Turkish Journal of Computer and Mathematics Education *Vol.13No.03 (2022),1173-1185*

Research Article

GSS : Global Services Selection approach

Mansour Mekour[a], Abdelkrim Houacine[a], Abdelkader Mostefai[b]

mansour.mekour@univ-saida.dz[a], abdelkrim.houacine@univ-saida.dz[a], abdelkader.mostefai@univ-saida.dz[b]

[a]LGACA Laboratory, University of Saida, Dr. Taher Moulay, Algeria
[b]Department of Computer Science, University of Saida, Dr. Taher Moulay, Algeria

Article History: Received: 13 March 2022, Accepted: 31 May 2022, Available online 6 June 2022.

Abstract: The most well-known use of service-oriented architectures has given rise to several challenging research issues. One of these is the ability to recursively create a composite service as a process of other already available services, which are created by various companies and offer a variety of functionalities. Due to the ever-growing number of functionally identical services supplied on the web, it is crucial to identify them using well defined non-functional criteria. This needs an efficient method of selecting such services to compose. In this paper, adopting a service process and directed acyclic graph, we present an efficient and scalable approach to address the issue. Services are chosen for their composition based on both positive and negative QoS qualities, aiming to optimize user satisfaction. Experimental results show that our proposal scales very well regarding the huge amount of candidate services. It is less complex, and it depends linearly on the services implied in the composition.

Keywords: Service, Selection, Composition, QoS, Global Optimization.

1. Introduction

Distributed applications like e-business and enterprise systems are increasingly using the Service Oriented Architecture (SOA) as a major software foundation. Distributed applications can be dynamically and flexibly assembled by combining new and existing component services that have been independently integrated to create complex business processes and interactions. As long as the functional interface specification of each component service is correctly specified in WSDL and corresponds to a service request, it can be executed on any platform. Thus, with the widespread proliferation of web services, Quality of Service (QoS) is becoming increasingly crucial in determining the success of service providers. QoS refers to a number of non-functional characteristics including response time, throughput, availability, and reliability. Different levels of QoS can be used to provide services with comparable and compatible operation, so that one of the value-added services can be created; which requires making good decisions. In this research area, authors make every effort to improve QoS when using the service composition technique to construct a new composite service or to enhance an already existing compositions. In this paper, we suggest a brand-new, efficient method to optimize performance time of service composition and selection. The proposed solution enables also a flexible-behavioral selection, integration and interleaving of web services [44, 45]. The behavioral selection, integration and interleaving of web services are not addressed here in this current paper. The suggested algorithm's complexity is independent to the number of web services, which is always increasing. Hence, this approach is scalable and less complex, particularly for dynamic service compositions. The experimental results, using tow real world datasets with huge amount of candidate services show this advantage as well. A related works are reviewed in Section 2. Section 3 the problem is defined. The GSS approch is detailed and formalised in Section 4, followed by the candidate services selection algorithm in Sub-Section 4.2. The algorithm is then introduced and with large datasets is expremetally evaluated in Section 5. Finally, We come to conclusion and outline our future directions in Section 6.

2. Related Works

In the services composition topic, the QoS-based selection of Web services is an important study area. These research can be categorised in two main classes:

a) **Exact algorithms class:** This class transform optimizing the QoS −aware service composition ($O-QSC$) problem to a known model with the intention of using already available tools:

Zeng et al in [1] present an extensible QoS model based on integer programming. Ardagna and Pernici in [2] extend linear programming to support the local constraints. In order to adapt to distributed systems, Alrifai et al. in [3] address the issue by local and global optimization methods hybridization. For each abstract service, local selection is used to locate the best services that satisfy these local requirements after using mixed integer programming (MIP) to find the best global QoS constraints decomposition into local ones. When the problem is small, such linear programming techniques work quite well. A QoS broker-based process model for web service composition was proposed by the authors in [4]. The concept ensures for consumers the quality of delivered service and also addresses issues like up to date services and publishing only QoS in the registry. In [5], Zou et al. suggeste a planning-based strategy to transform a composition problem with QoS supports to a planning one with temporal and numerical characteristics. In [6] a systematic approach is proposed to calculate QoS for composite services, taking into consideration probability of the execution paths. A selection procedure for user preference satisfaction is proposed by Haddad et al. in [7], expressed as weights over QoS criteria to describe transactional requirements semantically. Although the research in [8] also present the transactional QoS-driven approach, in which the QoS-driven service selection is local optimized and placed after the transactional service selection.

However, such optimization techniques and methods are impractical for real-time and dynamic applications because of thier exponential algorithmic complexity.

b) Approximate algorithms class: In this class, extensive studies on QoS-aware service composition has been proposed, and various heuristic and metaheuristic techniques for the problem are suggested in order to find a near to optimal solution [9-13] . The genetic algorithms are also broadly used and improved by many other authors for QoS-aware service composition [14-22]. Such algorithms ignore possible assignments of several tasks in combination to decrease the time complexity. In [9] a heuristic algorithm is sugested to find a sub-optimal solution. In order to manage services in grid systems with adaptive management and QoS support, an adaptive method is suggested [10]. In [15], a dynamic and penalized fitness function is adopted, and an integer array is used to represent the genomic structure of a genetic algorithm. Where, each entry in the array refers to a web service, and a two-point crossover operator based mutation enabals to substitutes the associated web service with a different one from the pool of available services. Mardukhi et al. in [21] propose a top-down approach based genetic algorithm for global constraints decomposition into local ones. Then, for each abstract service, an optimal one is selected. The genetic-based algorithms are also reported in [17, 23, 24] which mainly made variations on the coding scheme, QoS model, fitness function, or handling techniques of population diversity. In [25], Wu and Zhu introduce the transactional characteristics of services and focus on how to compose and select component services transactionally. The problem is transformed into a constrained directed acyclic graph based on the search for the best path satisfying the global constraints, then a sub-optimal solution is provided using the ant colony optimization algorithm [26]. In [27, 28] a multi-objective and bee colony algorithms are suggested to obtain the best QoS value of a sub-optimal solution. Wada et al in [19-31] propose an optimization framework to address the O-QSC problem by leveraging a multi-objective algorithms. Swarm and Memetic optimization algorithms are also adopted [32-34] for Services scheduling, and also for the Internet of Things Applications composition. Eagle strategies are investigated in [35, 36] for QoS-aware cloud service composition. The O-QSC is also addressed in [37-42]. Ma et al In [43] have proposed the CDP "compositional decision making process" approach for searching of Pareto-optimal composition solutions in QSCRS "QoS-driven service composition with reconfigurable services".

