented as follows:

$$maximize \quad Q(A_m, q) - \sum_{A_i \in A, i \neq m} Q(A_i, q) \tag{2}$$

$$Agg_{A_i \in A}\left(Q(A_i, q)\right) \leq Q(q), \forall A_i \in A \tag{3}$$

$$\sum_{A_i \in C_l} p_{li} = 1, \forall C_l \in SC \tag{4}$$

$$p_{li} \in \{0,1\}, \forall A_i \in C \tag{5}$$

$$A_m \in C \Rightarrow p_{lm} = 1 \tag{6}$$

$$Q(A_i, q) \leq V, \forall qc_i(MAX, V, q) \in QC \tag{7}$$

$$Q(A_i, q) \geq V, \forall qc_i(MIN, V, q) \in QC \tag{8}$$

$$Q(A_i, q) \in [Q(A_i, q)^{min}, Q(A_i, q)^{max}], \forall A_i \in A \tag{9}$$

The first constraint presents the objective function. This function allows computing the maximum quality value of each activity A_m for the attribute $q \in QS$ while minimizing the quality values of all the other tasks A_i, $i \neq m$. The maximum quality value of each activity A_m will be considered as its local threshold (i.e., $Q_{LT}(A_m, q) = Q(A_m, q), \forall A_m \in A$). To guarantee that the global QoS constraints are satisfied, we define Constraint (3) which ensures that the aggregation of the QoS parameters of component services corresponds to the global QoS. Unlike existing approaches which consider only sequential flow, our approach allows computing local thresholds considering several composition structures. In Table 2, we define the aggregation function of the composite service according to the different categories of QoS attributes while exploring the structure of the business process. To deal with choice structures, we add constraints (4) and (5). Usually, the probability of each choice branch is a value in [0,1] and the value of a quality attribute in a choice structure is measured by the sum of the different execution paths multiplied by their probabilities. However, since we consider the best case of all tasks A_i, $i \neq m$ and the worst case of the task A_m, we need to consider only one branch for each choice structure independently on the probabilities of the different branches. For this reason, we suppose that p_{li} can be equal to 0 or 1 for each task in a choice structure and that only one path can be considered for each choice pattern (s.t., $p_{li} = 1$ if the path i of the choice structure C_l is selected and 0 otherwise). Constraint (6) indicates that if the activity for which we compute the local threshold (i.e., A_m) is in a choice structure, it should be considered as the path that

will be executed and thus, its probability is equal to 1. In addition, local QoS constraints have to be guaranteed. For instance, given our motivating example introduced previously where $Q(cost) = 65$. By applying the proposed model and using the aggregation function of additive attributes (Table 2), the local threshold of the task A_5 is $Q_{LT}(A_5, cost) = 65 - (8 + 10 + 12 + 12 + 6) = 17$ if we do not consider local QoS constraints. Hence, only the service instance S_{512} is pruned. However, if we consider the constraint QC_5 which states that the price of the 5th task must not exceed 11 Euros, this threshold will be equal to 11 and thus more services can be pruned (i.e., S_{531}). To deal with such constraints, we add Constraints (7) and (8). Finally, Constraint (9) indicates that the quality value of each task A_i belongs to the interval $[Q(A_i, q)^{min}, Q(A_i, q)^{max}]$, $\forall A_i \in A$.

Table 2. Aggregation function of the composite service according to the different types of QoS parameters

Category	Aggregation Function $Agg_{A_i \in A}(Q(A_i, q))$
Additive	$\sum_{A_i \in S} Q(A_i, q) + \sum_{A_i \in P} Q(A_i, q) + \sum_{C_l \in SC} \sum_{A_i \in C_l} p_{li}\, Q(A_i, q) + \sum_{A_i \in L} \alpha_i Q(A_i, q)$
Average	$\frac{1}{ns} * (\sum_{A_i \in S} Q(A_i, q) + \sum_{A_i \in P} Q(A_i, q) + \sum_{C_l \in SC} \sum_{A_i \in C_l} p_{li}\, Q(A_i, q) + \sum_{A_i \in L} \alpha_i Q(A_i, q))$
Multiplicative	$\prod_{A_i \in S} Q(A_i, q) * \prod_{A_i \in P} Q(A_i, q) * \prod_{C_l \in SC} \sum_{A_i \in C_l} p_{li} Q(A_i, q) * \prod_{A_i \in L} Q(A_i, q)^{\alpha_i}$
Max-Operator	$\sum_{A_i \in S} Q(A_i, q) + \sum_{P_l \in PS} \max_{A_i \in P_l} \{Q(A_i, q)\} + \sum_{C_l \in SC} \sum_{A_i \in C_l} p_{li}\, Q(A_i, q) + \sum_{A_i \in L} \alpha_i Q(A_i, q)$

ns: the number of component services

Temporal Constraints based Pruning

Although QoS constraints based pruning keeps for each task only candidate services that are likely to be a member of the optimal solution based on the different types of QoS attributes, some uninteresting services still need to be removed when taking into consideration time-dependent QoS attributes and temporal constraints between services. Considering temporal constraints is inevitable to satisfy business process owners' requirements but makes it difficult to prune uninteresting services. In the following, we detail how we consider these constraints to further reduce the number of candidate services. Thus, two main issues have to be considered: (1) the *execution duration* of each activity regarding the global duration required by the user, and (2) the *time spans* (i.e., start and finish times) of each activity with respect to the required deadline.

Execution Duration

The execution duration of each task can be calculated using the model specified in the previous section since the execution duration attribute can be considered as a Max-Operator attribute. Nevertheless, when dealing with temporal constraints specified in the business process, this model is not sufficient to effectively measure local thresholds of execution duration attribute since several temporal constraints can affect the values of the service duration and thus, further candidate services can be pruned.

