Pruning based Service Selection Approach under QoS and Temporal Constraints

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

Pruning based Service Selection Approach under QoS and Temporal Constraints

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*—Dynamic selection of the best services to execute abstract tasks of business processes is very important. Indeed, it enables to cope with complex user's requirements that require the collaboration of several more elementary services. However, with the increasing amount of candidate services of each business task that offer different QoS (Quality of Service) parameters, the selection of the optimal combination of services becomes a very hard task. This problem is more complex when dealing with temporal properties of business processes associated with *time-dependent QoS parameters* that can change according to temporal properties such as the execution time. Unlike static QoS which have been deeply studied in the existing service selection approaches, time-dependent QoS are insufficiently taken into consideration. In this paper, we are interested in the problem of service selection to satisfy a given business process while considering *temporal properties associated to time-dependent QoS*.**

***Keywords*-Service selection; Time-dependent QoS; Pruning; Multi-objective optimization; Business process;**

I. Introduction

Service-Oriented Architecture (SOA) paradigm has been a promising area which attracts attentions from research and industry communities. This paradigm ensures flexible systems by integrating loosely-coupled components often offered as services to build complex applications. These applications are usually specified as business processes composed of a set of abstract tasks. To satisfy user end-to-end QoS requirements while optimizing the overall utility, concrete services have to be selected and instantiated for each business process's abstract task. Giving the growing number of candidate services of each abstract task that can offer different QoS values, the selection of the optimal combination of services that fulfils users global QoS constraints becomes a very complex and time consuming task.

Despite active research in the context of service selection for abstract business processes, some issues still remain unsettled so far. The first issue is that most of existing selection approaches assume that services are always available and that QoS values are static. However, services can have temporal constraints related to their availabilities. Moreover, within different time periods, QoS attributes of candidate services can have different values [5], [11]. For instance, the response time of a service during daytime can be higher than night time due to access tendency. Thus, considering permanent availability of services and assuming only static QoS values is very restrictive to effectively represent services and reflect the impact of time on the QoS attributes. This issue makes the selection of the optimal solution more complex since the selection of each service may influence or be influenced by the selection of other services.

The second issue is that besides QoS properties, in real world scenarios, several constraints have to be considered to cater for not only users' requirements and service provider offers but also constraints specified at the business process level (e.g., structural and temporal constraints). Current selection approaches consider only structural constraints and assume that temporal properties can be viewed as a kind of QoS criteria without considering dependencies between several tasks. Usually, temporal constraints are considered when modeling and verifying business processes [7] and neglected during the selection of the best service combination process. These constraints can be specified explicitly by process designers or implicitly imposed by business rules and laws [8]. Given for example a partner of electronics manufacturing organization that requires in its business process that the manufacturing of peripheral parts has to finish no later than 15 time units after the starting of the process and that its organization can receive orders only at business hours. Considering temporal constraints when selecting the best service combination is a vital task since the violation of one or more temporal constraints may affect the successful execution of business processes.

The third issue is that when dealing with temporal properties, adopting a global optimization approach [13], [2] is not a practical solution and can lead to more 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 [1], [10] 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 dependencies that can exist between services does not guarantee that the global collaboration of the selected services succeeds even though a solution to the problem does exist. Consequently, a novel approach that allows selecting the optimal solution that satisfies all users constraints, while considering both *time-dependent QoS attributes* and *temporal constraints* specified at business and service levels and guaranteeing a good level of the selection algorithm performance is still needed.

