Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 9, No. 2, May 2018), Pages: 91-102
www.jacr.iausari.ac.ir

Automatic Service Composition Based on Graph Coloring

Sepideh Sheivandi, Sima Emadi✉
Department of Computer Engineering, Yazd Branch, Islamic Azad University, Yazd, Iran
ms.sheivandi@iauyazd.ac.ir; emadi@iauyazd.ac.ir

Received: 2017/02/16; Accepted: 2017/09/15

Abstract

Web services as independent software components are published on the Internet by service providers and services are then called by users' request. However, in many cases, no service alone can be found in the service repository that could satisfy the applicant satisfaction. Service composition provides new components by using an interactive model to accelerate the programs. Prior to service composition, the most important issue in finding suitable candidate services samples is their compliance with non-functional requirements. Thus, designing an efficient way to combine a chain of connected services is important. Recently, numerous studies have been done to reduce the search time in finding a service composition. However, many of these methods to examine and investigate all Web services in a Web repository require a long time, which occupy the user's time significantly. This paper provides an approach for automatic quality-aware service composition as well as the users' preferences in achieving the optimum composition results. For this purpose, modified graph coloring method to filter the data before compositions in large-scale data is used which decreases selected services set. The application of KPL algorithm in this study provided some proper solutions to the user so that these solutions can be used instead of the best composition if necessary. Therefore, the results derived from the analysis of the proposed method, indicates a good optimization in runtime and memory consumption. The evaluation results show that the proposed method in memory consumption and runtime has improved by about 20%.

Keywords: *Coloring-Based; Service Composition; Top-K Algorithm; Quality-Aware Service; KPL Algorithm*

1. Introduction

Since finding composition results manually from high volume of services is impossible, automatic service composition is presented, which aims to enable automatic search of service composition for given requests. There are mainly two methods for composition: artificial intelligence planning and Graph search. Graph search methods first build a dependency graph based on user requests, and then use the search algorithm to find a solution. Despite similarities, these methods are different on searching methods, efficiency and quality of the solution.

Services composition refers to creation of a new performance for multiple services composition offered by different manufacturers. For distinguishing between Web services that have similar functionality, the quality of service is used. Due to the high dynamics and rapid growth in the number of Web services that are functionally similar, finding optimized web service composition at the right time to meet the needs of users has become a challenging task and a concern [1].

According to studies, it can be concluded that available studies on finding the optimal result of service composition has several shortcomings. Service providers need to improve the entry that causes the failure and its consequences; which for example can be lack of composition dynamism, lack of accuracy and speed, and the lack of choice for users. This could bring many limitations for users and service providers. To deal with this problem, Top-K algorithm was used in the method by Jiang. In addition, to evaluate the right choice for users to get the right composition, KPL algorithm is used [2].

Since the Top-K algorithm used in the method by Jiang has some shortcomings such as being costly and high runtime, in this study, two methods that are based on modified algorithm of graph coloring used to filter the number of services have been implemented with MGC-Top K and MGC-K. The filtering operation procedure is done in terms of quality of service. By doing so the number of services known as nodes of the graph will be declined and makes the selection process and service composition faster and memory usage will be less.

This paper is as follows: In the second section, definition of the KPL algorithm and Top-K algorithm is presented. In section 3, the proposed (MGC-K and MGC Top-K) method is described. Section 4 presents the evaluation. Section 5 presents related works. Finally, in Chapter 6, conclusion is presented.

2. Top-K Algorithm and KPL Algorithm

2.1 Top-K algorithm and extraction of dependency graphs

Since selecting an appropriate service from a large-scale set of service, increases processing time; therefore, it is better to reduce the scale of the candidates. For this purpose, it is necessary to filter useless services in the early stages.

There are several algorithms to reduce the number of services in the combination process, including C-Means Fuzzy Clustering [3], Hierarchical Tree classification [4], clustering based on Markov Model [5] and Top-K classification Algorithm [6].

In this paper, the Top-K algorithm has been used to resolve the problem of incompatibility of services to reduce the number of services.

When a request is received from the user, the services are fetched from rules repository and impossible services in final results are filtered according to user requests and rules contained in the respiratory [7].

