*Journal of **AI** and **D**ata **M**ining*
Vol 7, No 1, 2019, 181-190

DOI: 10.22044/JADM.2017.5487.1659

Semantic Constraint and QoS-Aware Large-Scale Web Service Composition

E. Shahsavari and S. Emadi*

Department of Computer Engineering, Yazd Branch, Islamic Azad University, Yazd, Iran.

Received 07 March 2017; Revised 06 March 2018; Accepted 02 June 2018
**Corresponding author: emadi@iauyazd.ac.ir(S. Emadi).*

Abstract

Service-oriented architecture and the related technologies facilitate the running time of interactions using business integration on the networks such as the Internet. Currently, web services are considered as the best option to provide Internet services. Due to the growth and an increasing number of Web users and the complexity of users' queries, simple and atomic services are not capable of meeting the needs of users, and to provide complex services, they require service composition. The Web service composition as an effective approach to the integration of business institutions' plans has taken significant acceleration. Nowadays, Web services are created and updated in a moment. Therefore, in the real world, there are many services that may not have composability according to the conditions and constraints of the user's preferred choice. In the proposed method for automatic service composition, the main requirements of users including the available inputs, expected outputs, quality of service, and priority are initially and explicitly specified by the user, and service composition is done with this information. In the proposed approach, due to a large number of services with the same functionality, at first, the candidate services are reduced by the quality of service-based Skyline method, and moreover, using an algorithm based on graph search, all the possible solutions will be produced. Finally, the user's semantic constraints are applied on service composition, and the best composition is offered according to the user's requests. The result of this work shows that the proposed method is more scalable and efficient, and it offers a better solution by considering the user's semantic constraints.

Keywords: *Service Composition, Web Services, Skyline Services, Qualitative Parameters, Service, Graph Search.*

1- Introduction

Service-oriented computing (SOC) is a new computing phenomenon that uses service as a basic structure to support the rapid, low-cost, interoperable, and viable development and simple combination of distributed applications, especially in heterogeneous environments. This type of composition prefers the network service components that are easily connected with each other to create dynamic and flexible business processes [1,2]. Service-oriented architecture (SOA) is the basic architectural model of this method that supports it in terms of architecture [3]. Therefore, SOC is a computing phenomenon, and SOA is an architectural model at a conceptual level. This means that they should be implemented by special technologies. The Web service technology is an example that is widely accepted and looks very promising. Web services are well-defined modules independent from platform; they are reusable and provide a standard performance for business. Web services can be published, discovered, located, called, and connected freely across the web and facilitate the integration of new structured applications both within and around the organizational boundaries [4].

With the advent and development of SOA and Web service technology, many companies and organizations provide their business plans in the form of well-defined services. Although these atomic Web services that are often simple can improve reusability and simplify the application logic, they can only meet a limited functionality. They cannot meet the personal and diverse requirements of users according to their requests or reflect complex business processes. Therefore, Web service compositions are very important for the development of applications [5, 22]. The ability of optimal and effective selection and integration of inter-organizational services is one of the challenges of the today's world.

Web services are created and updated in a moment. This is beyond our ability to analyze them and produce a composition plan manually [5]. Many methods have been proposed for Web Service Composition (WSC). Most of them are only based upon the conformity of input and output parameters, while some constraints are posed by the users. Service constraints are used to ensure the proper implementation of the service or its interaction with other services. As a result, they have a significant impact on service composition. The user constraints are often ignored by the existing methods of Web service composition. For example, the user's membership level in online shopping scenario, the total amount of the order, and the credit card can affect the service composition so that various amounts of each one of these cases may lead to the different selections of service at a later stage of composition [6].

However, in the real world, there are a lot of services that may not have composition capabilities according to the conditions and constraints of the users that can be defined as semantic. Thus a numerous number of Web services, permanent update, and different manufacturers with different models and objectives have made the Web service composition difficult.

Methods that have been proposed for WSC only have used service qualitative parameters in order to select the optimal composition, and the users' constraints are often ignored, while considering the users' constraints in the composition process may lead to more satisfied users. Furthermore, it is important that the service composition is conducted fast, especially in interactive systems in which long delays are unacceptable. Despite a large number of services in the real world, the use of methods that have been proposed is very time-consuming because the search space used to select the candidate service is large. Therefore, we can say that these methods are not very scalable.

In the proposed method, the Wang et al.'s algorithm [6], which uses a graph-searching algorithm to combine the services, is improved. In his research work, Wang has used only the constraints on services that serve as a parameter for evaluating the combined solution, and does not take into account the quality of services.

However, in the proposed method, the three parameters cost service quality, response time, and parametric reliability are used to evaluate the hybrid solution. Additionally, the Wang et al.'s approach is not scalable. In this work, using the Skyline service algorithm, the scalability of the method has been improved.

In the method proposed for automatic service composition, the basic needs of users including the available inputs, expected outputs, quality of service, and priority are initially and explicitly specified by the user, and service composition is done with this information.

The proposed solution includes an algorithm based on a search graph. The graph search algorithm is applicable to the general concept of Web service composition and produces all possible solutions. The preferences of users, as openly requested in their queries, are considered in service composition.

