An Efficient Service Selection Approach with Time-Dependent QoS

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

Tarak Chaari[3] 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
mohamed.jmaiel@enis.rnu.tn

***Abstract*—In advanced service-oriented computing, complex applications are usually specified as abstract business processes. The execution of these applications requires the selection of a set of services to invoke abstract business tasks. With the growing number of alternative services of each business task that differ in their QoS, the selection of the best combination of services that satisfies business process constraints and end-to-end users' requirements becomes a complex decision problem. Current selection approaches consider only static QoS and ignore the fact that QoS values can depend on temporal properties. In this paper, we propose a novel service selection approach considering *time-dependent QoS* attributes. The proposed approach introduces two search space reduction mechanisms that combine QoS and temporal constraints. The application of these mechanisms improves the performance of the selection process which is demonstrated by experimental results.**

***Keywords*-Service selection; Time-dependent QoS; Search space reduction.**

I. Introduction

Service selection for business process tasks has been widely treated in both research and industry communities. Usually, service selection approaches search for the optimal combination of services that meets user end-to-end QoS requirements while optimizing the overall utility. With the increasing amount of available candidate services for each task providing the same functionalities but differ on their quality values, the selection of the optimal solution that fulfills global QoS constraints specified in the user requirements becomes a very complex task.

Generally, users requirements are considered as global constraints of the composite service. Several methods have been proposed to tackle the selection problem while dealing with global QoS constraints. These methods fall in two categories. The first one consists in searching candidate services that optimize the QoS of composite service while satisfying global QoS constraints [1], [2], [3]. The second category assumes that global constraints can be considered as the aggregation of a set of local ones [4], [5], [6]. Based on these latter, local optimization algorithms are applied to select the best service for each abstract task. The aim of these works is to reduce the complexity of the service selection problem while increasing the performance of the proposed algorithm.

Figure 1. An example of a business process and candidate services of abstract tasks

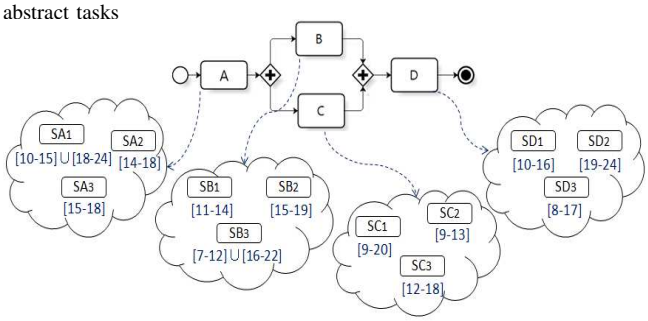


Table I
QoS VALUES OF SOME CANDIDATE SERVICE INSTANCES

Ser Ins	Dur	Cst	Ser Ins	Dur	Cst	Ser Ins	Dur	Cst
SA_{11}	5	20	SB_{11}	3	15	SC_{11}	4	23
SA_{12}	6	18	SB_{21}	4	20	SC_{21}	3	15
SA_{21}	4	25	SB_{31}	5	25	SD_{11}	3	20
SA_{31}	2	10	SB_{32}	3	35	SD_{21}	3	30

However, existing approaches overlooked a commonly important aspect in reality: most QoS values offered by service providers are not static and can change over time [7], [8], [9]. As a result, within different time periods, QoS attributes of candidate services can have different values. For instance, the invocation of the service during business hours is more expensive than invoking it outside these peak hours. Considering *time-dependent QoS* values makes the selection of the best solution more complex since the selection of one service may influence or be influenced by other services.

To better illustrate this issue, let us consider the example presented in Figure 1, where the business process has four tasks. Each abstract task has several candidate services and some of them have more than one service instance (e.g., SA_1 and SB_3) according to the values of quality attributes (Table I). For instance, the values of the execution duration and the cost offered by the service SA_1 are 5 and 20 respectively when the service is available from 10 to 15 units of time

and they are equal to 6 and 18 when it is available from 18 to 24 units of time. In this example, we suppose that the user requires three global constraints: (1) the price must not exceed 110 cost units, (2) the execution duration must be lesser than 15 units of time, and (3) the deadline must not exceed 11 PM (i.e., 23 units of time in our example).