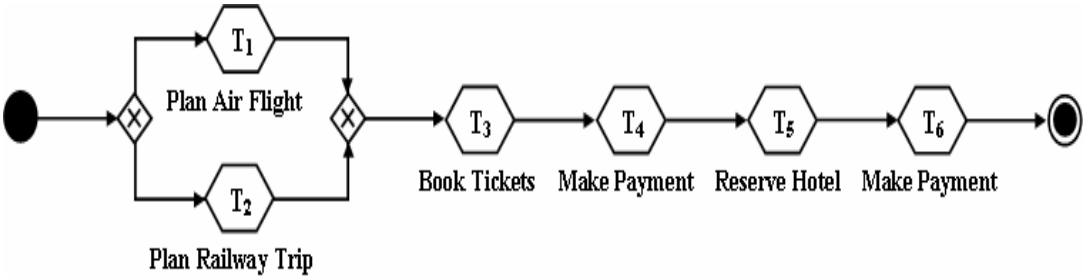
However, the drawback of those approaches is that either don't scale well regarding the huge amount of offered services or either don't find the beste optimale solution effeciently. Our proposal raise the efficiency and utility of services selection for this bouth aspects. On the other hand it enables a flexible-behavioral selection, integration and interleaving of composite web services [44, 45]. The behavioral selection, integration and interleaving of web services are not addressed here in the present paper.

3. $Q - WSC$ Statement

On the basis of a prespecified business process, service composition aims to select and connect services provided by various service providers. A workflow for web services, which specifies potential data dependencies

between tasks, can be used to represent the business process of the composite service. The most commonly-used control structures for service orchestration include: *sequence, choice, parallel and loop*. For example, Fig. 1 illustrates a workflow representing a *Make a Trip* composite service. In this composite service, a *Plan Air Flight* or a *Plan Railway Trip* service is performed. After that, services *Book Tickets*, *Make Payment*, *Reserve Hotel* and *Make Payment* are performed sequentially one after the other.

Fig. 1 Make a Trip Composite service example



Moreover, QoS is an essential aspect that describe the reliability and utility of a web service, and it is a primary key for the composition and the dynamic scheduling of provided services. Due to the ever-growing availability of functionally identical services on the web, it is necessary to be able to identify them using a set of precise QoS attributs. The quality model in this study is based on a collection of QoS that can be divided into negative and positive attributes. The values of negative properties require to be minimized (e.g. Price and response time etc.). Whereas, availability, throughput, and other positive attributes are to be maximized. Our solution for the $O - QSC$ problem is formulated based on the following definitions.

Definition 1 (Atomic service) *s_{ij} is a component service of a composite one associated with a set of QoS parametres.*

Definition 2 (Service Class) *(SC_i) encompass all atomic services collection that have the same functionality with nonfunctional properties (QoS), and are candidate for being associated to task T_i.*

Definition 3 (Quality array) *Q is a three dimensional array, $(n * m * n_i)$ where $(i = \overline{1,n})$, in which the entry Q_{ikj} defines the quality-property k value (such as price) for service j which is associated to task i, n is the tasks number (service classes), m is the quality-properties number, and n_i is size of the service classe i.*

Definition 4 (Weight matrix) *W is a matrix $(n * m)$ in which the entry W_{ik} defines weight of quality-property k regarding user's desires for the task i.*

Lemma 1 (quality-property weight) *If W_{ik} is the weight of quality-property k regarding user's preferences for the task T_i, then $\sum_k^m W_{ik} = 1$.*

Definition 5 (utility Function) *Each candidate service has a utility function F that is which is defined by a set of QoS attributes such as response time, availability, reliability, etc.*

Definition 6 (Abstract WorkFlow) *(AWF) represents the service structure that shows how several tasks are combined to provide that composite service.*

Definition 7 (Service Candidate Graph) *(SCG) is a directed acyclic graph built from AWF and S_i to represent all paths for all combinations of candidate services that can be used to fulfull the required task.*

4. GSS Optimization

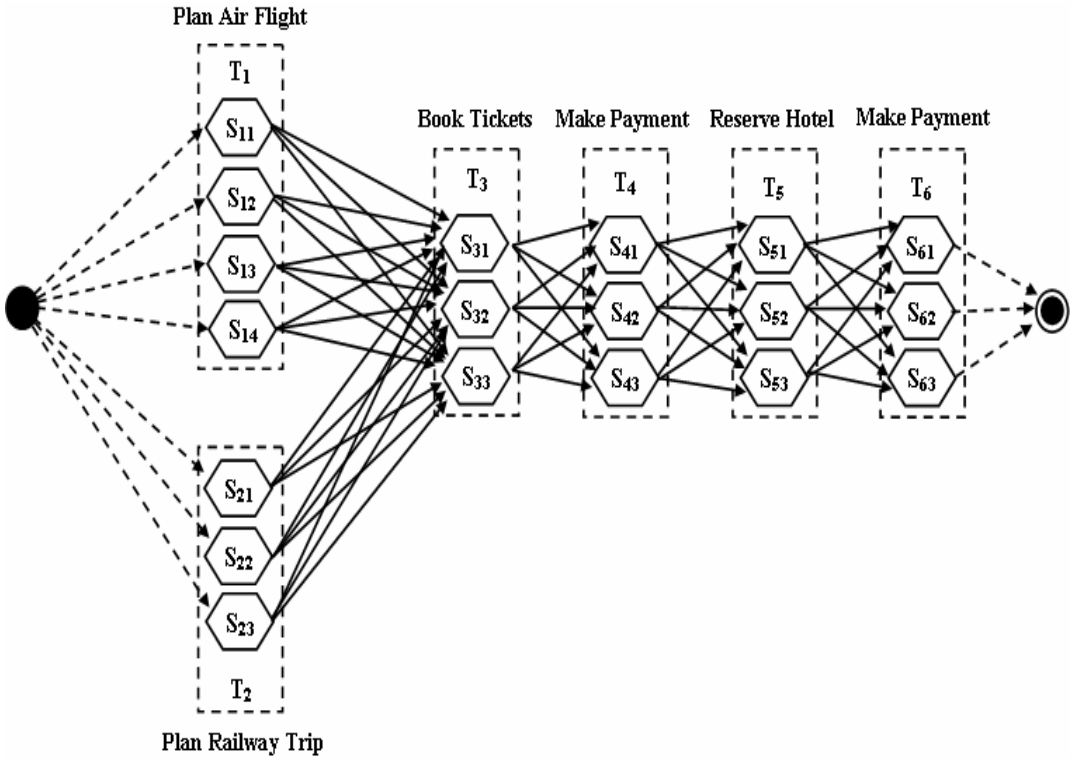
Given the workflow AWF of composite service CS, quality array Q and weights of quality attributes W, the O-QSC problem is the best combination of services which subject to all local and global QoS constraints on one hand, and optimizes the objective value of the composite service CS on the other hand. We can distinguish tow approaches for web service selection: local selection approach and global selection approach. Although very

efficient, local selection is a linear, $O(l)$, but it can't handle global QoS requirements. On the other hand, global optimization, while it can support global constraints regards, is impractical for real-time and dynamic applications due to its poor performance. Nevertheless, based on global approach, the $O-QSC$ is known as NP-hard problem [1, 15, 19]. Here, based on directed acyclic graph and abstract service, we propose our solution to tackle this problem.