One of the ways to filter useless services is using Top-K algorithm in response to the user's query, which produces compositions in parallel. The filtering process is as follows [7]:

Input set along with the primary elements including the entries of user requests will be created.

All elements that their input include input elements are found. Then, output of these services is added to the input set and services are kept.

Step 2 is repeated till another service whose entries are elements in the Input set does not exist. In other words, this set can be expanded.

All elements that their output include output elements are found. Then, output of these services is added to the output set and services are kept.

Services inputs which were stored in previous step are compared with the outputs of services that are still in service respiratory. And Step 4 is repeated until another service to process is not found.

After running this algorithm, services that cannot be implemented by user requests will be filtered out. This issue greatly reduces the number of candidate services.

2.2 KPL

According to previous works [1], [8], [9], we know that finding a good service composition, is not (almost) so easy and there is no need to mention Top-K results. Therefore, we have to find a way to simplify the problem. A very important feature from service composition: the key path. Key path is a chain of services that are in a sequence and has same overall quality of service in the results of the composition. In short, KPL algorithm involves the following steps [2]:

1. First, using the Sim-Dijkstra algorithm, forward checking is done from Start node in the dependency graph.
2. Based on the recorded information in the Sim-Dijkstra algorithm, optimal key path is retrieved by performing backtracking.
3. By using the optimal key paths, a directed acyclic graph is produced. If directed acyclic graphs is larger than K, the answer is returned to the user and it ends. Otherwise, the key path stays in the priority queue, and the next step is done in form of recurrence.
4. Retrieve the key path in the priority queue. Loose vertices using the KPL algorithm. So that worse key path are provided and are placed in the priority queue.

It should be noted that KPL algorithm is implemented if the number of produced directed acyclic graphs through Sim-Dijkstra algorithm be less than user's request.

3. Proposed Method

Since the Top-K algorithm used in the method by Jiang has some shortcomings such as being costly and high runtime, in this study, two methods that are based on modified algorithm of graph coloring used to filter the number of services have been implemented with MGC-Top K and MGC-K. The filtering operation procedure is done in terms of quality of service. By doing so the number of services known as nodes of the graph will be declined and makes the selection process and service composition faster and memory usage will be less.

The innovation of this article has been done in two ways which is based on a study carried out by Jiang.

In the first method, by implementation of the MGC-Top-K on the output of Top-K algorithm, we have filtered the services; whereas in the second method, Top-K algorithm has nothing to do and MGC-K is applied directly on the server for filtering operations.

3.1 The first method

1. In this step, like Jiang method, first, Top-k algorithm is executed on initial services that make up a large graph, and dependency graph will be extracted.
2. In this step, unlike Jiang, output of Top-K will be sent to MGC-Top-K instead of being sent to Sim-Dijkstra algorithm and filtering is done using forward checking.
3. In this step, like Jiang method, forward checking by Sim-Dijkstra Algorithm from Start node in dependency graph obtained from the second step will be performed.

4. In this step, like Jiang method, on the basis of the information recorded in Sim-Dijkstra Algorithm, optimal Key path is retrieved by performing backtracking.

5. In this step, like Jiang method, using the optimal key paths, a directed acyclic graph is produced. If directed acyclic graphs is larger than K, the answer is returned to the user and it ends. Otherwise, the key path stays in the priority queue, and the next step is done in form of recurrence.

6. Finally, like Jiang method, the key path in the priority queue is retrieve. Loose vertices using the KPL algorithm. So that worse key path are provided and are placed in the priority queue.

3.2 The second method

It is similar to the steps performed in the first method. Top-K algorithm is removed and MGC-K is used instead of MGC-Top-K. The procedure is briefly described below:

1. In this step, unlike Jiang, first MGC-K runs on the set of services that make up a large graph, and dependency sub-graph will be extracted.

2. In this step, like Jiang method, forward checking by Sim-Dijkstra Algorithm from Start node in dependency graph obtained from the second step will be performed.

3. In this step, like Jiang method, on the basis of the information recorded in Sim-Dijkstra Algorithm, optimal Key path is retrieved by performing backtracking.

4. In this step, like Jiang method, using the optimal key paths, a directed acyclic graph is produced. If directed acyclic graphs is larger than K, the answer is returned to the user and it ends. Otherwise, the key path stays in the priority queue, and the next step is done in form of recurrence.