If we assume that QoS values are static, the best combination of services can be C = (SA_{31}, SB_{11}, SC_{21}, SD_{11}) since all selected services are not dominated by any other services for all QoS attributes. Nevertheless, this solution can not be valid when dealing with time-dependent QoS. In fact, the service instance SC_{21} is valid in a time span before that of the service SA_{31} and thus these two service instances can not be part of the same solution. However, the combination $C^{'}$ = (SA_{11}, SB_{21}, SC_{11}, SD_{21}) where availability intervals are respectively [10,15], [15,19], [15,19] and [19,22] is a valid solution.

Considering *time-dependent QoS* values is not a trivial task and makes the selection problem more complex. In this paper, we propose two search space reduction mechanisms based on both QoS and temporal constraints. These mechanisms are applied prior to the selection process to narrow the number of candidate services while ensuring that only service combinations which are guaranteed to violate one or more constraints are not considered in the selection process. The rest of the paper is organized as follows. In the next section, we give a formal description of the service selection model. Section III details our search space reduction approach based on QoS and temporal constraints and Section IV presents our selection algorithm. In Section V, we give an evaluation of our approach. Finally, in Section VII, we discuss the related work and Section VII concludes the paper.

II. Service Selection Model

In this paper, we are interested in service selection problem which 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. Business Level Constraints

A business process is usually defined by a set of abstract tasks $\mathcal{A} = \{A_1, ..., A_n\}$. In this paper, we consider two types of structural dependencies between tasks (Sequential and Parallel). We denote by $Pd(A_i)$ the set of predecessors of the activity A_i.

B. Service Level Constraints

Apart from constraints specified at the business level, some 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 candidate 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 value direction (the quality is better when the attribute value increases) or a decreasing value direction (the quality is better when the attribute value decreases). For the sake of simplicity, henceforth we consider only QoS attributes with decreasing value direction.

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 (Figure 1). To represent QoS variations related to these time-dependent QoS, we consider one or several instances of each service. Each instance is associated to a time interval which specifies its start and end times. We denote by S_{ijk} the k^{th} 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}. The specification of time-dependent QoS models can be achieved using existing QoS prediction methods [8].

C. 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 $Q(q)$ 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. $\sum_{q \in \mathcal{QS}} W_q = 1$.

III. Search Space Reduction Approach

Intuitively, to select the best services to implement a business process, all candidate services can be taken into account. However, this is impracticable when the number of services and constraints increases since the time needed to solve the service selection problem becomes exponential. In fact, not all services are potential candidates for the feasible solution. To overcome this problem, we propose a search space reduction approach to reduce the number of candidate services of each task and thus, reducing the number of possible uninteresting combination of services so that the optimal solution still be found.

The basic idea of our search space reduction 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. We propose two search space reduction techniques: (1) *QoS constraints based reduction space* and (2) *time constraints based reduction space*.

In the following, we detail how we measure thresholds using these two techniques.

A. QoS Constraints based reduction space

The QoS based reduction space strategy allows computing QoS thresholds for individual tasks for each QoS attribute q that will be considered as local upper bound constraints such that the unsatisfaction of at least one of these constraints by a candidate service guarantees the violation of the global constraints and thus this service can be removed from the set of candidate services.

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 component services. The aggregation function Agg depends on the considered quality attribute and the structure of the business process [10]. In this paper, we consider the aggregation functions of four categories of QoS attributes: Additive, Average, Multiplicative and Max-Operator. 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. We denote by $Q(A_i, q)^{max}$ and $Q(A_i, q)^{min}$ respectively the minimum and maximum value of the q^{th} quality attribute of the task A_i 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 local threshold $Q_{LT}(A_i, q)$ for the q^{th} attribute of the task A_i depends on both the global constraint of the user $Q(q)$ and the minimum value of this QoS attribute offered by the candidate services of the task A_i (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. In the following, we present a set of formulas to compute local thresholds for each business task.

1) Additive Attributes: The value of an additive attribute (e.g., the execution cost) for the composite service can be determined through the sum of the values of this attribute for all the component services. To measure the local threshold of an additive attribute, we define Formula (1).

$$Q_{LT}(A_i, q) = Q(q) - \sum_{A_j \in \mathcal{A}, j \neq i} Q(A_j, q)^{min}, \forall A_i \in \mathcal{A} \quad (1)$$

For instance, given the example presented in the Figure 1 with $Q(cost) = 70$, the thresholds of the tasks A_1, A_2, A_3 and A_4 are respectively 20, 25, 25 and 30. Therefore, 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 SA_{21}).