4.1 GSS model

Given a workflow that has n tasks. we refer by S_i $(i = \overline{1,n})$ to the set of candidate services that can fulfill the i^{th} task functionality in the workflow. Furthermore, s_{ij} $(j = \overline{1,n_i})$ is used to represent the j^{th} candidate service in S_i. The directed acyclic graph G(C,E) describe the service composition problem, where C represent the starting and sink nodes, plus of course the availabals candidate services. E is the set of edges that connect candidate services from two adjacent sets S_i and S_{i+1}. The problem then becomes the best path search from the starting node to the sink node that satisfies all global requirements.

Fig. 2 SCG of the full Workflow



However, with respect to the considerable number of task to be realized and the huge amount of available candidate services associated to these tasks the O-QSC problem is becoming much more difficult. This problem is known as MMKP (multi dimensional multi objective knapsack) problem that has been proved to be NP-complete [9, 49]. To tackle this problem several approaches have been proposed (such as linear programming, heuristic, evolutionary algorithms, etc). In our solution, we adopt a locale approach for web services selection. Furthermore, to overcome the drawback of local strategy selection, we consider an abstratcs services in the workflow AWF, then, we propose our GSS algorithm below (see algorithm shown in Fig. 3). Then, candidate services for each task in an AWF are modeled as an acyclic directed graph (see Fig. 4). After that, the problem of finding the best path then is relaxed as best component service selection for each class i.

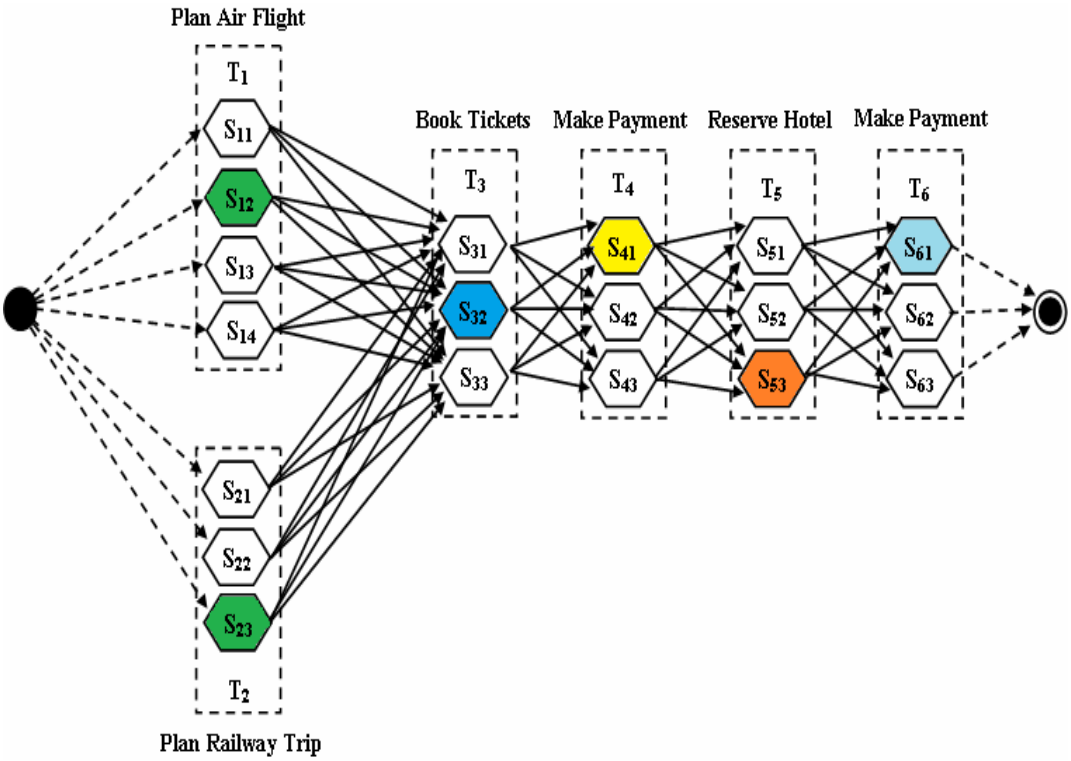
Fig. 3 Global Services Selection

Algorithm 1: *GSS* Algorithm

Inputs : AWF, CSC ; /* AWF is an Abstract Work Flow, CSC is a Candidate Service Class */
Output: CWF ; /* CWF is a Concrete Work Flow */

1 **for All** T_i **In** AWF **do**
2 **for All** Q **Of** T_i **do**
3 $Q_{ij} = 0$;
4 **for All** S_{ij} **In** C_i **do**
5 $Q^{max} = MaxQ_{ikj}$;
6 $Q^{min} = MinQ_{ikj}$;
7 **if** Q *is Positif QoS* **then**
8 $Q = Q^{min}$;
9 **else**
10 $Q = Q^{max}$;
11 **if** $Q^{max} \neq Q^{min}$ **then**
12 $Q_{ikj} = \frac{|Q - Q_{ik}|}{Q^{max} - Q^{min}}$;
13 **else**
14 $Q_{ikj} = 1$;
15 $Q_{ij} += W_{ik} * Q_{ikj}$;
16 Associate(T_i,$MaxQ_{ij}$); /* Associate is a function to bind the task T_i to the service S_{ij}, that have the Max Value for Q_{ij}, of the Class C_i */