In the proposed approach, first, the candidate services are reduced by the quality of the service-based Skyline method, and moreover, using an algorithm based on graph search, all the possible solutions will be produced. Finally, the user's semantic constraints are applied on service composition, and the best composition will be offered according to user's requests.

The structure of this paper is as follows: literature review in Section 2; proposed solutions in Section 3; analysis and expression of results in Section 4; and conclusion in Section 5.

2. Literature review

Thus far, various methods have been proposed for the automatic service composition. The proposed algorithms for the Web service composition aim to provide an optimal solution in selection or composition of their services. Some of these algorithms perform Web service composition based on the graph search method. In this method, services are displayed based on their inputs and outputs. For a service registry, a Service Dependency Graph (SDG) is created to display all the possible input-output dependencies between services. Afterwards, the Web service composition becomes a graph search problem that is based upon SDG to find a path from inputs toward outputs or from outputs to inputs to provide a service composition.

In a work carried out by Wang et al. [6], the proposed solution includes a graph search-based algorithm and two pre-processing methods. In the pre-processing methods, before the main algorithm is applied on the service community C for service composition, Web services are categorized in C in a community service, all services that can perform the same task but are applicable in a different situation form a set.

Alrifai et al. [7] have presented an approach based on the Skyline concept that will effectively reduce the candidate services. This approach prunes the services with respect to the quality parameters of the service. The performance of composite applications is determined by the performance of the Web services involved. Identification of the best candidate Web service is done by a similar performance based on the quality parameters of the service. In this way, the selected services will be conducive to optimizing the final composition, while satisfying the user constraints based on quality parameters of the service.

Wu et al. [8] have focused on selecting Skyline services in dynamic environments because new services may be created instantaneously or main services may be rejected or their quality parameters of the service may change. Therefore, using Skyline, a wide range of options are decreased. In order to deal with a dynamic environment of the service, they offered the paper tape that could quickly put the changes of services under its control.

In [9], the Skyline service to reduce the number of candidate services, the DPA algorithm to create an expansion tree, the BUA algorithm to sort linear services in ascending order, and the linear strategy to prune the unnecessary services have been used.

Yu et al. [10] have proposed a method to select services and their composition called “text information composition”. Text information composition consists of non-operational traits of services. They have produced a service selection approach based on text composition by the back-tracking method. In this method, in each one of the processes for selection of candidates, a backward step is taken to see whether this selected service from the previous stage provides the best composition or not. If the service is suitable, it will be called; otherwise, according to the selected service in the next stage, the best service of this stage will be selected.

Boruji et al. [11, 12] have discussed on whether or not a request can be met by a composition of services. They created a tree for each processing model that was stored in the registry to produce a graph that showed the dependencies between atomic services and processes, and moreover, this dependency graph was analyzed to determine whether or not a service composition could meet the request.

Lang & Su [13] have provided an AND/OR graph of a service dependency graph and its search algorithm to detect the Web service composition. The issue of service composition problem becomes AND/OR Graph Search.

The AND/OR Graph is a generalized directed graph, and is generally used in automating and analysis problems. This graph contains two types of nodes: AND nodes and OR nodes that are bonded together by edges. These edges are called connectors. Each connector on the AND/OR graph connects a set of nodes $\{V_i \mid i = 1, 2, ..., n\}$ to a node Vo. If a logical AND exists between $\{Vi\}$s, the connector shows an AND operation, and is referred to as an AND connector. Similarly, if a logical OR exists between $\{Vi\}$s, the connector shows OR operation, and is referred to as an OR connector. Here, the AND/OR graphs are used to display SDG for service composition. In this display, a service can be implemented when all the input parameters are available because a logical AND relationship exists between the nodes that are connected directly to the service node and all nodes of the service in this display are AND nodes. In other words [13]:

If $(I1 \wedge I2 \wedge ... \wedge In)$; then Select the service $Ik \in IA, 1 \leq k \leq n$

In contrast, a logical OR relationship exists between the parameters of the output generated by the service.

According to what was mentioned, the displayed dependence graph based on the AND/OR graph has the following characteristics:

- There is no direct edge between two AND nodes because a service operation cannot take another service operation as an input or produce it as an output.
- A data node can be used as an input parameter for a service operation as well as the output parameter for other service operations.
- A data node may be connected to other data nodes.

Min Lu et al. [14] have solved the Web service composition problem by creating a vector model of the QoS parameters and the QoS evaluation method. They provided a flexible and satisfactory constraint form for selecting services, a function for finding the optimum objective function, and used the branch and bound algorithm.

Wang et al. [15] have solved the Web service composition problem using the distributed cloud computing and constraints of service level. In order to solve situations in which databases exist in different geographical locations, they offered a hybrid approach based on the quality of service by considering the network environment. Then it provided a genetic algorithm based on the Skyline for solving the issued problems.

Mardukhi et al. [16] have changed the composition problem into independent applications to reduce the execution complexity. First, the global constraints are turned into local constraints using a genetic algorithm; finally, the best service for activities based on local constraints using a simple linear search algorithm are obtained.

Li et al. [17] have provided a framework for a reliable Web service composition, and they have considered global constraints. This framework can be embedded as a component of UDDI for service evaluation, dynamic selection, and composition. In this method, a fitness function is used to compute the three values of quality: generic quality parameters, QoS based on scope, and users' quality parameters of the service. They use a global heuristic algorithm.