5. Finally, like Jiang method, the key path in the priority queue is retrieve. Loose vertices using the KPL algorithm. So that worse key path are provided and are placed in the priority queue.

3.3 Modified Graph Coloring (MGC) Algorithm

At first, existing graph will be colored by graph coloring, such that no two adjacent vertices share the same color, and a maximum of m different colors be used. Each service indicates a vertex and the link between services that is based on input and output of each service shows an edge.

Graph coloring process is as follows:

1. Initially a vertex is considered as the starting node, and a color is assigned to it.

2. In the next step, the remaining vertices are met, and each vertex that is not colored, will be colored according to the rule of different colored of adjacent vertices.

3. This cycle continues until all vertices are surveyed and colored.

After the graph coloring, filtering is as follows:

1. First, from start node all children that share the same color will be selected, and each will be placed in an individual category.

2. Then, according to the number of vertices in the selected category is available for each color.

3. At this stage categories are sorted in ascending order based on the parameters' values of quality of service.

4. If a vertex has more than 3 children. The next step is executed for filtering vertices, otherwise, all children of that vertex will be surveyed.

5. Then filtering operations based on an interval that is one of the QoS parameters is defined for the selected category according to equation (1).

$$\begin{cases} if\ N\ is\ odd\ Then\ Max = \mathrm{W}_{(\mathrm{N}+1)/2} \\ if\ N\ is\ even\ Then\ Max = \left(\left(\mathrm{W}_{\mathrm{n}/2}\right)+\left(\mathrm{W}_{\frac{\mathrm{n}}{2}+1}\right)\right)/2 \end{cases} \gg 1 \le \mathrm{N} \le \mathrm{i} \tag{1}$$

N: number of same color nodes of a parent, and here it is the index of parametric values for the quality of service. W_i: the quality of service of each vertex. With these explanations, if QoS parameters from vertices of the same color are in form of Table 1, the range is calculated as follows by using equation (1).

Table 1: QoS parameters for 5 same color vertex

N	1	2	3	4	5
Wi	5	15	20	23	38

Since there are 5 vertices, and it is odd, with regard to the equation 1, W(1+5)/2 is equal to 20. The numbers range from is selected from 5 to 20 for vertices filtering.

6. Then, this range is applied on all vertices in the current category and the remaining vertices will apply. And each vertex service that its quality parameter value is obtained in the range will be selected, and the remaining vertices will be deleted.

7. Finally, the first step of algorithm is repeated, and continues until they reach the final point of the graph.

4. Evaluation

As previously stated, the composition and web service filtering is done based on QoS parameters by algorithms such as Top-K. In the proposed methods of MGC-K and MGC-Top-K, the composition has been done by Web services management using modified graph coloring and integrating it with the presented algorithms in [2].

In this section, we review and evaluate simulation results of MGC-K and MGC-Top-K. Before examining the results and the experiments, the method of implementation is described. Finally experiences and the evaluation of method will be discussed and the obtained results are analyzed.

4.1 The Simulation Results

For the analysis and evaluation of MGC-K and MGC-TopK, first the implementation of the approach proposed by Jiang is discussed. And then the desired results with the same data on algorithms by Jiang and proposed MGC-K and MGC-TopK methods were analyzed.

The results of proposed MGC-K and MGC-TopK methods for the problem of finding the optimal composition of several Web services are as follows:

4.1.1 Execution Time

In analyzing the obtained results, it is observed that by increasing Web services in data set, runtime increases.

Due to the fact that in the proposed methods of MGC-K and MGC-TopK, Web service composition is done by using the filtering algorithm Top-K and modified graph coloring. Therefore, filtering runtime for the number of same Web services, using the algorithm Top-K compared to MGC-TopK methods and MGC-K has decreased. The reason for increase in runtime is that first graph coloring is done, and then the filtering operation will be performed. However, in Top-K, only filtering algorithm is performed.

The MGC-K also benefited from more reduced runtime compared to MGC-TopK; and improvement occurred in running time. As seen in Figure 1, runtime difference is tangible in the data of different servicesin comparison with the Jiangmethod. In addition to using Top-K algorithm for filtering services, MGC-K and MGC-TopK methods in accordance with procedures that were previously stated, are used for filtering operation.