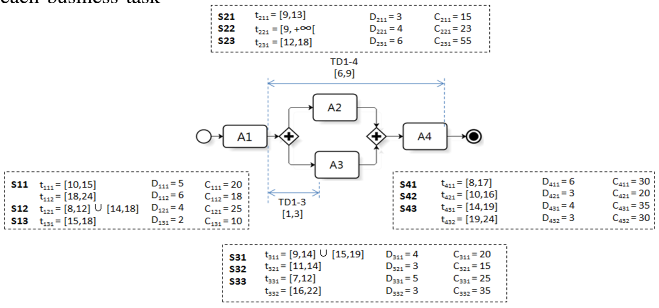
To adress these issues, in this paper we are interested in the problem of service selection to implement an abstract business process. We propose an hybrid approach that combines the use of local thresholds with global optimization to select the best solution in a reasonable time. At a first step, and in order to deal with scalability issues, we propose a *service pruning process* based on a set of computed local thresholds to narrow the search space and eliminate non adequate services prior to performing selection algorithm. The main idea is to identify significant thresholds that guarantee that the number of candidate services is large enough so that it is possible to find the optimal solution (if it exists), but also small enough to enhance the efficiency of the selection process. 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. At a second step, and based on the results of the pruning phase, we propose a selection algorithm that takes into consideration both time-dependent QoS attributes and temporal constraints and ensures the selection of the best services combination. To evaluate the proposed approach, we have conducted simulation experiments. The evaluation results demonstrate the effectiveness of our pruning based selection algorithm.

The rest of the paper is organized as follows. In the next section, we give a motivating example followed by a formal description of the service selection problem presented in the section III. Section IV details our pruning approach based on QoS and temporal constraints and section V presents our selection algorithm. In section VI, we evaluate our approach through experimental results. Finally, section VII illustrates some existing works and section VIII concludes the paper.

II. Global Overview

To illustrate the related features of our approach, let us consider a purchasing business process depicted in Figure 1, which characterizes a user's requirements.

This process has four abstract activities: first it starts by receiving the order (A_1), then the invoicing (A_2) and the manufacturing (A_3) of the ordered product are executed in parallel and finally it delivers the product (A_4). Two temporal dependencies are specified: $TD_{1,4}$ indicates that the time span between the end times of the reception and the delivery activities is between 3 and 6 time units and $TD_{1,3}$ denotes that the time period between the end of the


S21 t_{211} = [9,13] D_{211} = 3 C_{211} = 15
S22 t_{221} = [9, +∞[D_{221} = 4 C_{221} = 23
S23 t_{231} = [12,18] D_{231} = 6 C_{231} = 55
TD1-4
[6,9]
A1
A2
A3
A4
S11 t_{111} = [10,15] D_{111} = 5 C_{111} = 20
t_{112} = [18,24] D_{112} = 6 C_{112} = 18
S12 t_{121} = [8,12] ∪ [14,18] D_{121} = 4 C_{121} = 25
S13 t_{131} = [15,18] D_{131} = 2 C_{131} = 10
TD1-3
[1,3]
S41 t_{411} = [8,17] D_{411} = 6 C_{411} = 30
S42 t_{421} = [10,16] D_{421} = 3 C_{421} = 20
S43 t_{431} = [14,19] D_{431} = 4 C_{431} = 35
t_{432} = [19,24] D_{432} = 3 C_{432} = 30
S31 t_{311} = [9,14] ∪ [15,19] D_{311} = 4 C_{311} = 20
S32 t_{321} = [11,14] D_{321} = 3 C_{321} = 15
S33 t_{331} = [7,12] D_{331} = 5 C_{331} = 25
t_{332} = [16,22] D_{332} = 3 C_{332} = 35
Figure 1. An example of a business process with candidate services of each business task

reception activity and the start of the manufacturing activity is between 1 and 3 time units.

We suppose that three global constraints are associated to the process: (1) the price must not exceed 70 euros, (2) the duration of the execution must be lesser than 10 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 three concrete candidate services. Some of these services offer different QoS values according to their temporal properties related to their availability. For example, when the service S_{11} is available from 10 to 15 units of time, it offers a duration of 5 time units and a cost of 20 euros. When it is available from 18 to 24 units of time, it offers an execution duration equals to 6 time units with a cost of 18 euros.

Let us now search the best service combination to implement the business process where we consider only static QoS values. Thus, the best combination of services that satisfies the user requirements is C = (S_{13}, S_{21}, S_{32}, S_{42}). This combination can be selected based on the notion of dominance [1], [3]. 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. For example, the cost of the service S_{11} is equal to 20 euros form 10 to 15 units of time and to 18 euros from 18 to 24 units of time. Unfortunately, when considering time-dependent QoS, the combination C is no more valid even if the selected services are the best. Indeed, although the task A_2 should be executed after the task A_1, the service S_{21} is available in a time span before that of the service S_{13} and thus these two services can not be parts of the same solution. The combination $C^{'}$ = (S_{11}, S_{22}, S_{31}, S_{43}) where availability intervals are respectively [10,15], [15,19], [15,19] and [19,22] is a satisfactory solution.