Gabrel et al. [18] have evaluated the complexity and Web service composition models in their work. Their work has shown that the time-consuming factor in a Web service composition problem is the constraints. They have solved this problem using a linear composition. This technique allows Web service composition in the works that are in the form of AND or XOR.

Puttonen et al. [19] have provided a framework for Web service composition that is done using the semantic description.

Gamha et al. [20] have provided a strategy for the semantic composition of services to manage the user constraints, which is done in two stages. Firstly, an execution model is obtained by the user requests. The executive model is formulated to chart mode. The user's request turns into the Ontology Web Language, and will be formulated to a mapping from the OWL state diagram. In the second stage, the existing service is presented from the developed implementation model. The result includes an action plan that satisfies the user's constraints.

AllamehAmiri et al. [22] have proposed a new composition plan optimizer with constraints based on genetic algorithm. To escape from local optimums, they present some modifications in crossover, mutation and selection approach.

3. Proposed model for Web service composition

The number of service providers with the same performance but with different quality parameters has increased considerably, and consequently, choosing the optimal Web service based on quality parameters has become an important issue [21].

Therefore, efforts to reduce the number of redundant services can result in the better selection of candidate services, and consequently, more efficient composition.

In most methods, only quality parameters of the service are being used in order to select the optimal composition, and the users' constraints are often ignored. Users' constraints are used to ensure the proper implementation of the service or its interaction with other services. As a result, they have a significant impact on the service composition. However, these constraints can be defined in a semantic form, which can increase the users' satisfaction.

As mentioned in the introduction, in this research work, the three parameters cost of service quality, response time, and reliability are used for optimal service composition. Moreover, to improve the scalability of the proposed method, the Skyline service algorithm has been used to reduce the number of services based on the QoS parameters.

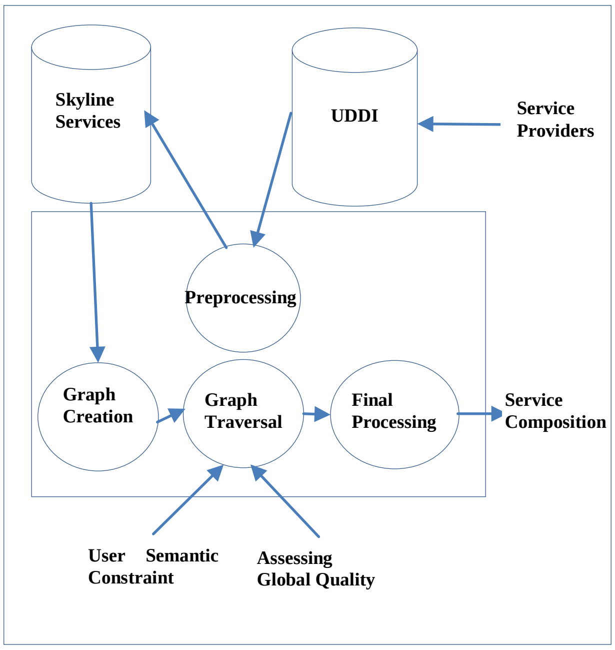

Figure 1. Block diagram of the proposed method.

In this section, a graph search-based algorithm and a pre-processing technique are used for the optimal service composition. According to what was said, the proposed model can be divided into the following steps based on figure 1.

1. Pre-processing
2. Graph creation and traversal
3. User's semantic constraints
4. Assessing the global quality of services
5. Final processing

In the proposed model, the Web service is displayed by tuple a $WS = (op, I, O; SC, QoS)$ that is derived from [6], where:

1. op describes service operation;
2. I is the set of input parameters of the service. Each service has some input values;
3. O is the set of semantic concepts that are referenced by the output parameters of the service;
4. SC represents the constraints of a service, which specify the context conditions that must be met to ensure the correct execution of the service;
5. QoS (Quality of service) is the set of quality parameters of the service such as cost, availability, permittivity, and reliability. In this work, three quality parameters of service including cost, response time and reliability are used.

In the proposed method, for automatic service composition, the main requirements of users including available inputs, expected outputs, quality of service, and priority are initially and explicitly specified by the user and service composition is done with this information. Therefore, the user's request can be represented by a tuple $\mathrm{Re}q = (IA, OE; P, Q)$ derived from [6], where:

1) I_A represents the set of available inputs that a service requester provides;
2) O_E[2] represents the set of outputs that a service requester expects;
3) P represents the set of personalized preferences and constraints defined by a service requester;
4) Q represents the set of standards of service quality parameters defined by a service requester.

3.1. Pre-processing

Web services are created and updated momentarily. The number of services with the same performance but with different quality parameters has increased considerably; hence, choosing the optimal Web service based on quality parameters has become an important issue [21]. Therefore, efforts to reduce the number of redundant services before the composition can be used to reduce the search space and to achieve increased scalability. Therefore, the Skyline technique is used in order to select the desired Web service.

Skyline includes a set of services that are not dominated by other services. A Skyline service selects those points that are not dominated by other points. The point P_i is dominant to P_j if P_i is better than or equal to all the dimensions of P_j, and is better at least in one dimension.