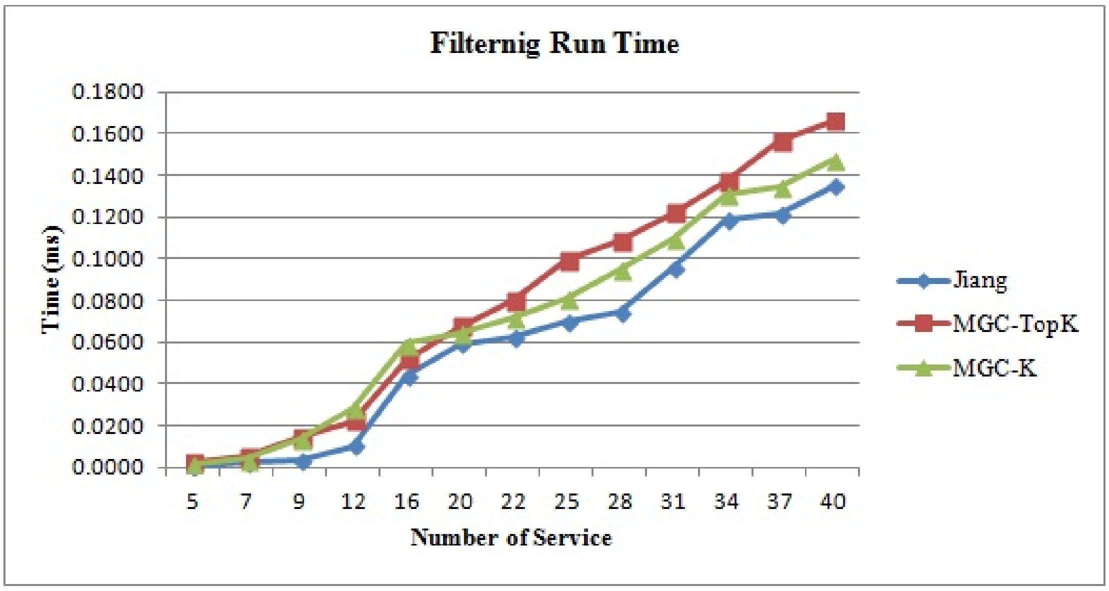

Figure 1: Comparison of filtering runtime in Jiang and MGC-TopK and MGC-K methods

4.1.2 Memory Usage

Another study case which can be analyzed in the results is memory usage by MGC-K and MGC-TopK methods.

Since in the proposed methods filtering operations based on graph coloring, Candidate set has decreased for composition (Figure 2). Therefore, memory usage compared to Jiang method is less and improvements occurred in memory usage.Calculation of memory consumption is such that each service occupies a space according to the input and output parameters, local variables, and so on. This amount of memory in the proposed method is reduced due to filtering operations and reducing the number of services.