Again, the combination $C^{'}$ can not be a satisfactory solution if we deal with further constraints expressed in the business process. Consider for example the temporal dependency $TD_{1,3}$. This constraint can not be satisfied by the combination $C^{'}$ and thus, another service combination should be selected such as $C^{''}$ = (S_{11}, S_{22}, S_{33}, S_{43}) with the following availability intervals [10,15], [15,19], [16,19] and [19,22].

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

III. Constraints Model 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.

A. Temporal Constraints

1) Business Level Constraints: A business process is usually defined by a set of activities (or abstract tasks) $\mathcal{A} = \{A_1, ..., A_n\}$. We denote by $Pd(A_i)$ the set of predecessors of the activity A_i. In addition, temporal constraints can be associated to business processes. We distinguish between intra and inter task temporal constraints [7].

- *Intra-task temporal constraints* relate to the start and the finish time of each task. We denote by $\mathcal{TC} = \{TC_1, ..., TC_u\}$ the set of intra-task temporal constraints. A temporal constraint $tc_i(TP, T) \in \mathcal{TC}$ is characterized by the activity concerned by the temporal constraint (i.e., $A_i \in \mathcal{A}$), 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* specify temporal dependencies between tasks that 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 $\mathcal{TD} = \{TD_1, ..., TD_v\}$. Each temporal dependency $td_{ij}(TP, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD}$ is characterized by a source and a destination tasks $A_i \in \mathcal{A}$ and $A_j \in \mathcal{A}$, a type $TP \in$ {start-to-start (SS), start-to-end (SE), end-to-start (ES), end-to-end (EE)} 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.

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

2) 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 $\mathcal{S}_i$ of potential candidate services. The potential services of an activity A_i are functionally equivalent and can be distinguished by their QoS attributes. Each QoS attribute $q \in \mathcal{QS}$ has either an increasing better value direction (the quality is better when the attribute value increases) or a decreasing better value direction (the quality is better when the attribute value decreases). 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.

As presented in Section II, services have temporal constraints related to their availabilities. Moreover, a service can associate temporal constraints to QoS attributes (i.e., time-dependent QoS). Each service $S_{ij} \in \mathcal{S}_i$ is characterized by a set $\mathcal{T}_{ij}$ of 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 end times. We denote by S_{ijk} the k^{th} timed instance of the service S_{ij} corresponding to the time interval $T_{ijk} \in \mathcal{T}_{ij}$. 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} QoS 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 [5].

B. Global User Constraints and Utility Function

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 $Q(q)$ denotes the global constraint value for the q^{th} QoS attribute of the composite service specified by the user (e.g., $Q(cost) = 70$ indicates that the cost of the required service has to be less than 70 cost units). Note that since we consider only quality 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. $\sum_{q \in \mathcal{QS}} W_q = 1$.

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 considered quality attribute and the structure of the business process. In our model we consider aggregation functions for four categories of QoS attributes that are widely used in the literature: Additive, Average, Multiplicative and Max-Operator. Table I shows examples of these aggregation functions. Thus, $Q(CS, q) = Agg_{A_i \in \mathcal{A}}(Q(A_i, q))$ with $Q(A_i, q)$ denotes the value of the quality attribute q of the component service corresponding to the task A_i.

Table I
EXAMPLES OF AGGREGATION FUNCTIONS

Attribute	**Sequential structure**	**Parallel structure**
Additive	$\sum_{A_i \in \mathcal{A}} Q(A_i, q)$	$\sum_{A_i \in \mathcal{A}} Q(A_i, q)$
Average	$\frac{1}{n} \sum_{A_i \in \mathcal{A}} Q(A_i, q)$	$\frac{1}{n} \sum_{A_i \in \mathcal{A}} Q(A_i, q)$
Multiplicative	$\prod_{A_i \in \mathcal{A}} Q(A_i, q)$	$\prod_{A_i \in \mathcal{A}} Q(A_i, q)$
Max-Operator	$\sum_{A_i \in \mathcal{A}} Q(A_i, q)$	$max_{A_i \in \mathcal{A}} \{Q(A_i, q)\}$