To illustrate this, let's take the example introduced at the beginning of this paper. If we apply the QoS model, in this case, for instance, the local threshold of the second task is equal to 4 using the aggregation function in Table 2. However, if the task A_2 will be executed in 4 time units, the temporal dependency $TD_{1,4}$ will be violated and thus, services that offer an execution duration equal or greater than 4 time units

for the task A_2 should be discarded. Another example is that, based on the proposed QoS model, the local threshold of the task A_6 is equal to 4. Nevertheless, services that have an execution duration value equals to 4 time units for the 6th task cannot participate in the selection process if we consider the temporal dependency $TD_{5,6}$. In fact, even if the remaining tasks will be executed at their minimum allowed execution duration values and considering the minimum values of the temporal dependencies, the global constraint (i.e., the duration ≤ 13 time units) will be violated. Hence, considering a local threshold that equals to 3 rather than 4 time units for the task A_6 will lead to further elimination of uninteresting candidate services. This issue is more complex to resolve when handling business processes where several structural and temporal constraints exist between tasks. This is explained by the fact that some temporal dependencies may be overlapped or included in each other. Additionally, temporal constraints may have different types and thus should be resolved differently.

In our previous work (Guidara, Guermouche et al., 2014), we proposed a constraint optimization model to measure local thresholds of the execution duration attribute. We supposed that the start and finish times belong to the interval [0, *Q(dur)*] in order to guarantee that the global duration is not violated. This model cannot deal with complex business structures and do not consider QoS constraints at the business level. Moreover, in this model, only temporal dependencies between tasks are considered. Nevertheless, local temporal constraints can also affect the execution duration of the business process. Let's consider the temporal constraint TC_2 in our previous example. Given this constraint, and considering the service instances of the first and the second task, the duration of the task A_2 cannot be greater than 3 time units. In fact, at the best case, the execution of the first task will finish at 12AM given the candidate services presented in Figure 2. Thus, to guarantee that the temporal constraint TC_2 is fulfilled, the execution duration of the second task should be less than 3 time units. To overcome these limitations, we rely on a new constraint optimization model that allows computing local maximum duration of each business task while taking into consideration complex structural dependencies in addition to local and inter-task temporal constraints. This model is applied for each task $A_m \in A$:

$$maximize \quad Q(A_m, dur) - \sum_{A_i \in A, i \neq m} Q(A_i, dur) \tag{10}$$

$$ft_n - st_1 \leq Q(dur) \tag{11}$$

$$ft_i = st_i + Q(A_i, dur), \forall A_i \notin L \tag{12}$$

$$ft_i = st_i + \alpha_i * Q(A_i, dur), \forall A_i \in L \tag{13}$$

$$\sum_{A_i \in C_l} p_{li} = 1, \forall C_l \in SC \tag{14}$$

$$p_{li} \in \{0,1\}, \forall A_i \in C \tag{15}$$

$$A_m \in C \Rightarrow p_{lm} = 1 \tag{16}$$

$$ft_i \leq st_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \notin C \tag{17}$$

$$\sum_{A_i \in C_l} p_{li} * ft_i \leq st_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \in C_l \tag{18}$$

$$Q(A_i, dur) \leq V, \forall qc_i(MIN, V, dur) \in QC \tag{19}$$

$$Q(A_i, dur) \geq V, \forall qc_i(MIN, V, dur) \in QC \tag{20}$$

$$st_i = T, \forall tc_i(MSO, T) \in TC \tag{21}$$

$$ft_i = T, \forall\, tc_i(MFO, T)\ \in\ TC \tag{22}$$

$$st_i \geq T, \forall\, tc_i(SNET, T)\ \in\ TC \tag{23}$$

$$ft_i + D_{ij}^{min} \leq st_j\ , \forall\, td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max})\ \in\ TD \tag{24}$$

$$st_j\ \leq ft_i + D_{ij}^{max}\ , \forall\, td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max})\ \in\ TD \tag{25}$$

$$Q(A_i\ , dur)\ \in [Q(A_i\ , dur)^{min}\ , Q(A_i\ , dur)^{max}], \forall\, A_i\ \in\ A \tag{26}$$

$$st_i, ft_i\ \in [min_{k \in T_{ijk}}\{t_{ijk}^{min}\}, max_{k \in T_{ijk}}\{t_{ijk}^{max}\}], \forall\, A_i \in A \tag{27}$$

The objective function of the proposed model allows computing for each task $A_m \in A$ its maximum duration while minimizing the duration of all other tasks. Then, the local threshold of the task A_m is $Q_{LT}(A_m, dur) = Q(A_m, dur)$. The computation of local thresholds must ensure that structural and temporal constraints are still satisfied. To guarantee that the global constraint of the duration attribute is satisfied, we propose constraint (11). Constraints (12) and (13) specify the relation between the start and the finish times of each task considering both non-loop and loop tasks. To guarantee that all loop structures are verified, here we consider that all preselected services of loop tasks must satisfy the maximum number of iterations. Similarly to the constraints from (4) to (6) specified in the previous section, Constraints from (14) to (16) handle choice structures. To deal with *structural constraints*, we add Constraints (17) and (18). Constraints (19) and (20) allow handling *QoS constraints* related to the execution duration attribute. To deal with *local temporal constraints*, we propose constraints from (21) to (23). For simplicity, we only present the temporal constraints (MSO, MFO and SNET). For example, the constraint (21) ensures that for each constraint of the form Must Start On T, the earliest start time of the corresponding task is equal to the time point T. To deal with *inter-task temporal constraints* (i.e., temporal dependencies), we propose the Constraints (24) and (25). For simplicity, only end-to-start temporal dependencies are considered since other dependencies can be defined by the same manner. For instance, the constraint (24) aims to ensure that once the finish time of an activity A_i arises, the activity A_j can begin at the earliest time after the minimal duration value D_{ij}^{min} of the inter-task temporal constraint td_{ij}. Finally, the Constraints (26) and (27) indicate respectively the domains of the duration and the start and the finish time of each business task. By applying this model on the motivating example introduced previously, the obtained maximum execution duration values of the tasks are respectively 4, 3, 2, 5, 7, 3 and 3.