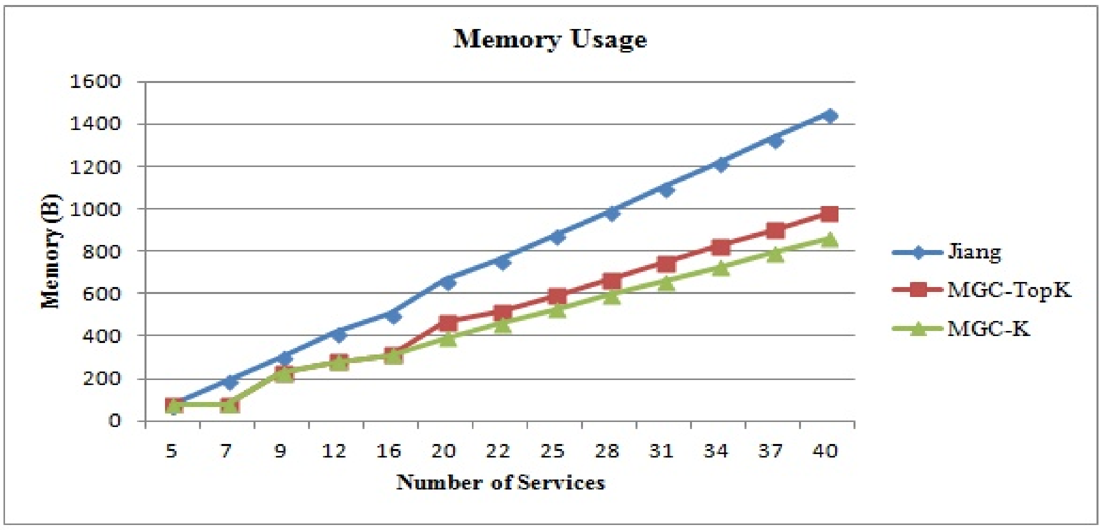

Figure 2. Comparison of memory usage in Jiang and MGC-TopK and MGC-K methods

4.1.3 Total Runtime

Another study case which can be analyzed in the results is the total running time by MGC-K and MGC-TopK methods.

Figure 3 shows the total runtime in Jiang method and proposed methods with different time intervals and same number of services.

The results showed that Jiang method compared to the MGC-K and MGC-TopK has much higher runtime. These proposed methods have been able to reduce runtime.

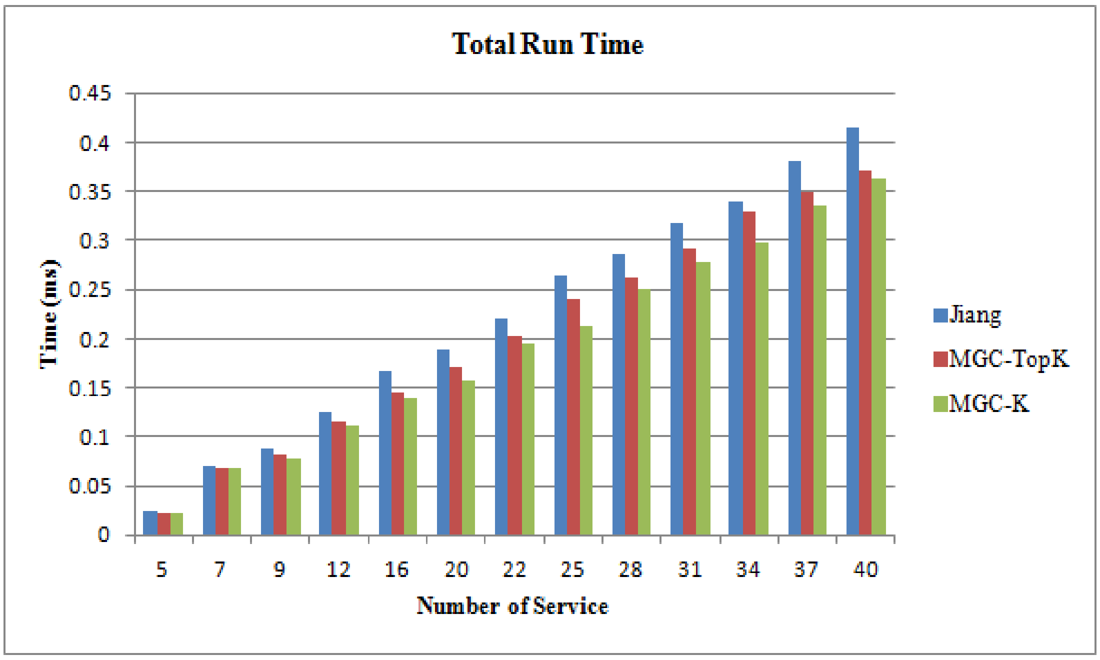

Figure 3: Total Runtime in Jiang method and proposed methods

5. Related Works

Automatic service composition based on graph search or programming artificial intelligence is one of the important issues in the service field. Considering the extent of the issue, several methods have been proposed for automatic service composition, that some of them have been studied in this paper:

Brahmi et al. studied the automatic service composition. The QoS-aware proposed method is based on peer-acting agents and is based on two main aspects: i) Self-organization of agents into dependency graph named social network agent, and 2) distributed computing of the optimal web services composition by a cooperative protocol among agents. The first step is done at design-time in order to manage the services dynamicity and the second step satisfies many user queries simultaneously. This method can create a precise composition in dynamic environment and will increase by number of web services. In this method, the response time is reduced by the distribution joint and creation of agreement through technology. The method is not efficient when increasing the number of services and state space explosion. Also, when the web service is invalid, other compounds can be selected as an alternative [10].