To evaluate the quality of the composite service based on the user preferences, we define an *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. The utility of a composite service CS is computed as:

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

Where $Q(q)^{max} = Agg_{A_i \in \mathcal{A}}(Q(A_i, q)^{max})$ and $Q(q)^{min} = Agg_{A_i \in \mathcal{A}}(Q(A_i, q)^{min})$ denote respectively the minimum and maximum aggregated values of the q^{th} quality attribute of CS with $Q(A_i, q)^{max} = max\{Q(S_{ijk}, q), \forall S_{ij} \in \mathcal{S}_i, \forall T_{ijk} \in \mathcal{T}_{i,j}\}$ and $Q(A_i, q)^{min} = min\{Q(S_{ijk}, q), \forall S_{ij} \in \mathcal{S}_i, \forall T_{ijk} \in \mathcal{T}_{i,j}\}$. 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.

Intuitively, to select the best services, all candidate services can be taken into account. However, this is impracticable when the number of services and constraints (QoS and temporal) increases since the time needed to solve the service selection problem becomes exponential. 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 potential 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 combination of services which are not relevant to the selection problem so that the optimal solution still be found.

IV. SERVICE PRUNING

The basic idea of our pruning approach is to avoid discarding any candidate service that might be part of a feasible solution. This is done by computing *local thresholds* of each task while ensuring that these thresholds are relaxed as much as possible. In our work, We propose two search space reduction techniques: (1) *QoS constraints based pruning* and (2) *temporal constraints based pruning*. In the following, we detail how we measure thresholds using these two techniques.

A. QoS Constraints based Pruning

The QoS based pruning strategy aims to compute QoS thresholds for individual tasks for each QoS attribute q that will serve as local upper bound constraints. In fact, if a service has at least one QoS value that does not satisfy the local thresholds, the selection of this service will violate the global constraints and thus it can be pruned.

A local threshold $Q_{LT}(A_i, q)$ for the q^{th} attribute of the task A_i depends on both the value required in the user global constraint $Q(q)$ and the minimum value of this QoS attribute (i.e., $Q(A_i, q)^{min}$). 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 constraint is satisfied. Computing these thresholds needs to consider both the structure of the business process and the distinctive characteristics for each QoS attribute. We have defined a set of formulas to compute local thresholds where we take account the categories of quality attributes presented in Table I. For lack of space, we present only the additive attributes QoS. The full set of formulas and more details are given in an extended version of this paper[1]. To measure the local threshold of an additive attribute, we define the formula (2).

$$Q_{LT}(A_i, q) = Q(q) - \sum_{A_j \in \mathcal{A}, j \neq i} Q(A_j, q)^{min}, \forall A_i \in \mathcal{A} \quad (2)$$

For instance, given the example in the Figure 1 with $Q(cost) = 70$. By applying the previous formula, we obtain the following thresholds for each task: $Q_{LT}(A_1, cost) = 70 - (15 + 15 + 20) = 20$, $Q_{LT}(A_2, cost) = 25$, $Q_{LT}(A_3, cost) = 25$ and $Q_{LT}(A_4, cost) = 30$. After computing the cost thresholds of each task, the number of candidate services is restricted. For example, all service instances that have a cost greater than 20 cost units for the first task will be eliminated (e.g., the service S_{12}).

B. Temporal Constraints based Pruning

Although QoS constraints based pruning keeps for each activity only candidate services that are likely to be a member of the optimal solution, some uninteresting services still need to be removed when taking into consideration temporal constraints. 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 end times) of each activity with respect to the required deadline.