Fig. 4 Selected services graph



4.2 Candidate services selection

Algorithms for selecting services aim pick componet services that achives to QoS requirements and also offer the best value for the user-defined objective function. A weighted average of the positive and negative QoS attributes enabals to define the utility function. Suppose a user has k quality of service attributes in a service request, Q_{ikj} ($i = \overline{1,n}$; $k = \overline{1,m}$; $j = \overline{1,n_i}$). In our solution, the utility function F is defined as follows:

definition 8 *A utility function* (F). *Suppose there are a set of (positive and negative) QoS attributes. the positive ones to be maximized and the negatives ones to be minimized. The utility function for service* s_{ij} *is defined as* $F = Q_{ij}$, *where :*

$$Q_{ij} = \sum_k W_{ik} * Q_{ikj} \tag{1}$$

and W_{ik} are the weights for each QoS attribute Q_{ik}, and Q_{ikj} is the k^{th} QoS attribute of the j^{th} service associed to the task i. In the utility function definition, all QoS attributes are normalized by the definition of following equation :

$$Q_{ikj} = \begin{cases} \frac{|Q_{ik} - Q_{ikj}|}{Q_{ik}^{max} - Q_{ik}^{min}} & if\, Q_{ik}^{max} \neq Q_{ik}^{min} \\ 1 & otherwise \end{cases} \tag{2}$$

Where Q_{ik} is defined as follow:

$$Q_{ik} = \begin{cases} Q_{ik}^{max} & if\ Q_{ikj}\ \ is\ a\ negative\quad QoS \\ Q_{ik}^{min} & if\ Q_{ikj}\ \ is\ a\ positive\quad QoS \end{cases} \tag{3}$$

Q_{ik}^{max} and Q_{ik}^{min} are evaluated by the following equations respectively.

$$Q_{ik}^{max} = Max[Q_{ikj}] \tag{4}$$

$$Q_{ik}^{min} = Min[Q_{ikj}] \tag{5}$$

5. Empirical studies

The goal of this study is to evaluate the effectiveness and efficiency of our approach, that achieves optimal outcomes with a significantly shorter computation time than the existing global and local optimization approaches. The global and locale approach are used as benchmark for the optimality and computing cost aspect respectively. In the following, we refer to our solution as GSS., the labels BF and MIP to refer to the global approaches based on the brute-force search and the standard mixed integer algorithm respectively, and the label Local to refer to the local services selection approach [7, 25, 21]. We have conducted several tests, which are discussed in this section, to evaluate the GSS efficiency and optimality.

5.1 Evaluation methodology

We have created a variety of configurations of the $O - QSC$ problem. Each configuration is an abstract workflow with n tasks, $|S_i|$ candidate services per task T_i, and m QoS attributes. We generated a collection of abstract workflows by varying these numbers, with each individual combination of these factors indicating a different abstract workflow. We applied each configuration test several times with various numbers of QoS attributes attributes in order to explore the effects of the number of QoS attributes in our proposal, and then we

computed the computation cost of the GSS algorithm. On the other hand, we consider this abstract workflows collection to compare our GSS approach to Local, BF, and MIP algorithms. The first is a Locale-based algorithm [7, 25], and the two last are a Global-based one [21]. These algorithms are exact optimization approach for a $O-QSC$ problem [25, 21]. Indeed, we applied the Local and GSS optimization for each configuration in the collection test, and then, we record the obtained computation time and the utility values. Then, for the same collection test, we applied the BF and MIP algorithms, to record the obtained computation time and the global utility value. Based on the formulas presented in the section 4.2, and the obtained utility values, we compute the optimality of GSS and Local algorithms in relation to MIP algorithm.

5.2 The Dataset and Experiment settings

We have used two data sources in this experimentation. The QWS real world dataset [46] is used to study the computing cost and optimality of the GSS algorithm compared to the Local, BF, and MPI algorithms. Indeed, the QWS data set includes measurements of 2 QoS attributes (Throughput and Response Time) for 5 825 real web services. As the number of QoS attributes is very limited in this QWS dataset (2 QoS attributes), we also have used another real world data set to study the effect of the number of QoS attributes in the GSS algorithm selection. The second QWS real word dataset includes measurements of 9 QoS attributes (Availability, Response Time, Throughput, Reliability, Successability, Compliance, Latency, Best Practices, and Documentation) for 2 507 real Web services that exist on the Web [46-48]. For the implementation aspect, we used the version 5.5 of the open source Lp-Solve system [19] for implementing the MIP algorithm, while the Local, BF, and GSS algorithms were implemented by Java. The experiments were run on an Intel i7 with 2.40 GHz processor and 4GB of RAM. The computer is using Windows 10.

5.3 Analysis and Results

Scalability of service selection systems is influenced by the time complexity of the used algorithm. The number of tasks in the composition, the number of service candidates per task, and the number of QoS attributes are often used to determine the size of the O-QSC. As the $O-QSC$ can be modeled as a $MMKP$, which is known as NP- hard problem [9, 49], For a few hundred of potential candidate services that offer the same functionality, the computing time of any global exact-methode may exceed the run-time requirements [25, 21]. Such global optimization approaches traditionally address the $O-QSC$ as a standard mixed integer program, which makes them unsuitable and unsupported for $O-QSC$ problem, where the number of candidate services is big. In worst case, the MIP algorithm time-complexity is exponential [25, 21]. Although they are very scalable, and there are no exact or approximate approaches better than they, the existing local optimization solutions will not be supported at all as long as they have an issue of optimality. In our approach, we adopt an abstract Workflow to solve the $O-QSC$ problem. However, the local selection technique, which is highly effective and scalable, is used to select services for AWF. The local utility computation for the required task is specifically linear in complexity with respect to the number of service candidates. Furthermore, as service composition can perform the local selection for abstract tasks in a AWF independently, the total time complexity of such independent selection is not influenced by the number of tasks in the workflow; hence, the time complexity of such services selection remains linear. Consequently, the GSS time complexity is linearly dependent on the number of available services, and on the number of tasks in workflow is fully independent. Such kind of features makes our solution highly scalable than existing global strategy based exact-solutions. On the other hand, the optimality of the GSS approach is controlled and fixed based on the abstract Workflows strategy. A candidate services for an AWF are modeled by a fully meshed DAG. In the other words, the $O-QSC$ problem is not modeled by not-fully meshed DAG, which is known as NP-Hard, but it is conceptually modeled by a one DAG that is fully meshed in our solution. Such fully meshed DAG insures in GSS solution the optimality of provided solutions contrary to the existing local strategy based exact-solutions and global strategy based approximate-ones.