Time Intervals

The selection of the best solution when dealing with both time-dependent QoS and temporal constraints of business tasks needs the specification of the start and finish time of each service. Nevertheless, the start time of each service is affected by the start times of its predecessors. Thus, selecting a wrong start time for one service can lead to several wrong choices for start times of its successor services which decreases the performance of the selection process. If for example the service instance S_{121} (Figure 2) has been selected since it offers the best cost and the minimum execution duration. Therefore, only the service instance $S_{2'12}$ can be selected for the task $A_{2'}$. This solution does not fulfill the required user constraints.

To avoid possible unnecessary combinations, we need to reduce the number of start and finish times to consider. To do this, we propose to compute the largest time span of each task based on the deadline required by the user while satisfying structural and temporal constraints. The boundaries of these time intervals (i.e., the earliest start and the latest finish time) will be considered as *local temporal thresholds* of each task such that all service instances whose time intervals do not belong to the computed time spans will be pruned. Two possible strategies can be applied forward (i.e., based on start times) and backward

(i.e., based on finish times) (Eder, Panagos et al., 1999). Nevertheless, the specification of the largest time spans of business tasks is a very hard problem when dealing with complex business processes.

To overcome this issue, we propose a constraint optimization model that computes the minimum start time and the maximum finish time of each business task so that all structural and temporal constraints are fulfilled and the deadline of the entire process is respected. To ensure that the local thresholds do not exclude any candidate service that can be part of a feasible solution, we propose the following constraint optimization model. The proposed model extends our previous model presented in (Guidara, Guermouche et al., 2014) to handle complex business structures and to eliminate further inappropriate services. It is executed for each task $A_m \in A$ to compute its time interval.

$$maximize \;\; ft_m - st_m \tag{28}$$

$$ft_i = st_i + Q(A_i, dur), \forall A_i \notin L \tag{29}$$

$$ft_i = st_i + \alpha_i * Q(A_i, dur), \forall A_i \in L \tag{30}$$

$$ft_n \leq deadline \tag{31}$$

$$st_i + Q(A_i, dur)^{min} \leq st_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \notin C \tag{32}$$

$$ft_i + Q(A_j, dur)^{min} \leq ft_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \notin C \tag{33}$$

$$min_{A_i \in C_l}\{st_i + Q(A_i, dur)^{min}\} \leq st_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \in C_l \tag{34}$$

$$min_{A_i \in C_l}\{ft_i + Q(A_j, dur)^{min}\} \leq ft_j, \forall A_j \in A, A_i \in Pd(A_j), A_i \in C_l \tag{35}$$

$$st_i = T, \forall tc_i(MSO, T) \in TC \tag{36}$$

$$ft_i = T, \forall tc_i(MFO, T) \in TC \tag{37}$$

$$st_i \geq T, \forall tc_i(SNET, T) \in TC \tag{38}$$

$$st_i + Q(A_i, dur)^{min} + D_{ij}^{min} \leq st_j, \forall td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max}) \in TD \tag{39}$$

$$st_j \leq st_i + Q(A_i, dur)^{min} + D_{ij}^{max}, \forall td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max}) \in TD \tag{40}$$

$$ft_i + Q(A_j, dur)^{min} + D_{ij}^{min} \leq ft_j, \forall td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max}) \in TD \tag{41}$$

$$ft_j \leq ft_i + Q(A_j, dur)^{min} + D_{ij}^{max}, \forall td_{ij}(FS, D_{ij}^{min}, D_{ij}^{max}) \in TD \tag{42}$$

$$Q(A_i, dur) \in [Q(A_i, dur)^{min}, deadline], \forall A_i \in A \tag{43}$$

$$st_i \in [min_{k \in T_{ijk}}\{t_{ijk}^{min}\}, max_{k \in T_{ijk}}\{t_{ijk}^{max} - Q(S_{ijk}, dur)\}], \forall A_i \in A \tag{44}$$

$$ft_i \in [min_{k \in T_{ijk}}\{t_{ijk}^{min} + Q(S_{ijk}, dur)\}, max_{k \in T_{ijk}}\{t_{ijk}^{max}\}], \forall A_i \in A \tag{45}$$

The first constraint allows computing the largest time intervals of each business task A_m (i.e., maximize the distance between the start time *st* and the finish time *ft*). Constraints (29) and (30) indicate the relation between the start and the finish time of each task. To guarantee that the deadline is not violated, we add Constraint (31). To deal with structural dependencies, we propose Constraints (32) and (33) which guarantee that the earliest start time of each task A_j occurs after the earliest start time and the minimum duration of each of its predecessor tasks. In addition, the latest finish time of each task A_j has to be greater than or equal to the sum of its minimum execution duration and the latest finish time of all its predecessor

tasks. Since we consider the largest time interval of each task, we suppose that the start time and the finish time of each task should be greater than the minimum start and finish times of its predecessors if these latter belong to a choice structure (34) and (35). Furthermore, it is vital to check if temporal constraints are satisfied when computing the largest time interval of each task. To deal with *local and inter task temporal constraints*, we propose Constraints from (36) to (38) and from (39) to (42) respectively. As previously, for simplicity, we present only a limited set of temporal constraints. The duration of each task A_i should be within the interval $[Q(A_i, dur)^{min}, deadline]$ (43). Finally, the last two constraints show the domains of the start and the finish time of each task $A_i \in A$.