Deng et al. presented a new method for automatic composition of QoS-aware web services through Top-K. The method comprises three successive stages [11]:

Firstly, a forward checking to create a design graph that greatly reduces the search space of the next two steps.

Secondly, the estimation of optimum local QoS will be done so that all values of local QoS in the design process is estimated.

In the third step, a backtracking to find compound web services of Top-K with optimum service quality levels according to the quality of service design graph is done.

Deng et al. proposed an approach for Top-K automatic service composition. The issue of scalability and presenting multiple methods to the user instead of providing single method made the team to explain a parallel method for providing different solutions of large-scale services set. This research explains scalability by Top-K solutions. In this method, a service composition becomes a search graph and the composition algorithm is based on the combination of backtrack search and depth-first search, which can be executed in a parallel way and return the best K compositions [7].

Zou et al provided an approach for dynamic Web service composition using an efficient design in large-scale service reservoirs. How to design an efficient way to combine a chain of connected services to find a service or services that could satisfy the applicant satisfaction is important. Survey and evaluation of all web services in a web repository need a long time that takes up the user's time considerably. Proposed approach tries to solve designing problems by using a combination of automated web services based on artificial intelligence to reduce the search time in finding a service composition. Unlike many traditional methods that service repository several times, this approach probes web service repository only once and converts it to a PDDL, that is updated when changes in the repository occurs. This behavior causes a significant decrease in response time and improves scalability of web services composition. [12].

Silva et al. provided a technique for graph-based particle swarm optimization, to choose and combine QoS-aware web services. Selecting and composition of Web services as well as creating composite service has done by using particle swarm algorithm. The algorithm used quality parameters is defuzzy and dynamic and uses a global strategy to solve the problem of Web service composition. The comparison between the two methods of graph-based greedy algorithm and particle swarm algorithm shows that despite the need for more time to implement particle swarm algorithm, it provides more consistent solutions than greedy method [13].

Yu & Chen presented a method that optimizes ant colony algorithm in applying service composition in cloud computing. Many traditional methods can only find a sequence of service composition and cannot encompass service database in several clouds. Therefore, the challenge to find an effective composition across multiple clouds is very important because it obtains more efficient service composition. In this study, a greedy algorithm called Greedy-WSC and an algorithm based on ant colony optimization called as ACO-WSC are provided. The aim of this method is to select practical and effective composition of clouds and minimum use of clouds. Experimental results show that the ant colony optimization technique can be effective and efficient in finding cloud composition with minimum clouds [14].

Shen et al. presented auto optimization QoS-aware service composition in run-time, which contains a framework and implementation of the prototype in a context-sensitive environment. The mechanism of reselecting context-aware services includes a framework based on context similarity analysis to improve the quality of the international service to carry out all tasks of composition and it also includes a genetic algorithm (MP-CS-repair) to reduce the additional delay in the calculations, when a service reprogramming occurs. Effectiveness, scalability, development and productivity by comparative tests on a prototype for this framework and credibility is approved. The mechanism of reselecting context-aware is compatible with the frequencies changes if the text and the user's requirements. Despite the benefits of this mechanism, the uncertainty of QoS and text information on the composition of QoS-aware services is not considered [15].

One of the major challenges faced by the service composition develops is how to select effectively a set of services across different independent areas. As an example of a service composition can satisfy the user's QoS constraints. Li et al. developed a method to solve the problem of global optimization selection (GOS) for QoS-aware Web service based on prediction mechanisms of local QoS values. GOS consists of two parts: first, selection of local preprocessing services algorithm can be used on the basis of predicting QoS parameters changes to increase the performance of compound services at run-time. Second, GOS aims to raise performance in the implementation of the global selection through decreasing the quality of intensive service operations. The simulation results show that GOS is an excellent choice and has better productivity than other composition methods and implementation cost is less than existing methods. However, this approach needs to be further improved in the field of efficient ontology management [16].

Immonen et al. designed a framework for defining the steps required for designing service composition and execution to gain access to a reliable service composition. Then they reviewed the literature of existing methods and their proximity to reliable service composition to determine to what extent and how the existing methods are consistent with the criteria of proposed framework. The literature review showed that none of the existing methods do not cover fully all the criteria of the proposed framework. Methods are often isolated and are not compatible with existing standards and practices. Most of all, lack supportive tools. The changes needed in the service composition and design of RE blocks methods application in industrial use. It is clear that the development of standard methods and tools to encourage the industry to change their ways in engineering services and service composition is inevitable [17].