[1]http://homepages.laas.fr/nguermou/ICWS14/ICWSExtendedVersion.pdf

1) Execution Duration: Computing local thresholds for the execution duration attribute is a very hard task when handling business processes where several structural and temporal constraints and more specifically temporal dependencies exist between tasks. This is explained by the fact that some temporal dependencies may be overlapped or included in each other. Additionally, temporal dependencies may have different types and thus should be resolved differently. To deal with this, we rely on a constraint optimization model while considering structural and temporal constraints of business process. The proposed model is applied for each task $A_i \in \mathcal{A}$ to search for its maximum duration while minimizing the durations of all other tasks. Therefore, the objective function of our model can be expressed as follows:

$$\text{minimize} \sum_{A_j \in \mathcal{A}, j \neq i} Q(A_j, dur) - Q(A_i, dur) \tag{3}$$

The duration of each task $Q(A_j, dur)$ belongs to the interval $[Q(A_j, dur)^{min}, Q(A_j, dur)^{max}]$, $\forall A_j \in \mathcal{A}$ and it is related to two main variables: its start time *(st)* and its finish time *(ft)*, thus:

$$ft_j = st_j + Q(A_j, dur), \forall A_j \in \mathcal{A} \tag{4}$$

The computation of local thresholds must ensure that structural and temporal constraints are still satisfied. For simplicity, only end-to-start temporal dependencies are considered since other dependencies can be defined by the same manner. Therefore, we add the following set of constraints to our model:

$$ft_j \leq st_k, \forall A_k \in \mathcal{A}, A_j \in \mathcal{P}d(A_k) \tag{5}$$

$$ft_j + D_{jk}^{min} \leq st_k, \forall td_{jk}(ES, D_{jk}^{min}, D_{jk}^{max}) \in \mathcal{TD} \tag{6}$$

$$st_k \leq ft_j + D_{jk}^{max}, \forall td_{jk}(ES, D_{jk}^{min}, D_{jk}^{max}) \in \mathcal{TD} \tag{7}$$

We assume that the start and finish times of each task should not exceed the global duration required by the user (i.e., $st_j, ft_j \in [0, Q(dur)], \forall A_j \in \mathcal{A}$). Then, the local threshold of each task A_i is $Q_{LT}(A_i, dur) = Q(A_i, dur)$.

2) 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, selecting a wrong start time for one service can lead to several wrong choices for start times of its successor services. To avoid possible unnecessary combinations, we need to reduce the number of start and finish times to consider. To do so, 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 consider four variables for each task: earliest start time (*est*), latest start time (*lst*), earliest finish time (*eft*) and latest finish time (*lft*). Our goal is to guarantee that largest intervals will be computed (i.e., maximize the distance between the *lft* (resp. *lst*) and the *eft* (resp. *est*) of all tasks). Therefore, the objective function is as follows:

$$\text{maximize} \sum_{A_i \in \mathcal{A}} lft_i - \sum_{A_i \in \mathcal{A}} eft_i \tag{8}$$

Constraints (9) and (10) guarantee that the *eft* (resp. *lft*) of each task is represented by the sum of its *est* (resp. *lst*) and its duration.

$$eft_i = est_i + Q(A_i, dur), \forall A_i \in \mathcal{A} \tag{9}$$

$$lft_i = lst_i + Q(A_i, dur), \forall A_i \in \mathcal{A} \tag{10}$$

The duration of each task A_i belongs to the interval $[Q(A_i, dur)^{min}, Q(A_i, dur)^{max}]$ and its start and finish times (i.e., $est_i, lst_i, eft_i, lft_i$) belong to the interval $[min_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{min}\}, max_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{max}\}]$. To guarantee that the deadline is not violated, we add these constraints:

$$eft_n \leq deadline \tag{11}$$

$$lft_n \leq deadline \tag{12}$$

To deal with structural dependencies, we propose the following constraints that garantee that for each activity A_j, its earliest start (resp. latest start) time occurs after the earliest finish (resp. latest finish) time of all its predecessor tasks.

$$eft_i \leq est_j, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \tag{13}$$

$$lft_i \leq lst_j, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \tag{14}$$

Furthermore, it is vital to check if temporal constraints are satisfied when computing the large schedule of each task. To deal with *Intra-task temporal constraints*, we propose constraints from (15) to (18). For simplicity, we only consider the temporal constraints (MSO, MFO and SNET). For example, the constraint (15) ensures that for each constraint of the form Must Start On T, the earliest start time and the latest start time are equal to the time point T.