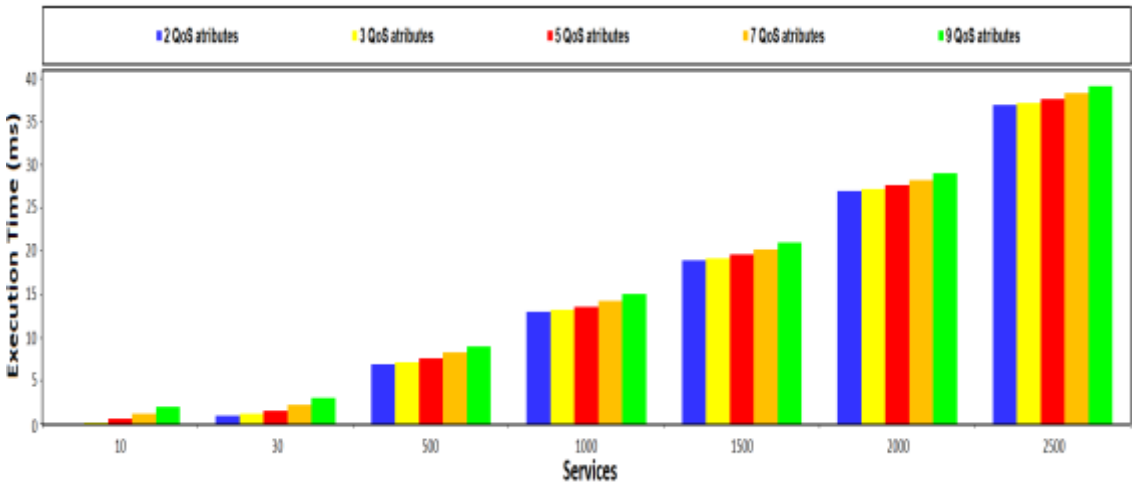
5.4 GSS performance and the impact of QoS attributes

Unlike the existing global approaches for $O-QSC$ problem, which are influenced by the number of QoS attributes involved by a service composition, the increasing of the number of QoS attributes in the GSS algorithm not increase the computation time. Indeed, the approximate approaches, usually, have such influence twice. When the number of QoS attributes is low, then they have a problem of optimality, and when it is a big,

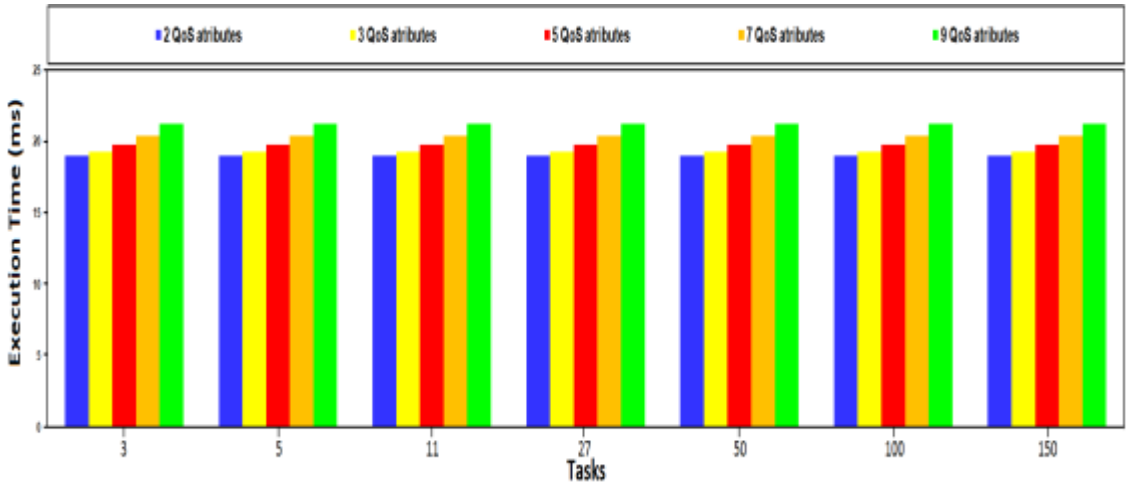
then they have a problem of performance cost. Also, with the increasing number of QoS attributes and tasks the computation time then is always the same in general. This is an expected behavior as we already discussed in section 4, because the GSS approach not depend neither the tasks number, nor the QoS attributes number, and the amount of candidate services affects it linearly. When there are more services, tasks, and QoS requirements, adopting GSS for O-QSC problems becomes more efficient. In Fig. 5.a, for each configuration, we study the performance of the GSS algorithm under the same number of tasks. We have used the QWS real world data set which has 9 QoS attributes. The range of service candidates for each task is 10 to 2500, while the number of tasks is fixed to 150. In this experiment, the computation times is measured for 2, 3, 5, 7 and 9 QoS attributes. The results indicate that the GSS approach significantly is very independent to QoS attributes number. By varying the number of QoS attributes, the required computation time of the GSS approach perhaps is the same (the difference is about 2 ns). On the other hand, we study also the performance of the algorithm, but in that case, with respect to the number of tasks implied in the workflow.

Fig. 5 The impact of QoS attributes in GSS algorithm

a) 150 Tasks



b) 1500 Candidate services

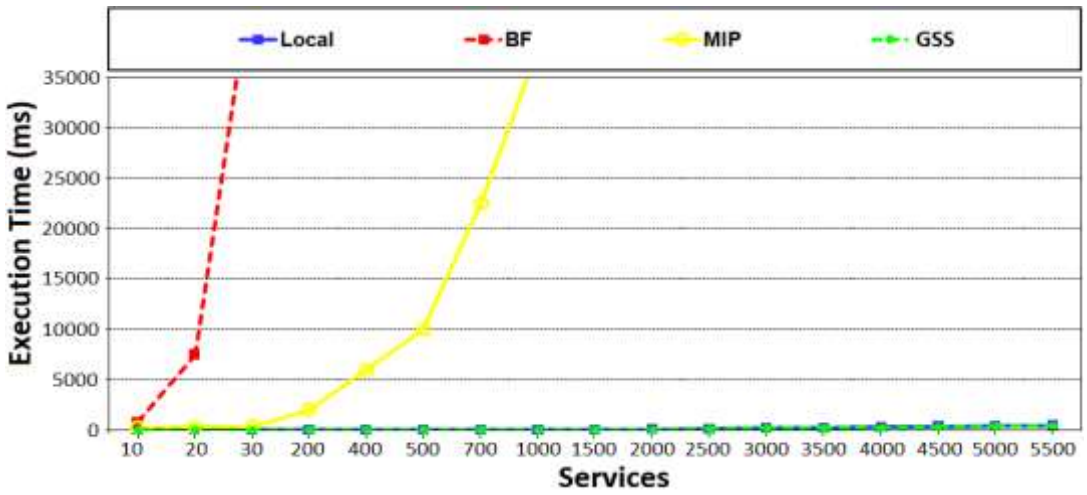


In the experiment shown by Fig. 5.b, we used the same QWS real world data set, and we have measured also the computation times for 2, 3, 5, 7 and 9 QoS attributes. The number of tasks is varied from 3 to 150 tasks, while the number of candidate services is fixed to 1 500 services. The results show, the computation time of GSS algorithm is also always the same not only for all QoS attributes, but also for all tasks number. Consequently, the GSS approach then is neither influenced by the amount of tasks, nor the amount of QoS attributes. This obviously will become more advantages for the real world, where the amount of tasks in WF, as well the amount of QoS attributes are very limited. The results shown by the figs. 5.a and 5.b indicate also that