A Skyline query selects the best or most interesting points in all dimensions. In this work, a relationship between the service quality parameters of the service is defined and implemented. This method is used to identify services of a class that is dominated by other services in the same class. These services can be pruned, and thus reduce the number of candidates for composition.

Consider a service class S and two services of x and y as $x, y \in S$. The two services each containing a set of quality parameters of the service Q. X is dominant to service y $(x \prec y)$, if x is better than or equal to y at all dimensions of Q parameters and better at least in one parameter. In other words [7]:

$$\begin{aligned}&\forall k \in [1,|Q|] : q_k(x) \le q_k(y)\ and \\ &\exists k \in [1,|Q|] : q_k(x) \prec q_k(y)\end{aligned} \quad (1)$$

where, Q represents a set of quality parameters of the service and q_k shows the k-th quality of service parameter.

Skyline services of class S that are shown by SL_S consist of a set of services that are not dominated by any services. In other words [7]:

$$SL_s = \{x \in S \mid \neg \exists y \in S : y \prec x\} \quad (2)$$

As stated, the goal is to select a set of services from each class so that it could satisfy global constraints and have the highest fitness as well. It should be noted that in the Skyline service, after a reduction in the number of services, the best services from each class are selected based on the quality parameters of the service, which may not

satisfy the global constraints. Hence, in the next step, user constraints are applied to combine and select the best service. Therefore, different compositions of each class should be considered. In this work, a Skyline service is executed for each class so that suitable or candidate services to be part of the composition or those that cannot be in the final solution can be identified. This method can effectively reduce the problem space.
The algorithm uses three parameters: cost, response time, and reliability to apply Skyline service algorithm. In other words, according to what was stated, services are provided as a point in 3D space and the coordinates of each point are related to levels of quality of service.

3.2. Graph creation and traversal
After removal of the redundant services, the optimal services are displayed as inputs and outputs. First, a dependency graph is created in the registry to display all the input-output data communication between services. Moreover, the Web service composition becomes a dependency graph search. In the proposed model, similar to [6, 13], the AND/OR graph is used to display the dependency graph.
Through this display for the service dependency graph, first, a SDG is created according to community C; and secondly, the search process in the dependency graph based on the information provided by the service applicant (e.g. user Req) begins, which is the depth-first search.
In the node search process, if some conditions are not met or a path is not found, it returns and completes the move in the entire graph. When the first service is selected, in case the kind of operation is the same as the operation of the first service selected by the user, it begins to examine and evaluate the inputs. The input of a service can sometimes be obtained using the output of the previously issued service.
If no proper service is found in services searching in an operation, recursive algorithm steps back to the previous operation and finds another appropriate service. Finally, the algorithm finds and displays all the possible compositions.
The characteristics of an AND/OR graph makes it possible that a service is called only if all of its inputs are available.

3.3. User's semantic constraints
In the ongoing implementation of the proposed model, the user's semantic constraints are assessed in any operation after matching the input and output parameters. Constraints are applied to certain attributes of the Web service to choose the right service and ensure its proper implementation. The range of these attributes can be displayed by numerical values or a set of semantic concepts.
The semantic relation between the words used to describe the service constraints leads to accuracy in exploring the Web service. In the proposed model, the semantic constraints of the user are evaluated using the conditional statements. In the composition process, if the constraint is raised by the user on the operation, it will be considered in the selection of service. All the constraints of a service can be displayed as follow:

$$SC_1 \wedge SC_2 \wedge \ldots \wedge SC_n \rightarrow WS, \quad SC_i \in SC, 1 \le i \le n \tag{3}$$

3.4. Assessing global quality of services
It should be noted that the best service from each operation does not necessarily guarantee that the global constraints are satisfied. As stated earlier, the goal was to select a set of services for each operation, and the global constraints are considered as well.
In the proposed model, after finding a composition strategy of graph traversal, QoSs of the service composition are calculated and evaluated based on the relationships in table 1. The criteria for assessing the cost quality parameters of the service are set by the users, and the criteria for assessing the other quality parameters of the service are specified in the program.
If this solution satisfies the global qualitative parameters, it will be considered as a solution.

Table 1. Estimation function of the consensus values of QoS [7].

QoS attribute	Function
Response time	$\sum_{i=1}^{m} T(t_i)$
Reliability	$\prod_{i=1}^{m} R(t_i)$
Cost	$\sum_{i=1}^{m} C(t_i)$

3.5. Final processing
After completing the composition process, if no composition strategy is produced in the previous step, the proposed algorithm re-runs the search process by changing the priorities in the user's constraints. Therefore, in order to increase the user's satisfaction, the algorithm can produce the proposals with lower priorities if no solution is produced.
According to the above-mentioned explanations, the proposed algorithm reduces the existing Web services through the Skyline technique according

to the quality parameters of the service cost, response time, and reliability. Afterwards, the algorithm presents all the possible composition solutions using the graph search-based algorithm based on the user's semantic priorities. However, in order to increase the user's satisfaction, the algorithm can produce the proposals with lower priorities if no solution is produced.

4. Analysis of results and discussion

As stated earlier, the automatic Web service composition based on graph search algorithms is carried out by the researchers. In the proposed method, in addition to consideration of the qualitative parameters and the user's constraints that can be deemed semantic, the Web service composition was studied in many service communities. This section studies and evaluates the simulation results of the proposed algorithm.