A solution of the optimization problem is then a set of the largest possible time intervals of all tasks. For instance, given the example presented in Figure 1, by applying our model, the largest time slots of all tasks when considering all constraints and with a deadline equals to 23 are respectively: [8,14], [10,15], [10,15], [11,19], [11,19], [15,21] and [18,23]. Based on these intervals some service instances have to be pruned (e.g., S_{121} and S_{332}) or some restrictions have to be performed to their intervals (e.g., S_{221}).

Pruning Algorithm

After explaining how to compute the local thresholds of each task based on QoS and temporal constraints, in this section, we present our pruning algorithm. We note that if at least one local QoS or temporal threshold cannot be computed (i.e., there is no solution for the optimization problem), we conclude that the service selection problem has no feasible solutions. The pruning steps are given by the Algorithm 1.

Algorithm 1 Service Pruning Algorithm

Require: The set of candidate services of each task
Ensure: The set of preselected services of each task

```
1:  for each A_i ∈ A do
2:    for each q ∈ QS do
3:      Q_LT (A_i, q) ← ComputeQoSThresholds(A_i, q)
4:    {st_i , ft_i} ← ComputeTemporalThresholds(A_i)
5:  for each A_i ∈ A do
6:    for each S_ij ∈ S_i do
7:      for each T_ijk ∈ T_ij do
8:        for each q ∈ QS do
9:          if Q(S_ijk , q) > Q_LT (A_i, q) then
10:           S_i ← S_i \ {S_ijk}
11:           break
12:         if [t_ijk^min , t_ijk^max] ∩ [st_i , ft_i] = ø then
13:           S_i ← S_i \ {S_ijk}
14:           break
15:         else {[t_ijk^min , t_ijk^max] ∩ [st_i , ft_i] = [X,Y] }
16:           if Y - X < Q(S_ijk , dur) then
17:             Si ← Si \ {S_ijk}
18:             break
19:           else
20:             if t_ijk^min < X then
21:               t_ijk^min ← X
22:             if t_ijk^max > Y then
23:               t_ijk^max ← Y
24:   if S_i = ø then
25:     There is no solution
```

The proposed algorithm takes as inputs the set of all available services of each task and returns the set of services which are likely to be candidate of the optimal solution (i.e., they do not violate any of the local thresholds of their corresponding task). The steps we follow in Algorithm 1 can be specified as follows. First, we compute local QoS and temporal thresholds based on the models presented in the previous sections (line 1 to 4). The second step is to prune services based on QoS thresholds (line 9 to 11) and then based on time spans (line 12 to 23). If a local threshold is violated, it is not worth to check the fulfillment of other thresholds and the service should be removed from the set of available services. If all QoS thresholds are verified, we compare the time span of each timed service instance with the interval of its corresponding task. If the intersection between these two intervals is empty or it does not cover the duration of the service instance, this instance should be eliminated (line 12 to 18). Otherwise, the time span of the service instance should be restricted to the span of its task (line 19 to 23). Finally, if at least one task does not have any candidate service, the selection problem has no feasible solutions (lines 24 and 25).

SERVICE SELECTION

Once the pruning process is fulfilled and the relevant candidate services are identified, we proceed to the selection of the best service combination. To do so, we model the selection problem as a constraint optimization problem. When dealing with time-dependent QoS values, determining the start and finish times of each service S_{ij} is crucial, since by delaying the execution of a service, some QoS attributes can be modified. Thus, in the optimization phase, two types of decision variables are taken into account. The first one is to select a concrete service for each atomic task and the second one is to determine a valid starting time for each selected service in order to match the global constraints.

The proposed model selects exactly one atomic service of each abstract task with the corresponding start and finish times while optimizing the overall utility and satisfying all constraints. Hence, in addition to QoS values, two temporal values are specified for each task: the start time *st* and the finish time *ft*.

Usually, there is no single combination of services that dominates all other solutions in terms of QoS values. To select the best solution, we use Simple Additive Weighting method to aggregate QoS values into a single utility value. Thus, the selection of the best combination of services can be achieved by optimizing the utility of the composite service. Therefore, the objective function of our optimization model is as follows:

$$maximize \quad \sum_{q \in QS} W_q * \frac{Q(q)^{max} - Q(CS,q)}{Q(q)^{max} - Q(q)^{min}} \tag{46}$$

Such that for each $q \in QS$:

$$Q(CS,q) = Agg_{A_i \in A}\Big(\sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} * Q(S_{ijk}, q) \Big) \tag{47}$$

Since the aggregation function depends on the quality attribute and the structure of the business process, we propose to use the aggregation functions presented in Table 2. We note that in this step, only preselected services after the pruning step are considered. Thus, the minimum and maximum values of each attribute (i.e., $Q(q)^{min}$ and $Q(q)^{max}$) have to be recomputed to consider only preselected services of each task with $Q(A_i, q)$ belongs to the interval $[Q(A_i, q)^{min}, Q(A_i, q)^{max}]$, $\forall A_i \in A$. To guarantee that only one service will be selected for each task we define the following constraint:

$$\sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} \ = 1, \forall\, A_i \in A, a_{ijk} \in \{0,1\} \tag{48}$$

With a_{ijk} =1 if the service S_{ijk} is selected and 0 otherwise. Since all global constraints have to be satisfied when selecting the optimal solution, we add the Constraint (49):

$$Q(CS, q) \leq Q(q), \forall\, q \in QS \tag{49}$$

Constraint (50) allows selecting only one service for each choice structure with $p_{li} \in \{0,1\}, \forall\, A_i \in C$.