2) Average Attributes: In this category, the value of the attribute (e.g., the availability) for the composite service is measured by the average of the values of the attribute of its atomic services. Formula (2) indicates how to compute local thresholds of average attributes. Let n be the number of business tasks.

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

3) Multiplicative Attributes: Multiplicative attributes of composite services (e.g., the reputation) can be computed by multiplying the values of the attribute of the component services. Formula (3) allows computing the local thresholds of multiplicative attributes for a sequential and parallel structures.

$$Q_{LT}(A_i, q) = Q(q) / \prod_{A_j \in \mathcal{A}, j \neq i} Q(A_j, q)^{min}, \forall A_i \in \mathcal{A} \quad (3)$$

4) Max-Operator Attributes: These attributes differ from other attribute categories in that different aggregation functions are used in sequential and parallel structures (e.g., the execution time). Max-operator attributes for a composite service are measured by the sum of attribute values of its atomic services in a sequential structures, and the highest branch value in each parallel structure. Thus, to compute local thresholds for each task in the business process, two cases are considered:

- If A_i belongs to a sequential structure S:

$$Q_{LT}(A_i, q) = Q(q) - \sum_{A_j \in \mathcal{S}, j \neq i} Q(A_j, q)^{min} - \sum_{l=1}^{m} max_{A_j \in P_l}\{Q(A_j, q)^{min}\}, \forall A_i \in \mathcal{A} \quad (4)$$

  With m is the number of parallel structures and P_l denotes the parallel structure number l.
- If A_i belongs to a parallel structure P_q:

$$Q_{LT}(A_i, q) = Q(q) - \sum_{A_j \in \mathcal{S}} Q(A_j, q)^{min} - \sum_{l=1, l \neq q}^{m} max_{A_j \in P_l}\{Q(A_j, q)^{min}\}, \forall A_i \in \mathcal{A} \quad (5)$$

B. Time Constraints based reduction space

Although QoS constraints based reduction space allows the elimination of services which are guaranteed to violate one or more global QoS constraints, further services can be removed when considering time-dependent QoS values. In the following, we detail how we consider temporal constraints to further reduce the number of candidate services based on the time span of each abstract task and the required deadline.

The main idea is to compute the earliest possible start time st_i and the latest possible finish time ft_i of each business task $A_i \in \mathcal{A}$ based on the deadline required by the user while satisfying structural constraints. These time intervals will be considered as local thresholds for the start and finish time of each task such that all service instances whose time intervals do not belong to the computed time spans will be pruned. Nevertheless, the specification of the largest time spans of business tasks is not a trivial task when dealing with complex business processes.

To deal with this issue, we propose a constraint optimization model that computes the largest time span of each task so that all structural 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 the following objective function:

$$\text{maximize} \sum_{A_i \in \mathcal{A}} ft_i - \sum_{A_i \in \mathcal{A}} st_i \tag{6}$$

Constraint (7) guarantees that the finish time of each task is represented by the sum of its start time and its duration.

$$ft_i = st_i + Q(A_i, dur), \forall A_i \in \mathcal{A} \tag{7}$$

Each task $A_i \in \mathcal{A}$ has a duration within the interval $[Q(A_i, dur)^{min}, Q(A_i, dur)^{max}]$ and its start and finish times (i.e., st_i, ft_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 Constraint (8):

$$ft_n \leq deadline \tag{8}$$

To deal with structural dependencies, we propose the following constraints 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.

$$st_i + Q(A_i, dur)^{min} \leq st_j, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \tag{9}$$

$$ft_i + Q(A_j, dur)^{min} \leq ft_j, \forall A_j \in \mathcal{A}, A_i \in \mathcal{P}d(A_j) \tag{10}$$

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 intervals of all tasks when considering all structural and temporal constraints and with a deadline equals to 23 are respectively: [8,16], [10,20], [10,20] and [14,23]. Based on these intervals some service instances have to be pruned (e.g., SA_{12} and SB_{31}) or some restrictions have to be performed to their intervals (e.g., SD_{31}).