4.1. Simulation tools and system specifications

For simulation and implementation of the algorithm, Visual Studio 2012 was used.
The system configuration used in the implementation and simulation of the algorithm is as in table 2.

Table 2. System configuration used in the simulation.

Processor	Intel (R) Core (TM) i5-2430 M CPU @ 2.40 GHZ
Main memory	4GB
Operating system	Microsoft Windows 10 Home, 64-bit

4.2. Selected dataset

The service dataset is created by the program as a prototype for a traveling tour. Each Web service, as defined in Section 3.2, represents an operation that has input and output parameters and can own some constraints. Furthermore, random values are assigned to the QoS of reliability, response time, and cost.

4.3. Simulation results of proposed model

In order to analyze and evaluate the proposed method, firstly, the approach by Wang et al. [6] was studied, and secondly, the desired results on the same data provided by the Wang algorithm and the proposed algorithm were analyzed. The comparison is performed in three aspects: runtime, memory usage, and fitness. The results of the proposed algorithm in the optimal composition of Web services are as what follow.

4.3.1. Runtime

In analyzing the results obtained from the implementation of the proposed algorithm, it was observed that by increasing the number of Web services of the dataset, the runtime increased. In figure 2, the runtime of a different number of Web services per composition operations is shown.
It is clear that by increasing the number of Web services, the runtime increases. However, as it can be seen in figure 2, increasing the number of operations has little impact on the different Web service execution times.

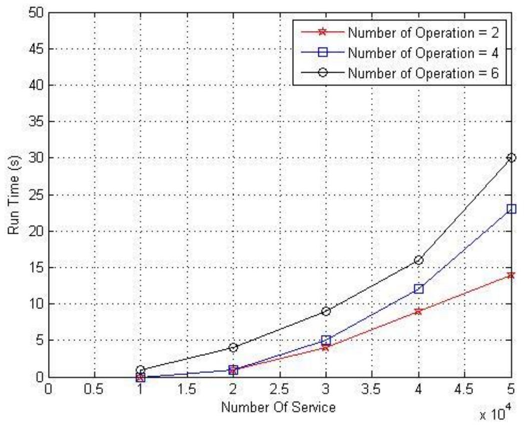

Figure 2. Runtime of executing a number of different Web services.

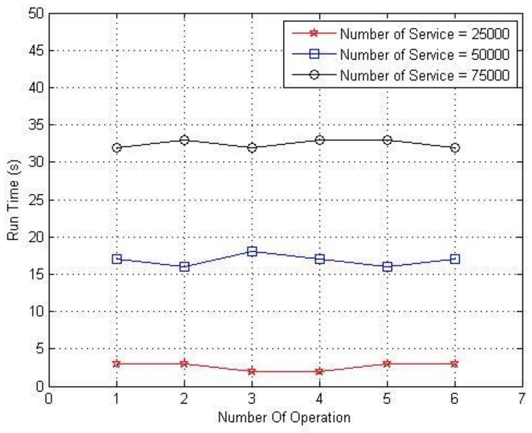

Figure 3. Runtime of proposed algorithm for the number of steps combined.

Figure 3 shows the runtime of the proposed algorithm for Web service composition per the number of composition operations of 25000, 50000, and 75000 services. In the proposed method, due to the elimination of redundant services and reducing the searching space, the time spent for Web service composition per number of composition operation had no dramatic increases.
By assessing figure 3, it can be concluded that increasing the number of Web services has a significant impact on the running time.

Figure 4 compares the runtime of the proposed algorithm with the algorithm provided by Wang and colleagues [6]. According to what can be seen in the graph, the runtime difference is much more intuitive in a higher number of Web services.

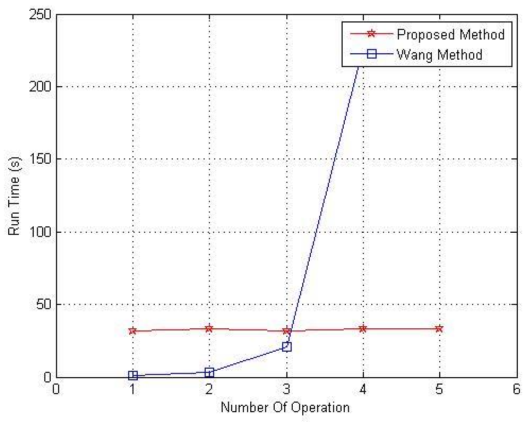

Figure 4. Comparison of runtimes in a different number of Web services.

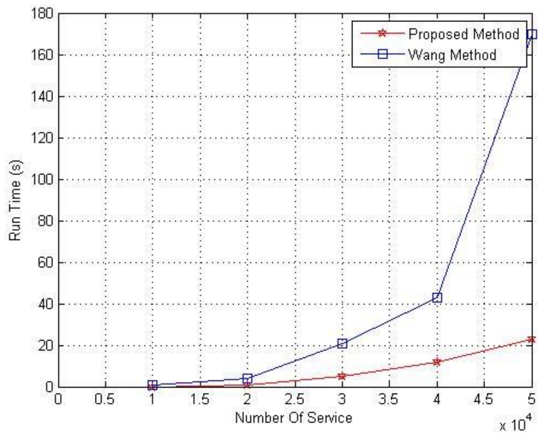

Figure 5. Comparison of runtimes per composition operations.