$$est_i = lst_i = T, \forall tc_i(MSO, T) \in \mathcal{TC} \tag{15}$$

$$eft_i = lft_i = T, \forall tc_i(MFO, T) \in \mathcal{TC} \tag{16}$$

$$est_i \geq T, \forall tc_i(SNET, T) \in \mathcal{TC} \tag{17}$$

$$lst_i \geq T, \forall tc_i(SNET, T) \in \mathcal{TC} \tag{18}$$

To deal with *Inter-task temporal constraints* (i.e., temporal dependencies), we propose the following constraints considering only end-to-start temporal dependencies.

$$eft_i + D_{ij}^{min} \leq est_j, \forall td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD} \tag{19}$$

$$est_j \leq eft_i + D_{ij}^{max}, \forall td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD} \tag{20}$$

$$lft_i + D_{ij}^{min} \leq lst_j, \forall td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD} \tag{21}$$

$$lst_j \leq lft_i + D_{ij}^{max}, \forall td_{ij}(ES, D_{ij}^{min}, D_{ij}^{max}) \in \mathcal{TD} \quad (22)$$

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, the largest time slots of all tasks when considering all imposed constraints and with a deadline equals to 23 are respectively: [8,16], [10,20], [11,20] and [14,23]. Based on these intervals some service instances have to be pruned (e.g., S_{112} and S_{131}) or some restrictions have to be performed to their intervals (e.g., S_{222}).