$$\sum_{A_i \in C_l} p_{li} = 1, \forall\, C_l \in SC \tag{50}$$

Moreover, we should ensure that the start and finish times of each task belong to the time span of the same selected service instance. For this, for each task $A_i \in A$ we propose the following constraints:

$$\sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} * \ t_{ijk}^{min} \ \leq \ st_i \,, \forall\, A_i \in A \tag{51}$$

$$st_i \leq \sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} \ * \ (t_{ijk}^{max} - \ Q(S_{ijk} \,, dur)), \forall\, A_i \in A \tag{52}$$

To specify the relation between the start and finish times of each task A_i, we add these Constraints:

$$ft_i = st_i + \sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} * \ Q(S_{ijk} \,, dur) \,, \forall\, A_i \notin L \tag{53}$$

$$ft_i = st_i + \alpha_i * \sum_{S_{ij} \in S_i} \sum_{T_{ijk} \in T_{ij}} a_{ijk} * \ Q(S_{ijk} \,, dur) \,, \forall\, A_i \in L \tag{54}$$

The start and finish times of each task A_i (i.e., *sti* and ft_i) belong to the interval $[min_{k \in T_{ijk}}\{t_{ijk}^{min}\}, max_{k \in T_{ijk}}\{t_{ijk}^{max}\}]$. To check the satisfaction of structural dependencies, we use constraints (17) and (18). Finally, to check the satisfaction of inter-task temporal constraints, we use constraints (24) and (25). Note that it is not worth to check the satisfaction of local QoS and temporal constraints since we consider only preselected services.

PERFORMANCE EVALUATION

To evaluate the effectiveness of our pruning based service selection approach, we have conducted performance evaluation experiments.

Experiment Settings

These experiments have been performed on a laptop with a 32 bit Intel Core 2.20 GHz CPU and 4GB RAM and Windows 7 as operating system. To implement the proposed constraint optimization models,

we used the open source constraint solver Choco 2.1.5 [1] and Java 1.6. Candidate services for each abstract task are generated randomly based on given minimum and maximum values for QoS attributes and time availabilities (i.e., start and finish time). Further, in this study, all local and global constraints were randomly chosen. In all test cases, we consider one local QoS constraint, one local temporal constraint and one inter-task temporal constraint since these constraints can positively affect the computation time. For simplicity, all QoS attributes are assumed to have the same weights. Time-dependent QoS attributes for each service instance were generated with a uniform distribution for a time horizon with 150 time points and distributed in the range between 1 and 150. In these experiments, we consider the four categories of QoS attributes described in the previous sections. The computation time of each process was averaged over 50 randomly generated problem instances.

Performance Results

In the following, we present a comparison of the performance of different test cases regarding the execution time of two main processes: the pruning process and the selection process.

Performance of the pruning process

The pruning process includes the computation of QoS and temporal thresholds and the elimination of uninteresting services before selecting the best combination of services. It is worth noting that the execution time needed to eliminate inadequate services is approximately negligible compared to the execution time of the thresholds computation process. Thus, the time of the pruning process is equal to that of computing local thresholds. Hence, hereafter, we only present the time needed to compute local thresholds.

Since local thresholds of QoS attributes and time intervals (i.e., the minimum start and the maximum finish time) of each task can be computed independently from each other, we propose that the computation of all thresholds can be done in parallel in order to reduce the execution time. Then, this latter can be measured by the maximum of all time values of computing the different thresholds. We note that the computation time does not change when the number of services changes. This is obvious since the computation of the local QoS and temporal thresholds does not depend on the number of candidate services per task. In fact, it only depends on the domains of their QoS values and time intervals (i.e., their minimum and maximum values). Since these values are independent of user requirements, they can be determined at the design time and then can be continuously updated in response to the environment changes (e.g., the addition of a new service, the change of quality values of one or more services).

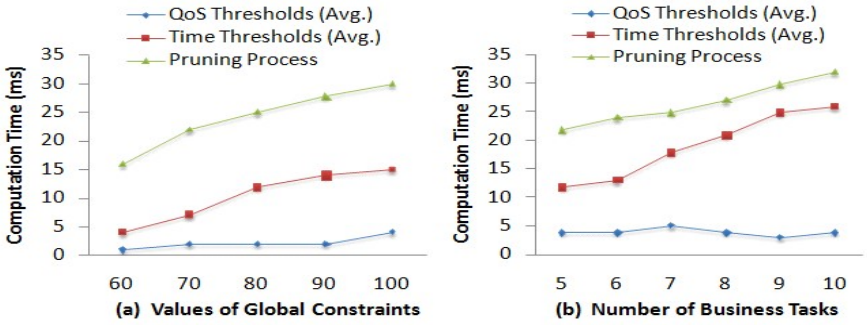


Fig. 3. Computation time of the pruning process

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

Furthermore, the execution time does not depend on the number of global constraints. This is due to the fact that local thresholds are computed in parallel. Nevertheless, the execution time of this process mainly depends on the domains of QoS values and time intervals, the required global constraints and the number of abstract business tasks. Hence, in Figure 3, we present the time required to compute thresholds in response to the values of global constraints and the number of business tasks. For each case, we present the average time needed to compute QoS and temporal thresholds and the execution time of the thresholds computation process (i.e., the maximum time required to measure all thresholds).