Comparison of runtime per composition operation between the proposed algorithm and the method developed by Wang is shown in figure 5. The number of Web services is equal to 75,000. Increasing the number of operations increases the depth of Search Graph. From the following graph, one can conclude that in the method developed by Wang for these numbers of services, the Graph Search space becomes very large and leads to an exponential increase in the runtime. However, in the proposed method, due to a decrease in the number of Web services before composition, increasing the depth of Search Graph will not have much impact on the runtime.

4.3.2. Memory usage

Another case that can be analyzed in the results obtained is the amount of memory consumed by the proposed algorithm.

In figure 6, the memory usage per different number of Web services is compared in the proposed method and the method provided by Wang et al. [6]. According to what can be seen in the graph, the memory usage difference is more tangible in a higher number of Web services. Since the pre-processing in the proposed method reduces the number of Web services, the composition process will be done with a smaller number of Web services; this method consumes less memory compared to the Wang's method. In this comparison, the number of considered operations is 4.

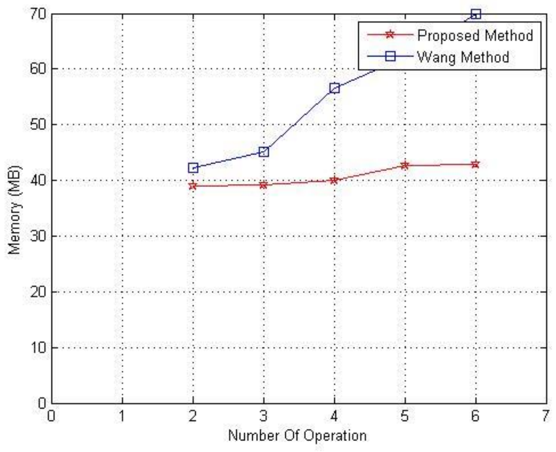

Figure 6. Comparison of memory consumption by different numbers of web services.

Memory usage per number of composition operation is compared in the proposed method and the method provided by Wang et al. [6] in figure 7.

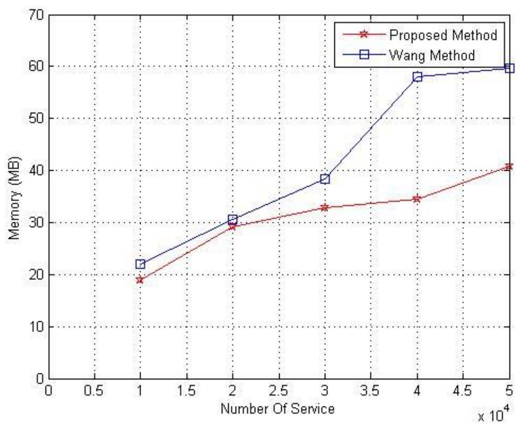

Figure 7. Comparison of memory usage per composition operation.

The number of Web services is equal to 75000. As it can be seen, an increase in the number of operations in the proposed method does not have a significant effect on the memory usage. Since the Skyline service has reduced the number of services to be selected from them, the search graph has become smaller.

4.3.3. Fitness

The effect of number of Web services and operations on fitness: the results obtained are shown in table 3.

Table 3. Fitness of Wang Method and Proposed Method.

	Number of services			Number of operations		
	25000	50000	75000	3	4	5
Wang method fitness	0.57	0.61	0.47	0.56	0.41	0.41
Proposed method fitness	0.69	0.65	0.58	0.63	0.59	0.58

4.3.4. Effect of pre-processing on number of services

Pre-processing in the method provided by Wang is done as follows: first, all the services in a community that can do the same thing but are applicable in a different situation form a service set called SIDE. In this method, before the main algorithm is applied in community C for service composition, SIDE is categorized in C. Thus pre-processing will not reduce the number of services; in fact, it provides a condition to meet the service constraints.

However, in the proposed method, services are decreased using the Skyline concept. In the diagram of figure 8, the effect of processing on the number of services is shown.

The assessments carried out indicate the following results:

• The effect of number of Web services on runtime: the assessment results indicate that the Web service composition using the proposed algorithm is very large, leading to a faster solution.

• The effect of number of composition operations on the runtime: as evaluations were presented, the composition with the proposed method achieves a better solution in a much less time, and the slope of comparison graph is significant.

• The effect of number of Web services on the memory usage: the assessment results show that the proposed algorithm uses less memory than the Wang's method as the number of Web services increases.

• The effect of numbers of composition operations on the memory usage: the assessment results show that an increase in the composition operations has no significant impact on the memory usage in the proposed algorithm.

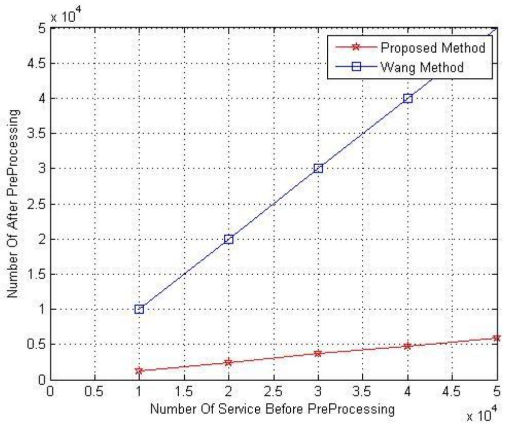

Figure 8. Pre-processing effect on the number of services.