IV. Service Selection

Once the search space reduction process is finished and the relevant candidate services are selected, we proceed to the selection of the best service combination to handle the user requirements. To do so, we model the selection problem as a constraint optimization problem. The proposed model selects exactly one atomic service of each abstract task with the corresponding start and finish times (i.e., st_i, ft_i) while optimizing the overall utility and satisfying all constraints. Therefore, the objective function of our optimization model is as follows:

$$\text{maximize} \sum_{q \in \mathcal{QS}} W_q * \frac{Q(q)^{max} - Q(CS, q)}{Q(q)^{max} - Q(q)^{min}} \tag{11}$$

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

$$Q(CS, q) = Agg_{A_i \in \mathcal{A}}(\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * Q(S_{ijk}, q)) \tag{12}$$

$$Q(q)^{max} = Agg_{A_i \in \mathcal{A}}(Q(A_i, q)^{max}) \tag{13}$$

$$Q(q)^{min} = Agg_{A_i \in \mathcal{A}}(Q(A_i, q)^{min}) \tag{14}$$

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 constraint:

$$\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} = 1, \forall A_i \in \mathcal{A}, a_{ijk} \in \{0, 1\} \tag{15}$$

Since all global constraints have to be satisfied when selecting the optimal solution, we add Constraint (16):

$$Q(CS, q) \leq Q(q), \forall q \in \mathcal{QS} \tag{16}$$

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

$$\sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * t_{ijk}^{min} \leq st_i \tag{17}$$

$$st_i \leq \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * (t_{ijk}^{max} - Q(S_{ijk}, dur)) \tag{18}$$

$$ft_i = st_i + \sum_{S_{ij} \in \mathcal{S}_i} \sum_{T_{ijk} \in \mathcal{T}_{ij}} a_{ijk} * Q(S_{ijk}, dur) \tag{19}$$

The start and finish times of each task A_i belong to the interval $[min_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{min}\}, max_{k \in \mathcal{T}_{ijk}}\{t_{ijk}^{max}\}]$. Finally, to check the satisfaction of structural constraints, we add the following constraint:

$$ft_j \leq st_i, \forall A_i \in \mathcal{A}, A_j \in \mathcal{P}d(A_i) \tag{20}$$

Figure 2. Performance vs. Number of candidate services

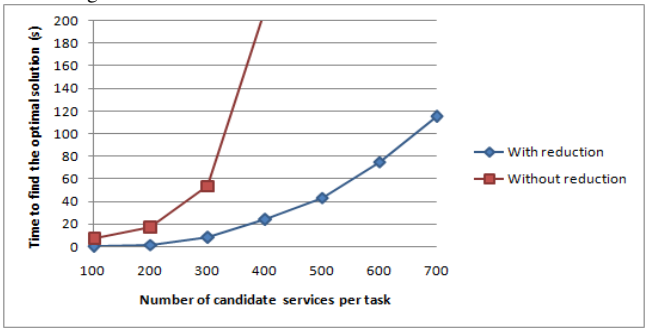
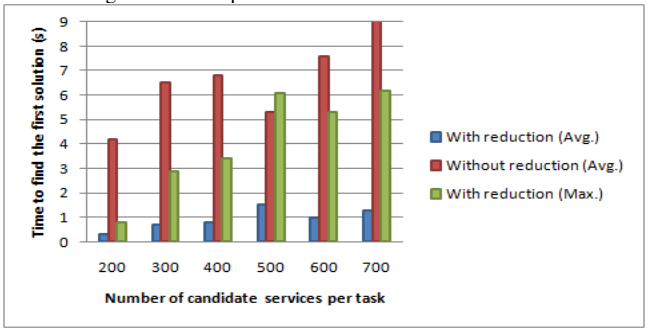


Figure 3. Computation time to find the first solution



V. Experimental Results

In this section, we present the experimental results of our approach focusing on its performance in terms of the computation time. In these experiments, 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 use Choco[1] as constraint solver.

First, experiments were conducted in relation to the computation time required to find the optimal solution. The number of candidate service instances per task varies from 100 to 700 with 5 business tasks. Here, each service instance is associated with 3 QoS values (price, execution duration and reliability). The results provided in Figure 2 show that while the computation time of the basic algorithm (i.e., no preprocessing techniques are applied) increases exponentially by increasing the number of candidate services per task, the time of our algorithm increases very slowly even when the number of services is very high. This is an expected behavior since the number of eliminated services grows especially when the number of candidate services increases.