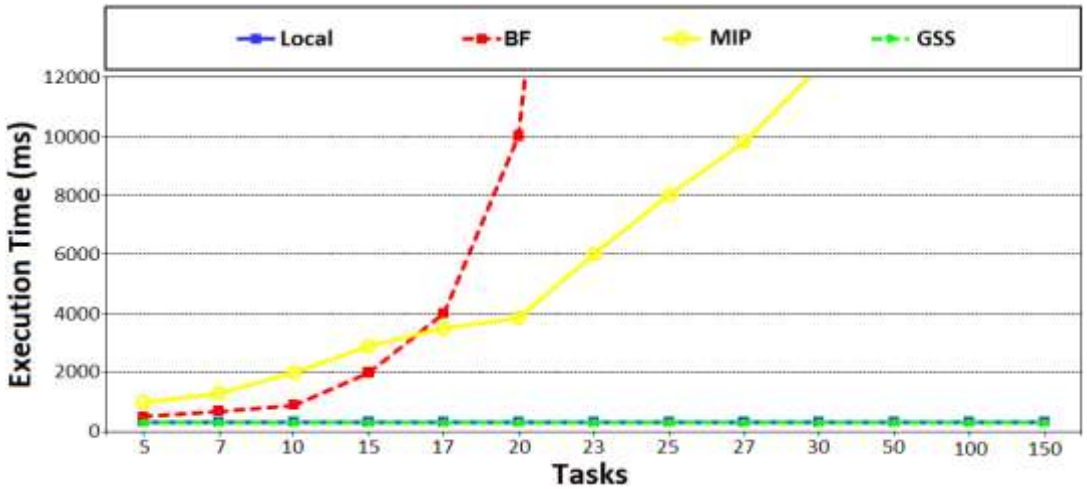
our solution performs very well regarding the huge number of services. For example, when the number of tasks is 150 tasks with 9 QoS attributes, and for 2500 candidate services per tasks (i.e. 375 000 services), the GSS algorithm find the solution in 39.08 ms (millisecond). Also, when the services number is 1500, the solution is constantly found within time of 21.22 ms.

Fig. 6 Algorithms Performance Results

a) 31 Tasks



b) 500 Candidate services



5.5 MIP, *GSS* and Local Performance Results

In Fig. 6.a, the computation times is measured for GSS, Local, BF, and MIP optimization algorithms for each configuration. We compare the performance of the algorithms under the same candidate services number. In this experiment, and for all test cases, we used the first QWS real world data set (where the number of QoS attributes is 2), the range of service candidates for each task is 10 to 5500, while the number of tasks is fixed to 31. The results indicate that the GSS performs significantly better than the BF and MIP algorithms. the GSS is particularly scalable because, in contrast to the MIP algorithm, the required computing time of the GSS approach increases relatively slowly with the number of service candidates. Indeed, the computation time is increased in the same manner as the Local algorithm increasing under the same number of candidate services per task, which say that our solution complexity is linear as the local one. In the experiment shown in Fig. 6.b, we study also the performance of the algorithms, but in that case, with respect to the number of tasks implied in the workflow. In this experiment, we used the first QWS real world data set, where the number of tasks is varied from 5 to 150,

and there are 500 service candidates available for each task. The results show that, in every test instance, our method consistently outperforms the global ones (BF and MIP algorithm).

5.6 Optimality

As the MIP algorithm is an exact algorithm, it has not the issue of optimality, and it outperforms the BF one, By analyzing the results of the GSS and Local algorithms with those of the MIP optimization algorithms, we have assessed the quality of the result given by these algorithms. By comparing the overall utility value obtained by the GSS and Local algorithms for the selected services to the overall utility gains made for the optimal selection given by the MIP algorithm, we conclude if the solutions of the GSS and Local algorithms are optimal. The GSS and $Local$ optimality is computed by the following formulas 6 and 7).

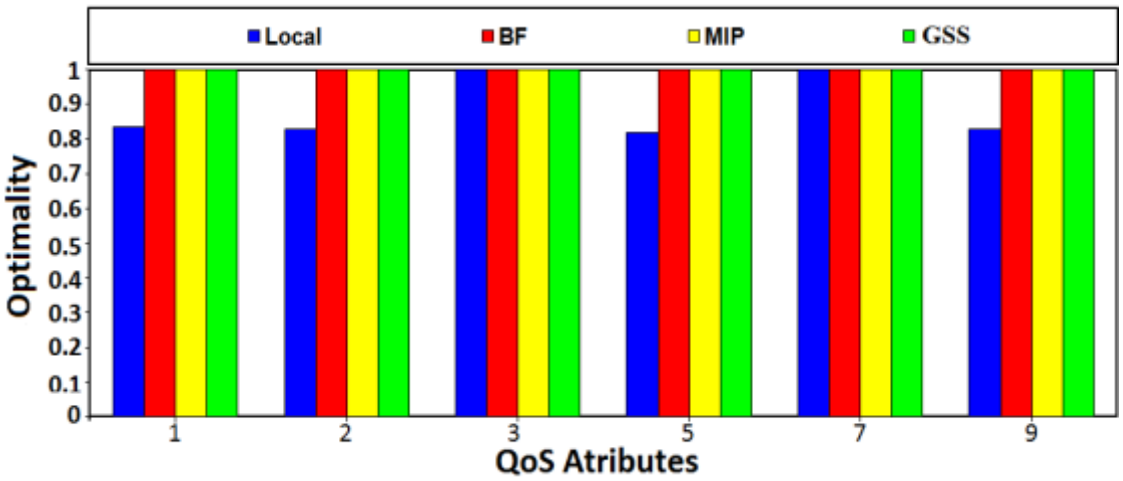
$$GSS\_Optimality = \frac{GSS\_UV}{MIP\_UV} \tag{6}$$