5. Conclusions and suggestions

With the expansion of services for enterprises and organizations on the Internet, the demand for communication and interaction has increased, and new technologies have been presented in the context of this type of communication. Currently, Web services are the best options to provide Internet services. Due to the growth and an increasing number of Web users and the complexity of the users' queries, simple and atomic services are not able to meet the needs of the users; and to provide complex services, they require service composition.

Several methods have been proposed to solve the Web service composition. Some of them create only one service composition and some produce a solution that cannot truly be considered as a service composition because each stage of the solution includes a set of redundant services and all services are combined in such solutions. Finding all composition approaches for the Web service composition is difficult and complex. Large search space, redundant services, etc. limit the performance of all composition solutions, and therefore, the previous methods have focused on finding an optimal service composition. However, in this work, the proposed algorithm could generate all the possible services for the WSC problem.

Despite multiple providers, different services are provided that are similar in terms of performance and can replace each other but these services are different in terms of qualitative criteria such as the response time, availability, security, reliability, and running costs. For example, if a composite service is composed of n atomic services and M candidate services exist for each atomic service, the usual way to solve this problem will be m^n, which is very costly and time-consuming. In this work, search space for selecting Web services was reduced using the Skyline service.

Afterwards, a graph search-based for Web service composition was proposed. The constraint set by the user, which may be defined as semantic, was considered. The proposed method produces all the possible solutions, and includes preferences of the users based on their requests.

In the previous section, the results obtained show that the proposed algorithm is acceptable because:

- Using the Skyline algorithm reduces the number of existing services, and thus leads to the scalability of the proposed approach.
- Applying the user's constraints in addition to QoS like the response time, cost, and reliability resulted in the optimal selection and optimal service composition.
- Service composition is done considering the main requirements of the users including the available inputs, expected outputs, quality of service, and priority.
- Applying meaning to the constraints requested by the user leads to more accurate results.

Despite generating all the possible solutions, the Web service composition problem is still complex and difficult, which is due to the nature of the problem mentioned earlier.

In this work, the Web services were considered to be compatible. It is notable that sometimes it is required to call some other services for one service. For example, to request the map service provider, it requires two operations of geocoding and map production, and it is possible that the dependency constraints exist between these two operations (such as map production, which depends on the geographic code). Consideration of this issue can increase the capabilities of the proposed model.

References

[1] Papazoglou, M. P., Traverso P., Dustdar S. & Leymann F., (2007). Service-oriented computing: State of the art and research challenges, Computer, vol. 40, no. 11, pp. 38–45.

[2] Papazoglou, M. P., Traverso P., Dustdar S. & Leymann F.,(2008). Service-oriented computing: A research roadmap, International Journal of Cooperative Information Systems, vol. 17, no. 2, pp. 223–255.

[3] Zhang, L. J., Cai, H., & Zhang, J. (2007). Services computing. Beijing: Tsinghua University Press.

[4] Ter Beek, M., Bucchiarone, A., & Gnesi, S. (2007). Web service composition approaches: From industrial standards to formal methods. Second International Conference on Internet and Web Applications and Services, pp. 15-20, Morne, Mauritius, 2007.

[5] Rao J. & Su X., (2004). A survey of automated web service composition methods, in Proceeding Semantic Web Services and Web Process Composition, pp. 43–54, Springer, Berlin, Heidelberg, 2004.

[6] Wang P. W., Ding Z. J., Jiang C. J., & Zhou M. C., (2014). Constraint-Aware Approach to Web Service Composition, IEEE, Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 6, pp. 770-784.

[7] Alrifai M., Skoutas D. & Risse T., (2010). Selecting Skyline Services for quality of service-based Web Service Composition, ACM, Proceedings of the 19th international conference on World Wide Web, No. 10, pp. 11-20, North Carolina, USA, 2010.

[8] Wu, J., Chen L. & Liang T. (2014). Selecting Dynamic Skyline Services for quality of service-based Service Composition, Applied Mathematics & Information Science, vol. 8, no. 5, pp. 2579-2588.

[9] Yu, Q. & Bouguettaya A. (2013). Efficient Service Skyline Computation for Composite Service Selection, IEEE Transaction On Knowledge And Data Engineering, vol. 25, no. 4., pp.776-789.

[10] Yu, H. Q. & Reiff S., (2009). A Backwards Composition Context-Based Service Selection Approach for Service Composition, IEEE International Conference on Services Computing, pp. 419 – 426, Bangalore, India, 2009.

[11] Brogi, A., Corfini, S., & Popescu, R. (2005). Composition-oriented service discovery. In International Conference on Software Composition, pp. 15-30, Springer, Berlin, Heidelberg, 2005.

[12] Brogi, A., Corfini, S., & Popescu, R. (2008). Semantics-based composition-oriented discovery of web services. ACM Transactions on Internet Technology (TOIT), vol. 8, no. 4, pp. 1-39.