Wu et al. used Dynamic Skyline Composition Algorithm for combining web services based on service quality, where the compositions of web-based services are performed dynamically. With the emergence of new services, old services will be removed and quality of service also changes. The advantages of this approach is to identify and select the best web services through a set of services based on service quality, and also using linear compositions to reduce the number of selected web services from the set of available services. The disadvantage of this study was lack of evaluating the algorithm on a composition of web services in real and unreal data sets [18].

One of key characteristics of service-oriented architecture is that it allows a business process for flexible services composition. Although previous works related to service composition pave the way to for automatic composition, these techniques limit the implementation when the composition of complex workflows based on functional requirements, partly reach to the large search space of the services. To solve this problem, Lee et al. have proposed a new concept in the scope of services. Unlike existing abstract services which possess fixed service interfaces, a prospect service has a flexible interface to allow functional flexibility. Furthermore, they defined a meta-model to specify service patterns with prospect services and adaptable workflow constructs to model flexible and adaptable process templates. An automated instantiation method is introduced to instantiate concrete processes with different functionalities from a service pattern. Since the search space for automatically instantiating a process from a service pattern is greatly reduced compared to that for automatically composing a process from scratch, the proposed approach significantly improve the feasibility of automated composition. Simulation results demonstrate that the proposed automated instantiation method is efficient. [19].

Nagamouttou et al. developed a new model to study the web service composition that uses Enhanced Stacked Automata Model (ESAM). The correctness properties of the non-deterministic system have been evaluated based on the properties like dead transition, deadlock, liveness and reachability. Initially web services are composed using Business Process Execution Language for Web Service (BPEL4WS) and it is converted into ESAM and it is transformed into Promela language, an input language for Simple ProMeLa Interpreter (SPIN) tool. The model is verified using SPIN tool and the results revealed better recital in terms of finding dead transition and deadlock in contrast to the existing models. [20].

Silva et al. presented a genetic programming method for the selection and composition of QoS-aware web services. This article describes three methods of GP for composition of QoS-aware web services proposed: In the first method (GP-I), fitness function is used to encourage accuracy of composition solutions. In the second method (GP-II), the accuracy of solutions is determined by limiting initial population and genetic operators are used. The third method (GP-III) enables the second method to create solutions with selected structures. The results showed that the composition produced by the second method better quality than the first method. In comparison to PSO algorithm, the second method uses lower fitness than those graph-based PSO algorithm, as well as the execution time is shorter. The third method manages selecting structure and composition possibility without reducing the quality or the need for long implementation time. Finally, they concluded that GP method can be scalable in automatic web services composition and enough flexibility can be considered for the implementation of complex composition structure. In the second and third methods, obtained solutions need to be translated in the form of a tree diagram that is incurred additional costs and is considered as disadvantages of this method [21].

6. Conclusion

In recent years, the number of Web services that have the same functionality but different quality are increased. Also, through developing Web services, service engineers were seeking algorithms for automatic service composition that not only do operations for composition of services for thousands of service, it also provides the quality requirements of users.

In fact, web services allow exchange between computers in a heterogeneous environment that is composed of different systems. Basic services are often unable to meet user needs; therefore, they need to combine Web services to increase efficiency and performing more complex services of the important topics in the field of web services.

Most of the methods available for Web service composition regardless of the user's interests, returns only one solution that in some certain situations are efficient and in terms of flexibility and diversity are very low. In this study, a method is provided that, takes into account the various parameters of quality of service, and provides several solution based on graph coloring. The results of the simulation show that the MGC-K and MGC-TopK methods, in addition to all the advantages of optimum service composition have better results in the entire run time and memory consumption than previous methods.

Suggestions for future studies can be as follows:

1. Improving algorithm in terms of filtering running time.
2. More attention to management services with multiple inputs and outputs that have similarities between their inputs and outputs.
3. Use factors in filtering operations.
4. The use of heuristic algorithms for service selection and composition.