$$Local\_Optimality = \frac{Local\_UV}{MIP\_UV} \tag{7}$$

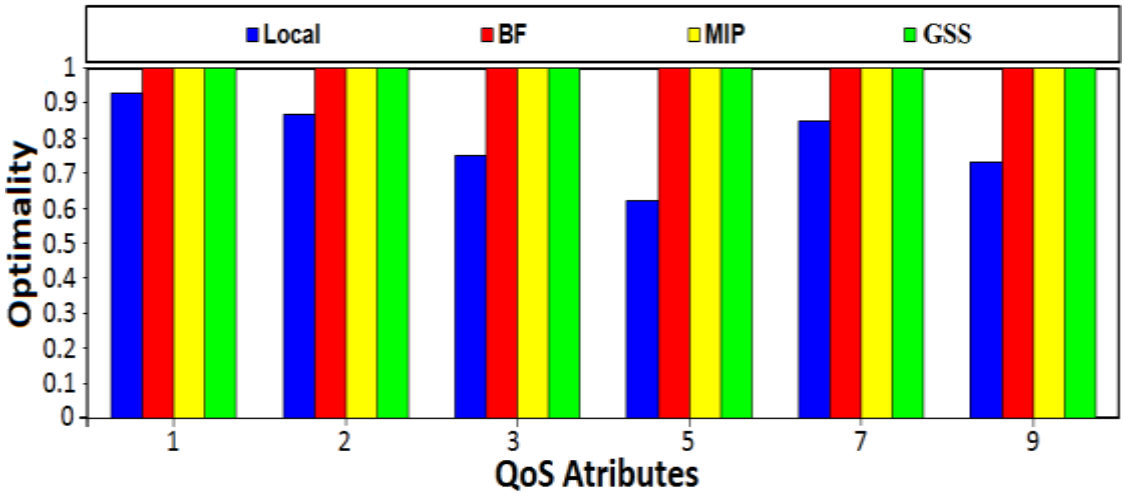
Where MIP_UV , GSS_UV, and $Local\_UV$ is the overall utility value obtained by MIP, GSS and $Local$ algorithm respectively.

Fig. 7 Algorithms Optimality

a) 51 Tasks



b) 2500 Candidate services



The achieved optimality is shown in Fig. 7.a in numerous configurations with varied numbers of candidates service, and in Fig. 7.b in several configurations with variable numbers of trasks. The results show, except two specific cases, that the Local algorithm was able to achieve only a bit higher than 0.8 and 0.6 as indicated by the figs. 7.a and 7.b respectively. While the GSS algorithm achieves exactly the same optimality that the MIP and BF algorithms achieves. The results also show that the Local algorithm achieves the same optimality of the MIP and BF algorithms in some test cases. These test cases are done on workflows that are constituted by sequential flows only, and from this specific case the idea was conceived.

6. Conclusion and future work

We described our scalable solution to the O-QSC problem in this paper. Unlike the other already existing works, which do not use the fuly mushed DAG (linear programming, heuristic, evolutionary algorithms, etc) to solve the issue. Our solution use such DAG to reduce the complexity. Firstly the global workflow is constituted by tasks called abstracte services. Then the best service for each task T_i in AWF is found through an efficient algorithm. The complexity of the proposed algorithm is linearly dependent with the increasing amount of services. As a result, this method is less complex and scalable, particularly for service composition. This benefit is further confirmed by the extensive experimental analysis utilizing two real-world datasets with a large amount of potential services. As was mentioned, as the number of tasks increases, the computation time of our proposal becomes a little bit more. This is because the suggested algorithm has some innate qualities. The huge amount of tasks in WF, however, is not an issue because it is extremely constrained and unchanging. Furthermore, the proposed approach helps to enhance the organization's applications integration based on services. Through a dynamic and flexible integration, software components (as a service) are efficiently located and dynamically bounded. Precisely, with an increase of such components, service quality is paramount in the application integration. Such integration must reflect the business needs of customers and providers so that to receive their confidence and achieve their goals. Also, the present approach opens a wide research on web service based applications. Further works will be done on the improvement of service selection performance and composition. In addition, other problems will be tackled such social and behavioral aspects.

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 Transactions on Software Engineering, 30 (5) (2004) 311-327.

[2] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Transactions on Software Engineering 33 (6) (2007) 369-384.

[3] M. Alrifai, T. Risse, W. Nejdl, A hybrid approach for efficient web service composition with end-to-end QoS constraints, ACM Transactions on the Web 6 (2) (2012) 1-31.

[4] M. Rathore, and U. Suman, "A Quality of Service Broker Based Process Model for Dynamic Web Service Composition". Journal of Computer Science, 7 (8) (2011) 1267-1274.