Results displayed in Figure 3 show that in all cases the time needed to compute local thresholds is dominated by the computation of local temporal thresholds. This is due to the fact that the number of decision variables is bigger than those used to compute QoS thresholds. First, in Figure 3 (a), we present the computation time while the values of global constraints vary from 60 to 100 and the number of tasks and global constraints are fixed to 5 and 4 respectively. The computation time increases when the values of global constraints increase. This is explained by the fact that the number of possible values of local thresholds becomes bigger. Second, Figure 3 (b) presents the execution time when the number of business tasks varies from 5 to 10. The number of global constraints is equal to 5 and their values vary from 80 to 120. Results indicate that the computation time increases very slowly when the number of tasks increases because the number of decision variables increases as well in this case. Experiments show that in both cases, the time needed to compute local thresholds is very small (32 ms in the worst case) while dealing with large domains for all variables. This allows our approach to be used in large problems where QoS and temporal values can be set in very large intervals with a negligible overhead that does not affect the efficiency of the selection process.

Performance of the pruning process

In what follows, we compare the computation time of the selection process with respect to the number of services, global constraints and abstract business tasks obtained in two cases: (1) *With Pruning*: that presents the process of pruning uninteresting services and selecting the best solution (the time of this process is equal to the sum of the times of the pruning and the selection processes). Here we notice that the computation time of the pruning process is very negligible compared to the time of the selection process. (2) *Without Pruning*: that presents the selection process using the basic selection algorithm where no pruning techniques are applied.

- *Performance vs Number of Services*:

First, experiments were conducted in relation to the number of candidate service instances per task that varies from 200 to 800 with 5 business tasks and 5 global constraints. The results provided in Figure 4 (a) indicate that applying the pruning process significantly outperforms the basic algorithm. Although the computation time of the basic algorithm is close to the time of our approach when the number of services is small (around 200 service instances), the computation time of our approach increases very slowly compared to the basic one by increasing the number of candidate services per task.

Figure 4 (b) shows the computation time required to find a feasible solution. In these experiments, we consider the average time of the selection process with and without pruning as well as the maximum time required to find a feasible solution based on our pruning process. The results indicate that the performance of the selection process increases significantly when applying our search space reduction mechanisms prior to the selection process compared to the basic algorithm. In fact, the maximum time needed to search a feasible solution based on our approach is less than the average time to search a feasible solution based on the basic selection algorithm in most test cases. Thus, our approach can find a feasible solution in a very small time even though the number of candidate services per task is very large. Hence, it can be used in practical applications when a feasible solution has to be selected in a very short time.

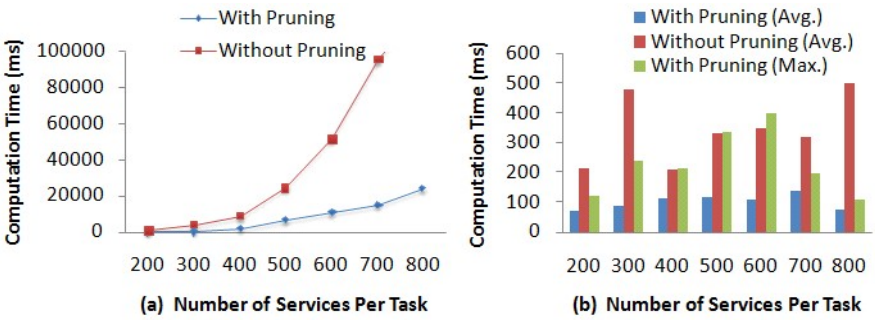


Fig. 4. Computation time of the selection process with respect to the number of candidate services per task

- *Performance vs Number of Constraints*:

Figure 5 (a) shows the computation time of the selection algorithms when the number of global constraints varies between 5 and 10 and the number of tasks is fixed to 5 with 200 candidate service instances for each task. As before, while the computation time of the basic algorithm significantly increases due to the fact of the growing number of optimal instances that should be compared, our algorithm increases very slowly and leads to better performance even when the number of constraints is high. This is an expected behavior since by considering several global constraints, the number of feasible solutions decreases. Hence, more services are likely to violate one or more constraints and thus they should be pruned.

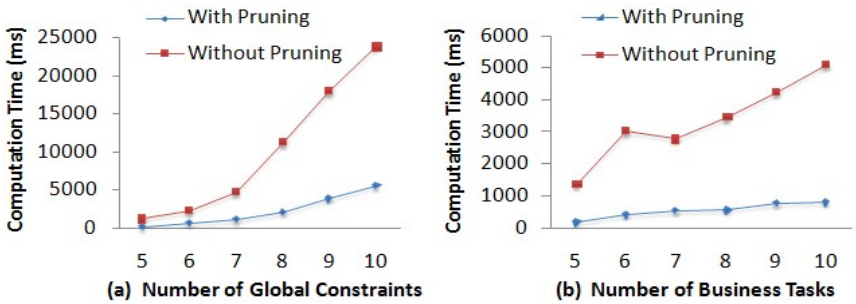


Fig. 5. Computation time of the selection process with respect to the number of global constraints and the number of business tasks

- *Performance vs Number of Tasks*:

Finally, to further compare the performance of our pruning approach, in Figure 5 (b) we give experimental results based on the computation time with respect to the number of business tasks. The number of candidate service instances of each task is fixed to 200 and the number of global constraints is fixed to 5. Business tasks depend on each other with several complex structures including different composition patterns. Again, results are the same as previously and the optimal solution can be found rather fast when applying our pruning approach.

In all the test cases, the results show a considerable gain in performance when applying the pruning approach which scales better than the traditionally algorithm where all candidate services are considered. The performance of our approach is significant particularly in the context of complex selection problems where the number of candidate services, tasks and constraints can be huge.