Figure 3 shows the computation time required to find a feasible solution with respect to the number of candidate services per task which varies between 200 and 700. 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. We note that the time needed to compute local thresholds is negligible compared to the computation time of the selection process. This is explained by the fact that the search space reduction process is independent on the number of candidate services.

VI. Related Work

Several works have been proposed to tackle the problem of the selection of the best combination of services that satisfies users global QoS constraints while maximizing the overall utility. Some approaches adopt exhaustive methods to find the optimal solution. In [1], Zeng et al. use mixed linear programming techniques to select the optimal component 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. In [11], Ben Hassine et al. discuss a web service composition approach using constraint programming. Nevertheless, these methods are only suitable for small problems, as the complexity of the selection algorithm increases exponentially when the problem size increases.

To deal with scalability issues, other researchers adopt approximate methods to find a near-to-optimal solution more efficiently than exact solutions. Yu et al. [12] introduced two alternative models for the QoS-based service composition problem: the combinatorial model and the graph model. Based on these models, authors propose heuristic algorithms to find a solution to the selection problem. In [3], Canfora et al. model and resolve the service selection problem based on genetic algorithms. These approaches, consider only static QoS values and do not provide strategies to reduce the search space before selection.

To reduce the computational time of service selection algorithms, an alternative proposal is to narrow the search space. For instance, Alrifai et al. [4] decompose global constraints into local ones using mixed integer programming. Then, a local selection strategy is applied to select the best service for each task. As a step forward, Qi et al. [5] suggest a local optimization method to further reduce the number of candidate services based on QoS levels and enumeration. Similar works are proposed to identify local constraints based on global ones. In [13], Sun et al., introduce a QoS decomposition approach based on the mean and the standard deviation of each QoS attribute while considering several

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

composition structures. Another approach has been proposed in [6] 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. In [14], Barakat et al. introduce two space reduction techniques to reduce the number of candidate services and the number of alternative abstract plans. In contrast to our approach, all the previous approaches are not able to handle time-dependent QoS attributes.

Temporal properties have been considered by some works when selecting the best service composition. In [15], 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. In addition, the proposed approach does not guarantee that the optimal solution will be found. Wagner et al. [7] 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 according to the QoS values at each time period. In [16], Klopper et al. take into consideration time-dependent QoS values when selecting the best service instances. In this work, authors suppose that all QoS attributes are monotonically decreasing. Moreover, these works do not provide any search space reduction techniques prior to the selection process.

VII. Conclusion

In this paper, we presented a novel service selection approach taht allows the selection of the optimal solution (i.e., the best combination of services) while taking into account time-dependent QoS values. The proposed approach relies on two search space reduction mechanisms: QoS constraints and temporal constraints based Pruning. To reduce the number of candidate services based on QoS attributes, we proposed a set of formulas that allows computing local QoS thresholds while considering several categories of QoS attributes. In addition, we introduced a constraint optimization model to compute local time thresholds for each business task. Our evaluations show a significant gain in performance when applying our approach which scales better than the traditionally algorithm where all candidate services are considered. The accuracy of the proposed approach is more obvious with complex selection problems where the number of candidate services is very high. In the future work, we plan to consider further constraints at business level (e.g., temporal dependencies between business tasks, local temporal constraints) and investigate correlations between services when pruning uninteresting service instances. We also aim to study more complex composition structures and evaluate our approach based on real world scenarios.

References

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

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

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

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

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

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

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

[8] L. Chen, J. Yang, and L. Zhang, "Time based qos modeling and prediction for web services," in *ICSOC*, 2011, pp. 532–540.

[9] S. Son and K. M. Sim, "A price- and-time-slot-negotiation mechanism for cloud service reservations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 3, pp. 713–728, 2012.

[10] M. C. Jaeger, G. Rojec-goldmann, and G. Muehl, "Qos aggregation for web service composition using workflow patterns," in *EDOC*. IEEE CS Press, 2004, pp. 149–159.

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

[12] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *TWEB*, vol. 1, no. 1, 2007.

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

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

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

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