C. 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. The pruning steps are given by the Algrithm 1.

Algorithm 1 Service Pruning Algorithm
```
 1: for each $A_i \in \mathcal{A}$ do
 2:     for each $S_{ij} \in S_i$ do
 3:         for each $T_{ijk} \in \mathcal{T}_{ij}$ do
 4:             for each $q \in \mathcal{QS}$ do
 5:                 if $Q(S_{ijk}, q) > Q_{LT}(A_i, q)$ then
 6:                     $S_i \leftarrow S_i \setminus \{S_{ijk}\}$
 7:                     break
 8:             if $[t_{ijk}^{min}, t_{ijk}^{max}] \cap [est_i, lft_i] = \emptyset$ then
 9:                 $S_i \leftarrow S_i \setminus \{S_{ijk}\}$
10:                 break
11:             else $\{[t_{ijk}^{min}, t_{ijk}^{max}] \cap [est_i, lft_i] = [X, Y]\}$
12:                 if $Y - X < Q(S_{ijk}, dur)$ then
13:                     $S_i \leftarrow S_i \setminus \{S_{ijk}\}$
14:                     break
15:                 else
16:                     if $t_{ijk}^{min} < X$ then
17:                         $t_{ijk}^{min} \leftarrow X$
18:                     if $t_{ijk}^{max} > Y$ then
19:                         $t_{ijk}^{max} \leftarrow Y$
20:     if $S_i = \emptyset$ then
21:         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 prune services based on QoS thresholds (line 5 to 7) and then based on time spans (line 8 to 19). If a local threshold is violated, it is not worth to check the fulfilment 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 8 to 14). Otherwise, the time span of the service instance should be restricted to the span of its task (line 16 to 19). Finally, if at least one task does not have any candidate service, we conclude that the selection problem has no feasible solutions (lines 20 and 21).

V. 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 end times of each service S_{ij} is a 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.

To give a more flexible solution, we search for the service combination that has the best utility function while specifying the largest execution time interval of each selected service rather than selecting a single starting time point. Hence, for each service in addition to its QoS values, four temporal values are specified: *est*, *eft*, *lst* and *lft*. These values will be useful at execution time to monitor the execution of services and predict the impact of one or more violations on the execution plan. The proposed model selects exactly one atomic service of each abstract task with the corresponding earliest and latest start and finish times while optimizing the overall utility and satisfying all constraints. The objective function of our optimization model is as follows:

$$\text{maximize} \sum_{q \in \mathcal{QS}} W_q * F_u(q) + \Big(\sum_{A_i \in \mathcal{A}} lft_i - \sum_{A_i \in \mathcal{A}} eft_i\Big) \quad (23)$$

With: $F_u(q) = \dfrac{Q(q)^{max} - Q(q, CS)}{Q(q)^{max} - Q(q)^{min}}$

Such that for each $q \in \mathcal{QS}$:

$$Q(CS, q) = Agg_{A_i \in \mathcal{A}}\Big(\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * Q(S_{ijk}, q)\Big) \quad (24)$$

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 $[Q(A_i, q)^{min}, Q(A_i, q)^{max}]$, $\forall A_i \in \mathcal{A}$ and $\forall q \in \mathcal{QS}$. To guarantee that only one service will be selected for each task we define the following formula:

$$\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} = 1, \forall A_i \in \mathcal{A}, a_{ijk} \in \{0, 1\} \quad (25)$$

Since all global constraints have to be satisfied when selecting the optimal solution, we add the following constraint:

$$Q(CS, q) \leq Q(q), \forall q \in \mathcal{QS} \quad (26)$$

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

$$\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * t_{ijk}^{min} \leq est_i \quad (27)$$

$$est_i \leq \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * (t_{ijk}^{max} - Q(S_{ijk}, dur)) \quad (28)$$

$$\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * t_{ijk}^{min} \leq lst_i \quad (29)$$

$$lst_i \leq \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * (t_{ijk}^{max} - Q(S_{ijk}, dur)) \quad (30)$$

The start and finish times of each task A_i (i.e., $est_i, lst_i, eft_i, lft_i$) belong to the interval $[min_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{min}\}, max_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{max}\}]$. To specify the relation between the start and finish times of each task A_i, the two following constraints are specified:

$$eft_i = est_i + \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * Q(S_{ijk}, dur) \quad (31)$$

$$lft_i = lst_i + \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * Q(S_{ijk}, dur) \quad (32)$$

Finally, to check the satisfaction of structural and temporal constraints, we use constraints from (13) to (22). Note that it is not worth to check that the deadline will be fullfilled since we consider time intervals of preselected services that belong to the time spans of the corresponding tasks.

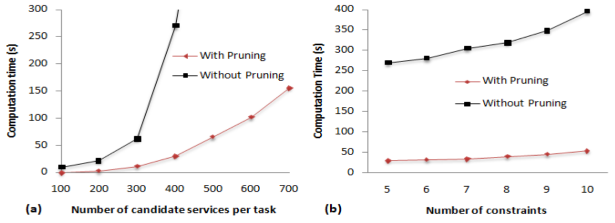
VI. Experimental Results

In this section, we present experimental results of the pruning based service selection approach we propose. For this purpose, the time-dependent QoS attributes of all candidate services and all constraints were randomly chosen. The QoS attributes for each service instance were generated for a time horizon with 150 time points and distributed in the range between 1 and 100. The computation time of each selection algorithm was averaged over 50 randomly generated problem instances. 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 our approach, we used the constraint solver Choco[2]. To evaluate the effectiveness of the proposed approach, we compare the computation time obtained in two cases: when no pruning procedure is applied and when applying both QoS and temporal constraints based pruning approach.

First, experiments were conducted in relation to the number of candidate service instances per task that varies from 100 to 700 with 5 business tasks. Here, each service instance is associated with 3 QoS values and 2 temporal constraints. The results provided in Figure 2(a) indicate that applying both pruning techniques significantly outperforms the basic

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

Figure 2. Evaluation of the pruning based selection approach



algorithm. In fact, by increasing the number of candidate services, the computation time of our approach increases very slowly compared to the basic one.