RELATED WORK

QoS based service selection problem has been increasingly investigated in multiple domains. To solve this problem, some works adopt exhaustive methods to find the best assignment of services to abstract tasks. In (Zeng, Benatallah et al, 2004), Zeng et al. use mixed linear programming techniques to select the optimal component services for the composition and to achieve global optimization of QoS attributes. This work is extended in Ardagna et al. (Ardagna and Pernici, 2007) to include local constraints and loop peeling to deal with composition structures with cycles. In (Hassine, Matsubara et al., 2006), Ben Hassine et al. discuss a web service composition approach using constraint programming. Nevertheless, these approaches do not cater for temporal properties. Moreover, these methods are only suitable for small problems, as the complexity of the selection algorithm increases exponentially when the problem size increases. This causes several scalability issues especially when dealing with a large number of candidate services for each task.

Some researchers adopt approximate methods and propose heuristics that can be used to find a near- to-optimal solution more efficiently than exact solutions. Yu et al. (Yu, Zhang et al., 2007) introduce two alternative models for the QoS-based service composition problem: the combinatorial model to define the problem as a multi- dimension multi-choice knapsack problem (MMKP) and the graph model to define the problem as a multi- constraint optimal path (MCOP) problem. Based on these models, the authors propose heuristic algorithms to achieve better performance. In (Canfora, Penta et al, 2005), Canfora et al. model and resolve the service selection problem based on genetic algorithms. In contrast to our approach, these approaches, do not provide strategies to reduce the search space before selection, leading to potential inefficiencies due to the consideration of uninteresting candidate services especially when the tasks and candidate services are numerous. Moreover, they do not consider temporal properties.

To reduce the computational time of service selection algorithms, an alternative proposal is to narrow the search space. Some works reduce services based only on functional properties (Oster, Santhanam et al., 2011) and thus they cannot be applied in QoS aware service selection problems. Other proposals apply the decomposition techniques to decompose global constraints into local ones and then, reduce the number of candidate services. For instance, Alrifai et al. (Alrifai, Risse et al., 2012) use a mixed integer programming model to compute local constraints of each task based on a set of QoS levels for each QoS attribute. After that, a local selection strategy is applied to select the best service for each task. As a step forward, Qi et al. (Qi, Tan et al., 2010) suggest a local optimization method to further reduce the number of candidate services based on QoS levels and enumeration. In (Sun and Zhao, 2012), 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 (Mardukhi, Nematbakhsh et al., 2013) to define local constraints using genetic algorithm. Although the proposed solutions scale better when dealing with large problems, they rely on greedy pruning methods when computing local constraints that can affect the ability to find an optimal solution. Additionally, these works are not able to handle time- dependent QoS attributes associated with temporal constraints and dependencies between services.

In (Huang, Jiang et al., 2009) Huang et al. propose pruning techniques to reduce the number of services to be considered. First, they remove services that cannot be executed (i.e., they have no inputs). Then, they eliminate services that cannot offer optimal QoS values. Another pruning process is proposed in (Mabrouk, Beauche et al., 2009) to prune services using (1) incremental computation when dealing with a large number of activities and (2) utility values approximation when handling a large number of candidate services per task. Barakat et al. (Barakat, Miles et al., 2011) apply two space reduction techniques to reduce the number of candidate services and the number of alternative abstract plans. Authors use the notion of dominance to select representative services for each task without affecting the optimal solution. Unlike our approach, these works do not allow the computation of local thresholds of each task.

Moreover, they cannot be applied when dealing with time-dependent QoS attributes since non dominated services may have different temporal properties that prevent a possible collaboration between them.

Temporal properties have been considered by some works when composing several services (Guermouche and Godart, 2014). Zhang et al. (Zhang, Ma et al., 2013) take into consideration the dynamic aspect of QoS attributes to compose services in multi- domain environments. In (Liang, Du et al, 2013), Liang et al. propose a penalty-based genetic algorithm to select services under temporal constraints. However, authors assume that QoS values do not depend on the time of the execution and only upper bound constraints between activities are considered. To deal with time-dependent QoS values, Wagner et al. (Wagner, Klein et al., 2012) propose a service selection approach with time and input-dependent QoS attributes. Authors define a multi-objective optimization based approach that selects the best combination of services while specifying the start and finish time of each service instance according to the values of QoS attributes at each time period. In (Klöpper, Ishikawa et al., 2010), Klöpper 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 (Wagner, Klein et al., 2012) and (Klöpper, Ishikawa et al., 2010) do not consider temporal constraints at the business level and do not provide any search space reduction techniques prior to the selection process.

CONCLUSION

In this paper, we have tackled the problem of service selection for business processes. Unlike existing works, we cater for time-dependent QoS attributes associated with complex business constraints. This is an important step towards the consideration of complex business models and service offers in practical applications. To handle a large number of services, an efficient service pruning approach is proposed and implemented to reduce the number of candidate services to be considered prior to the selection process while ensuring that the optimal solution still be found. After that, an optimization algorithm has been applied based on a constraint optimization model. The proposed algorithm allows the selection of the best combination of services with general composition structures including sequential, parallel, choice and loop patterns while considering complex QoS and temporal constraints.

Experimental results show that although our approach significantly enhances the computational time, it remains suitable in the context of static environments. To tackle the selection problem in dynamic environments, in our future work, we aim to define and implement an approximate selection approach to search for a near to optimal solution when dealing with very huge number of services. We also plan to address further possible correlations between quality attributes of candidate services (i.e., inter-task QoS dependencies) and more complex time-based QoS change cycles and patterns.

REFERENCES

S.-Y. Hwang, E.-P. Lim, C.-H. Lee, and C.-H. Chen, (2008). Dynamic web service selection for reliable web service composition, *IEEE T. Services Computing*, vol. 1, no. 2, pp. 104–116.

M. Alrifai, T. Risse, and W. Nejdl, (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints, *ACM Trans. the Web*., vol. 6, no. 2, p. 7.