[5] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, Y. Xiang, QoS-Aware Dynamic Composition of Web Services Using Numerical Temporal Planning, IEEE Transactions On Services Computing, 7 (1) (2014) 18-31.
[6] H. Zheng, W. Zhao, J. Yang, A. Bouguettaya, QoS Analysis for Web Service Compositions with Complex Structures, IEEE Transactions On Services Computing, 6 (3) (2013) 373-386.
[7] J. El Hadad, M. Manouvrier, M. Rukoz, TQoS: transactional and QoS-aware selection algorithm for automatic web service composition, IEEE Transactions on Services Computing 3 (1) (2010) 73-85.
[8] Q. Zhu, Q. Wu, G. Dai, M. Zhou, An approach for transactional QoS-driven service composition, Journal of Computational Information Systems, 7 (10) (2011) 3398-3405.
[9] T. Yu, Y. Zhang, K.J. Lin, Efficient algorithms for web services selection with end-to-end QoS constraints, ACM Transactions on the Web, 1 (1) (2007) 1-26.
[10] J.Z. Luo, J.Y Zhou, Z.A. Wu, "An adaptive algorithm for QoS-aware service composition in grid environments". Service Oriented Computing and Applications, 3 (3) (2009) 217-226.
[11] A. RamÃ-rez, J. Parejo, J. Romero, S. Segura, A. CortÃ©s. "Evolutionary composition of QoS-aware web services: A many-objective perspective". Expert Systems with Applications, 72 (2017) 357-370.
[12] C. Li, J. Li; H. Chen. "A Meta-Heuristic-Based Approach for Qos-Aware Service Composition". IEEE Access, 8 (2020) 69579-69592.
[13] H. Naghavipour, M. Idris. "Hybrid Metaheuristics Using Rough Sets for QoS-Aware Service Composition". IEEE Access, 10 (2022) 112609-112628.
[14] Y. Ma, C. Zhang, Quick convergence of genetic algorithm for QoS-driven web service selection, Computer Networks 52 (5) (2008) 1093-1104.
[15] G. Canfora, D.I. Penta, M.R. Esposito, M.L. Villani, A framework for QoS-aware binding and re-binding of composite web services, Journal of Systems and Software, 81 (10) (2008) 1754-1769.
[16] L. Ai, M. Tang, C. Fidge, Partitioning composite web services for decentralized execution using a genetic algorithm, Future Generation Computer Systems 27 (2) (2011) 157-172.
[17] X.L. Wang, Z. Jing, H.Z. Yang, Service selection constraint model and optimization algorithm for web service composition, Journal of Information Technology 10 (5) (2011) 1024-1030.
[18] Q.Wu, Q. Zhu, P. Li, A caching mechanism for QoS-aware service composition, Journal of Web Engineering 11 (2) (2012) 119-130.
[19] H. Wada, J. Suzuki, Y. Yamano, K. Oba, E ^{3} : A Multiobjective Optimization Framework for SLA-Aware Service Composition, IEEE Transactions On Services Computing, 5 (3) (2012) 358-372.
[20] D. Palanikkumar, G. Kousalya, An evolutionary algorithmic approach based optimal web service selection for composition with quality of service, Journalof Computer Science, 8 (4) (2012) 573-578.
[21] F. Mardukhi, N. NematBakhsh, K. Zamanifar, A. Barati, QoS decomposition for service composition using genetic algorithm, Applied Soft Computing 13 (7) (2013) 3409-3421.
[22] D. Wang, Y. Yang, Z Mi. "A genetic-based approach to web service composition in geo-distributed cloud environment". Computers & Electrical Engineering, 43 (2015) 129-141.
[23] C. Jatoth, G. Gangadharan, U. Fiore, R. Buyya. "QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation". Future Generation Computer Systems, 86 (2018) 1008-1018.
[24] C. Jatoth, G.R. Gangadharan, R. Buyya. Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm. Future Generation Computer Systems. 94 (2019) 185-198.
[25] Q. Wu, Q. Zhu, Transactional and QoS-aware dynamic service composition based on ant colony optimization, Future Generation Computer Systems, 29 (5) (2013) 1112-1119.
[26] Q. Yu L. Chen B. Li. "Ant colony optimization applied to web service compositions in cloud computing". Computers & Electrical Engineering, 41 (2015) 18-27.
[27] S. Zhang, S. Xu, X. Huang, W. Zhang, M. Chen. Networked correlation-aware manufacturing service supply chain optimization using an extended artificial bee colony algorithm. Applied Soft Computing Journal. 76 (2019), 121-139.
[28] F. Seghir. "FDMOABC: Fuzzy Discrete Multi-Objective Artificial Bee Colony approach for solving the non-deterministic QoS-driven web service composition problem". Expert Systems with Applications, 167 (2021) 114413.
[29] M. Cremene, M. Suciu, D. Pallez, D.Dumitrescu. "Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition". Applied Soft Computing, 39 (2016) 124-139.
[30] F. Chen, R. Dou, M. Li, H. Wu. "A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing". Computers & Industrial Engineering, 99 (2016) 423-431.
[31] N. Kashyap, A. Kumari, R. Chhikara. "Multi-objective Optimization using NSGA II for service composition in IoT". Procedia Computer Science, 167 (2020) 1928-1933.

[32] X. Xu, H. Rong, E. Pereira. "Predatory Search-based Chaos Turbo Particle Swarm Optimisation (PS-CTPSO): A new particle swarm optimisation algorithm for Web service combination problems". Future Generation Computer Systems, 89 (2018) 375-386.
[33] C. Li, J Li, H. Chen, A. Heidari. "Memetic Harris Hawks Optimization: Developments and perspectives on project scheduling and QoS-aware web service composition". Expert Systems with Applications, 171 (2021) 114529.
[34] B. Benmessahel, F. Nouioua. "DDAPSO: Hybrid Discrete Dragonfly Algorithm and Particle Swarm Algorithm to Service Selection and Composition for the Internet of Things Applications". Revue d'Intelligence Artificielle, 36 (3) (2022) 417-425.
[35] S. Gavvala, C. Jatoth, G. Gangadharan, R. Buyya. Huang. "QoS-aware cloud service composition using eagle strategy". Future Generation Computer Systems, 90 (2019) 273-290.
[36] H. Jin, S. Lv, Z. Yang, Y. Liu. Huang. "Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition". Applied Soft Computing, 114 (2022) 108053.
[37] H. Wang, W.Chiu, S. Wu. "QoS-driven selection of web service considering group preference". Computer Networks, 93 (1) (2015) 111-124.
[38] A. Mousa J. Bentahar. "An Efficient QoS-aware Web Services Selection Using Social Spider Algorithm". Procedia Computer Science, 94 (2016) 176-182.
[39] S. Wang, L. Huang, L. Sun, C. Hsu, F. Yang. "Efficient and reliable service selection for heterogeneous distributed software systems". Future Generation Computer Systems, 74 (2017) 158-167.
[40] W. Serrai, A. Abdelli, L. Mokdad, Y. Hammal. "Towards an efficient and a more accurate web service selection using MCDM methods". Journal of Computational Science, 74 (2017) 253-267.
[41] H. Wang, D. Yang, Q. Yu, Y. Tao. "Integrating modified cuckoo algorithm and creditability evaluation for QoS-aware service composition". Knowledge-Based Systems, 140 (2018) 64-81.
[42] L. Zhao, W. Tan, N. Xie, L. Huang. "An optimal service selection approach for service-oriented business collaboration using crowd-based cooperative computing". Applied Soft Computing, 92 (2020) 106270.
[43] H. Ma, F. Bastani, I-L. Yen, H. Mei, QoS-Driven Service Composition with Reconfigurable Services, 6 (1) (2013) 20-34.
[44] M. Mekour and S. M. Benslimane, BRC: Behavior Reconfiguration and Combination to Enhance the Dynamic Semantic Web Services Composition, Seventh International Conference on Next Generation Web Services Practices, (2011) 410-415.
[45] M. Mekour and S. M. Benslimane, SP4PS: service process rewriting for efficient and proper web services composition, International Journal of Web Engineering and Technology, 8 (4) (2013) 327-346 .
[46] Z. Zheng, Y. Zhang, and M. R. Lyu, Investigating QoS of Real- World Web Services, IEEE Transactions on Services Computing, 7(1), 2014, pp. 32-39.
[47] E.Al-Masri, and Q. H.Mahmoud, QoS-based Discovery and Ranking of Web Services, IEEE 16th International Conference on Computer Communications and Networks (ICCCN), 2007, pp. 529-534.
[48] E. Al-Masri, and Q.H. Mahmoud, Investigating web services on the world wide web, in: proceeding of the 17th international conference on World Wide Web, WWW08, ACM, 2008, pp. 795-804.
[49] S. Martello and P. Toth, Algorithms for Knapsack problems, Discrete Math, 31 (1987) 70-79.