Figure 2(b) shows the computation time of the selection algorithms when the number of constraints varies between 5 and 10 and the number of tasks is fixed to 5 with 400 candidate service instances for each task. As before, while the computation time of the basic algorithm significantly increases due to the fact of the increased 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 very high. This is an expected behavior since by considering several QoS and temporal constraints, the number of feasible solutions decreases. Hence, more services are likely to violate one or more constraints and thus they should be pruned.

In all the test cases, the results show a significant gain in performance when applying the pruning approach which scales better than the traditionally algorithm where all candidate services are considered. The accuracy of the new algorithm is more obvious with complex selection problems where the number of candidate services, tasks and constraints is very high. This is due to the fact that the number of eliminated services significantly increases when the number of candidate services, tasks and constraints increases.

VII. Related Work

QoS based service selection problem has attracted researchers' attention in multiple domains. To solve this problem, some works adopt exhaustive methods to find the best assignment of services to abstract tasks. In [13], Zeng et al. adopt mixed linear programming techniques to select the optimal services for the composition and achieve global optimization of QoS attributes. This work has been extended in Ardagna et al. [2] to include local constraints and loop peeling to deal with composition structures with cycles. To solve the scalability issues, some researchers have adopted approximate methods and proposed heuristics to find a near-to-optimal solution more efficiently than exact solutions. Yu et al. [12] introduce two alternative models for the QoS-based service composition problem: the combinatorial model and the graph model. Based on these models, authors proposed heuristic algorithms to achieve better performance. In [4], Canfora et al. model and resolve the service selection problem based on genetic algorithms. These approaches, however, do not provide strategies to reduce the search space before selection and do not cater for 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 [9] and thus they cannot be applied in QoS aware service selection problems. Other proposals have applied the decomposition techniques to decompose global constraints into local ones and then, reduce the number of candidate services. For instance, Alrifai et al. [1] use a mixed integer programming model to compute local constraints based on QoS levels. After that, a local selection strategy is applied to select the best service for each task. As a step forward, Qi et al. [10] suggest a local optimization method to further reduce the number of candidate services based on QoS levels and enumeration. 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. Barakat et al. [3] 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. The proposed solution does not allow the computing of local thresholds of each task based on QoS constraints. Moreover, it cannot be applied when dealing with time-dependent QoS attributes.

Temporal properties have been considered by some works when selecting the best service composition [6]. Zhang et al. [14] take into consideration the dynamic aspect of QoS attributes to compose services in multi-domain environments. To deal with time-dependent QoS values, Wagner et al. [11] define a multi-objective optimization based approach that selects the best combination of services while specifying the start and finish time of each service according to the QoS values at each time period. Nevertheless, this work does not consider temporal constraints at the business level and no pruning approach has been applied. In [8], Liang et al. propose a penalty-based genetic algorithm to select services under temporal constraints. Authors assume that QoS values do not depend on the time of the execution and only upper bound constraints between activities are considered.

VIII. 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 temporal constraints. This is an important step towards the consideration of complex business models and service offers in practical applications. The selection approach we propose relies on two pruning mechanisms: QoS constraints and temporal constraints based pruning. These mechanisms are defined upon a set of formulas and constraint optimization algorithms that allows computing local thresholds for each QoS attribute as well as start and end times of each task. Based on these thresholds, the aim of the pruning phase is to reduce the number of candidate services to be considered while ensuring that the optimal solution still be found. To evaluate the effectiveness of the proposed approach, an optimization algorithm has been applied based on constraint optimization programming to select the best combination of services. Experimental results show a significant improvement in performance especially in terms of computational time through applying our pruning techniques prior to the selection process. In the current approach, only structural and temporal dependencies are considered between services. As a future work, we aim to consider further possible correlations between quality attributes and more complex time-based QoS change cycles and patterns when pruning unadequate services. We also plan to evaluate our approach based on real world scenarios and to study the complexity of the proposed pruning and selection algorithms.

References

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

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

[3] L. Barakat, S. Miles, I. Poernomo, and M. Luck. Efficient multi-granularity service composition. In *ICWS*, pages 227–234, 2011.

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

[5] L. Chen, J. Yang, and L. Zhang. Time based qos modeling and prediction for web services. In *ICSOC*, pages 532–540, 2011.

[6] N. Guermouche and C. Godart. Compoisition of web services based on timed mediation. *International Journal of Newt-Generation Computing (IJNGC'14)*, 2014.

[7] A. Lanz, J. Kolb, and M. Reichert. Enabling personalized process schedules with time-aware process views. In *CAiSE Workshops*, pages 205–216, 2013.

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

[9] Z. J. Oster, G. R. Santhanam, and S. Basu. Identifying optimal composite services by decomposing the service composition problem. In *ICWS*, pages 267–274, 2011.

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

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

[12] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *TWEB*, 1(1), 2007.

[13] 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.*, 30(5):311–327, 2004.

[14] T. Zhang, J. Ma, C. Sun, Q. Li, and N. Xi. Service composition in multi-domain environment under time constraint. In *ICWS*, pages 227–234, 2013.