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

D. Ardagna and B. Pernici, (2007). Adaptive service composition in flexible processes, *IEEE Trans. Software Eng*., vol. 33, no. 6, pp. 369–384.

A. B. Hassine, S. Matsubara, and T. Ishida, (2006). A constraint- based approach to horizontal web service composition, in *International Semantic Web Conference*, pp. 130–143, November 5-9, USA.

T. Yu, Y. Zhang, and K.-J. Lin, (2007). Efficient algorithms for web services selection with end-to-end qos constraints, *ACM Trans. the Web*., vol. 1, no. 1.

G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, (2005). An approach for qos-aware service composition based on genetic algorithms, in *GECCO*, pp. 1069–1075, July 8-12, USA.

M. Alrifai and T. Risse, (2009). Combining global optimization with local selection for efficient qos-aware service composition, in *WWW*, pp. 881–890, April 20-24, Spain.

L. Qi, Y. Tang, W. Dou, and J. Chen, (2010). Combining local optimization and enumeration for qos-aware web service composition, in *ICWS*, pp. 34–41, July 4-9, USA.

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

F. Mardukhi, N. Nematbakhsh, K. Zamanifar, and A. Barati, (2013). Qos decomposition for service composition using genetic algorithm, *Appl. Soft Comput*., vol. 13, no. 7, pp. 3409–3421.

S. Cheikhrouhou, S. Kallel, N. Guermouche, and M. Jmaiel (2014). Enhancing formal specification and verification of temporal constraints in business processes, in *SCC*, pp. 701–708, June 27- July 2, USA.

L. Chen, J. Yang, and L. Zhang, (2011). Time based qos modeling and prediction for web services, in *ICSOC*, pp. 532–540, December 5-8, Cyprus.

F. Wagner, A. Klein, B. Klöpper, F. Ishikawa, and S. Honiden, (2012). Multi-objective service composition with time- and inputdependent qos, in *ICWS*, pp. 234–241, June 24-29, USA.

B. Klöpper, F. Ishikawa, and S. Honiden, (2010). Service composition with pareto-optimality of time-dependent qos attributes, in *ICSOC*, pp. 635–640, December 7-10, USA.

O. Martín-Díaz, A. R. Cortés, A. Durán, and C. Müller, (2005). An approach to temporal-aware procurement of web services, in *ICSOC*, pp. 170–184, December 12-15, The Netherlands.

A. Lanz, J. Kolb, and M. Reichert, (2013). Enabling personalized process schedules with time-aware process views, in *CAiSE Workshops*, pp. 205–216, July 17-21, Spain.

H. Liang, Y. Du, and S. Li, (2013). An improved genetic algorithm for service selection under temporal constraints in cloud computing, in *WISE (2)*, pp. 309–318, October 13-15, China.

S. Cheikhrouhou, S. Kallel, N. Guermouche, and M. Jmaiel, (2013). Toward a time-centric modeling of business processes in bpmn 2.0, in *iiWAS*, pp. 154, December 2-4, Austria.

J. Eder, E. Panagos, and M. Rabinovich, "Time constraints in workflow systems," in *CAiSE*, 1999, pp. 286–300, June 14-18, Germany.

Z. J. Oster, G. R. Santhanam, and S. Basu, (2011). Identifying optimal composite services by decomposing the service composition problem, in *ICWS*, pp. 267–274, USA.

L. Barakat, S. Miles, I. Poernomo, and M. Luck, (2011). Efficient multi-granularity service composition, in *ICWS*, pp. 227–234, July 4-9, USA.

T. Zhang, J. Ma, C. Sun, Q. Li, and N. Xi, (2013). Service composition in multi-domain environment under time constraint, in *ICWS*, pp. 227–234, June 27-July 2, USA.

N. Guermouche and C. Godart, (2014). Composition of web services based on timed mediation. *IJNGC*, 5(1).

B. Benatallah, Q. Z. Sheng, A. H. H. Ngu, and M. Dumas, (2002). Declarative composition and peer-to-peer provisioning of dynamic web services, in *ICDE*, pp. 297–308, February 26-March 1, USA.

J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, and K. Kochut, (2004). Quality of service for workflows and web service processes, *J. Web Sem*., vol. 1, no. 3.

N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, (2009). Qos-aware service composition in dynamic service oriented environments, in *Middleware*, pp. 123–142, November 30- December 4, USA.

I. Guidara, T. Chaari, and M. Jmaiel, (2014). An efficient service selection approach with time-dependent qos, in *WETICE*, pp. 320–325, June 23-25, Italy.

H. Ma, F. Bastani, I.-L. Yen, and H. Mei, (2013). Qos-driven service composition with reconfigurable services, *IEEE T. Services Computing*, vol. 6, no. 1, pp. 20–34.

K. . P. Yoon and C.-L. Hwang, (1995). *Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences)*. Sage Publications.

I. Guidara, N. Guermouche, T. Chaari, S. Tazi, and M. Jmaiel, (2014). Pruning based service selection approach under qos and temporal constraints, in *ICWS*, pp. 9–16, June 27- July 2, USA.

Z. Huang, W. Jiang, S. Hu, and Z. Liu, (2009). Effective pruning algorithm for qos-aware service composition, in *CEC*, pp. 519–522, May 18-21, Norway.

W. Li, X. Li, X. Jun Liang, and X. Zhou, (2011). Qos-driven service composition with multiple flow structures, in *SCC*, pp. 362–369, July 4-9, USA.

M. C. Jaeger, G. Rojec-Goldmann, and G. Mühl, (2004). Qos aggregation for web service composition using workflow patterns, in *EDOC*, pp. 149–159, September 20-24, USA.