[13] Liang, Q. A., & Su, S. Y. (2005). AND/OR graph and search algorithm for discovering composite web services. International Journal of Web Services Research, vol. 2, no. 4, pp. 48-67.

[14] Liu, M., Wang M., Shen W., Luo, N. & Yan J., (2012). A quality of service (quality of service)-aware execution plan selection approach for a service composition process, Science Direct, Future Generation Computer Systems, vol. 28, no. 7, pp. 1080–1089.

[15] Wang, D., Yang Y. & Mi Z., (2015). A genetic-based approach to web service composition in geo-distributed cloud environment, Science Direct, Computers and Electrical Engineering, vol. 43, pp. 129–141.

[16] Mardukhi, F., NematBakhsh, N., Zamanifar K. & Barati, A., (2013). quality of service decomposition for service composition using genetic algorithm , Applied Soft Computing, vol. 13, no. 7, pp. 3409–3421.

[17] Li, J., Zheng X. L., Chen S. T., Song. W. W. & Chen D. (2014). An efficient and reliable approach for quality-of-service-aware service composition, Information Sciences, vol. 269, no. 16, pp. 238–254.

[18] Gabrel, V., Manouvrier M. & Murat C (2015). Web services composition: Complexity and models, Discrete Applied Mathematics, vol. 196, no. 16, pp. 100-114.

[19] Puttonen, J., Lobov, A., Cavia Soto, M. A. & Lastra J. L. (2015). Planning-based semantic web service composition in factory automation, Science Direct, Advanced Engineering Informatics, vol. 29, no.4, pp. 1041–1054.

[20] Gamha, Y. & Bennacer, N. (2008). A Framework for the Semantic Composition of Web Services Handling User Constraints, ICWS '08 International Conference on Web Services, IEEE, pp. 228-237, Beijing, China, 2008.

[21] Himi, F., Ben, S. Y. & Ben, S. A., (2015). Efficient Skyline Computation For optimal Service Composition with Fuzzy preference relationships, International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6, Yasmine Hammamet–Tunisia, 2015.

[22] AllamehAmiri, M., Derhami, V., & Ghasemzadeh, M. (2013). QoS-Based web service composition based on genetic algorithm. Journal of AI and Data Mining, vol. 1, no. 2, pp. 63-73.

نشریه هوش مصنوعی و داده کاوی

ترکیب سرویس های وب در مقیاس بزرگ با آگاهی از پارامترهای کیفیت سرویس و محدودیت‌های معنایی کاربر

الهام شهسواری و سیما عمادی*

گروه مهندسی کامپیوتر ، دانشگاه آزاد اسلامی، یزد، ایران.

ارسال ۲۰۱۷/۰۳/۰۷؛ بازنگری ۲۰۱۸/۰۳/۰۶؛ پذیرش ۲۰۱۸/۰۶/۰۲

چکیده:

معماری سرویس‌گرا و تکنولوژی‌های مرتبط، با استفاده از یکپارچه سازی تجاری بر روی شبکه‌هایی مانند اینترنت زمان اجرای تعاملات را تسهیل می-بخشند. در حال حاضر سرویس‌های وب به عنوان بهترین گزینه جهت ارائه خدمات اینترنتی می‌باشند. با توجه به رشد روز افزون کاربران وب و پیچیده شدن درخواست کاربران، سرویس‌های ساده و اتمیک قادر به پاسخگویی به نیاز کاربران نبوده و جهت ارائه سرویس‌های پیچیده نیاز به ترکیب سرویس‌ها می‌باشد. وب سرویس‌ها امروزه به صورت لحظه‌ای ایجاد و بروز رسانی می‌شوند لذا در دنیای واقعی تعداد زیادی سرویس وجود دارد که ممکن است قابلیت ترکیب نسبت به شرایط انتخابی و محدودیت‌های مورد نظر کاربر که می‌تواند به صورت معنایی نیز تعریف شود برای آنها وجود نداشته باشد. در روش پیشنهادی برای ترکیب خودکار سرویس‌ها، نیازمندی‌های اصلی کاربران شامل ورودی، خروجی‌های مورد انتظار، کیفیت سرویس‌ها و اولویت‌ها می‌باشد که تمامی آن‌ها از ابتدا توسط کاربر به صراحت مشخص و ترکیب سرویس‌ها با این اطلاعات انجام می‌شود. در راهکار ارائه شده، به دلیل تعداد زیاد سرویسهای با وظیفه مندی یکسان، ابتدا سرویس‌های کاندید برای ترکیب با استفاده از روش skyline برمبنای کیفیت سرویس‌ها کاهش یافته و سپس با استفاده از یک الگوریتم مبتنی بر جستجوی گراف تمامی راهکارهای ممکن تولید می‌شود. در نهایت محدودیت‌های معنایی کاربر در ترکیب سرویس‌ها اعمال می‌گردد و بهترین ترکیب با توجه به درخواستهای کاربر ارائه می‌شود. نتایج حاصل از تحقیق نشان می‌دهد، روش پیشنهادی مقیاس پذیر و کارآمد است و با توجه به در نظر گرفتن محدودیت‌های معنایی کاربر، راه حل بهتری را ارائه می‌دهد.

کلمات کلیدی: ترکیب سرویس، وب سرویس، سرویس skyline، پارامترهای کیفی، سرویس، جستجوی گراف.