References

[1] E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, “HTN planning for web service composition using SHOP2,” Web Semantics: Science, Services and Agents on the World Wide Web 1, no. 4: 377-396, 2004.

[2] W. Jiang, S. Hu and Z. Liu, “Top K query for QoS-aware automatic service composition,” IEEE Transactions on Services Computing 7, no. 4: 681-695, 2014.

[3] Allahverdipour A, Soleimanian Gharehchopogh F. A New Hybrid Model of K-Means and Naïve Bayes Algorithms for Feature Selection in Text Documents Categorization. Journal of Advances in Computer Research. 2016 Nov 25.

[4] Omidvar R, Eskandari A, Heydari N, Hemmat F, Feyli M. An Improved SSPCO Optimization Algorithm for Solve of the Clustering Problem. Journal of Advances in Computer Research. 2017 Mar 19.

[5] Pourabdi L, Harounabadi A. Providing a Method to Identify Malicious Users in Electronic Banking System Using Fuzzy Clustering Techniques. Journal of Advances in Computer Research. 2017 May 1;8(2):67-77.

[6] S. Deng, L. Huang, W. Tan and Z. Wu, "Top-automatic service composition: A parallel method for large-scale service sets," IEEE Transactions on Automation Science and Engineering 11, no. 3: 891-905, 2014.

[7] S. Deng, L. Huang, W. Tan and Z. Wu, "Top-automatic service composition: A parallel method for large-scale service sets," IEEE Transactions on Automation Science and Engineering 11, no. 3: 891-905, 2014.

[8] A. Zhou, S. Huang and X. Wang, "Bits: A binary tree based web service composition system," International Journal of Web Services Research (IJWSR) 4, no. 1: 40-58, 2007.

[9] A. Marconi, M. Pistore, P. Poccianti and P. Traverso, "AutomatedWeb Service Composition at Work: the Amazon/MPS Case Study," In IEEE International Conference on Web Services (ICWS 2007), pp. 767-774. IEEE, 2007.

[10] Z. Brahmi, "QoS-aware Automatic Web Service Composition based on Cooperative Agents," In Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on, pp. 27-32. IEEE, 2013.

[11] S. Deng, B. Wu, J. Yin and Z. Wu, "Efficient planning for top-K Web service composition," Knowledge and information systems 36, no. 3: 579-605, 2013.

[12] G. Zou, Y. Gan, Y. Chen and B. Zhang, "Dynamic composition of Web services using efficient planners in large-scale service repository," Knowledge-Based Systems 62: 98-112,2014.

[13] A.S. da Silva, H. Ma and M. Zhang, "A graph-based particle swarm optimisation approach to qos-aware web service composition and selection," In 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 3127-3134. IEEE, 2014.

[14] Q. Yu, L. Chen and B. Li, "Ant colony optimization applied to web service compositions in cloud computing," Computers & Electrical Engineering 41: 18-27, 2015.

[15] Y.H. Shen and X.H. Yang, "A self-optimizing QoS-aware service composition approach in a context sensitive environment," Journal of Zhejiang University SCIENCE C 12, no. 3: 221-238, 2011.

[16] M. Li, D. Zhu, T. Deng, H. Sun, H. Guo and X. Liu, "GOS: a global optimal selection strategies for QoS-aware web services composition," Service Oriented Computing and Applications 7, no. 3: 181-197, 2013.

[17] A. Immonen and D. Pakkala, "A survey of methods and approaches for reliable dynamic service compositions," Service Oriented Computing and Applications 8, no. 2: 129-158, 2014.

[18] J. Wu, L. Chen and T. Liang, "Selecting dynamic skyline services for QoS-based service composition," Applied Mathematics & Information Sciences 8, no. 5: 2579, 2014.

[19] C. H. Lee, S. Y. Hwang, I. L. Yen and T. K. Yu, "A service pattern model for service composition with flexible functionality," Information Systems and e-Business Management 13, no. 2: 235-265, 2015.

[20] D. Nagamouttou, I. Egambaram, M. Krishnan and P. Narasingam, "A verification strategy for web services composition using enhanced stacked automata model," SpringerPlus 4, no. 1: 1, 2015.

[21] A. S. da Silva, H. Ma and M. Zhang, "Genetic programming for QoS-aware web service composition and selection," Soft Computing: 1-17, 2016.