A Top-k QoS-Optimal Service Composition Approach Under Dynamic Environment

Cheng Tian[1](✉), Zhao Zhang[2,3], Sha Yang[2,3], Shuxin Huang[2,3], Han Sun[2,3], Jing Wang[2,3], and Baili Zhang[1]

[1] School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
220181734@seu.edu.cn

[2] State Key Laboratory of Smart Grid Protection and Control, Nanjing 211106, China

[3] Nari Group Corporation, Nanjing 211106, China

Abstract. To deal with low efficiency of existing Web service composition algorithms under dynamic environment, this paper proposes a Top-k QoS-optimal service composition algorithm based on QWSC-K static algorithm. The algorithm first considers the influence of dynamic environment on the existing service composition and service dependency graph, and classifies the changed services in dynamic environment. Moreover, different strategies are adopted to obtain the intermediate results of service composition for different categories of services. The influence of different service categories on the service dependency graph is analyzed to realize the updating of the service dependency graph. Finally, based on the updated service dependency graph and the intermediate results of the service composition, the adaptive incremental update can be applied to change the original service composition results, avoiding the time-consuming re-query process. Experiments show that the algorithm is efficient and accurate in dynamic environment.

Keywords: Dynamic environment · Service composition · Intermediate results · Dependency graph · Incremental updating · Adaptive

1 Introduction

Due to the limited functions of single Web service, how to combine exiting Web services to meet complex business requirements has become a research focus [1, 2]. The existing Top-k QoS-optimal service composition schemes have been utilized to provide users with multiple service compositions to meet their requirements [3–6]. However, most of these service composition methods are based on the static environment with fixed services. Faced with the dynamic changing service environment, it is often necessary to recalculate the built composite services, which is very inefficient. Therefore, many researchers propose their own solutions to the service composition problem in dynamic environment.

Literature [7, 8] proposes that artificial intelligence planning method can be used to process dynamic services and update composite services accordingly. However, because

QoS of services is not taken into account in the process of service composition update, these methods usually cannot guarantee that the updated composite services still meet the non-functional requirements.

Literature [9] proposed a service composition method oriented to dynamic service environment. It is based on the shortest path graph search algorithm to search for chain-like composite services. However, this method cannot search for non-chain composite services that actually exist, such as composite services that can be modeled as Directed Acyclic Graph (DAG).

Literature [10, 11] respectively gives the methods of automatic service composition oriented to dynamic environment. However, these methods only focus on the changes of services functional factors (such as the change of service interface), and do not consider the influence of non-functional factors (such as service QoS). Thus, it is difficult to recommend a service composition under constrains of user QoS. Contrary to the above two works, literatures [12, 13] focus on improving the quality of service according to the change of QoS of atomic services.

Literature [14] proposes a composite service adaptive algorithm to deal with the dynamically changing service environment. However, this method only performs adaptive updating for a single optimal service composition, and does not solve the adaptive updating problem of the first k optimal service composition.

Therefore, this paper proposes a Top-k QoS-optimal service composition algorithm DQWSC-K in dynamic environment. The algorithm is based on the updated service dependency graph and uses an incremental adaptive algorithm to partly update the affected service state, avoiding time-consuming re-composition process.

The second section introduces the knowledge related to the proposed algorithm. The third section describes the service composition algorithm in dynamic environment. Section 4 gives the experimental results. The last part is the summary and prospect of the paper.

2 Related Knowledge

2.1 Service Dependency Graph Model

In order to accurately express the association relationship of Web services in the Web services set and their QoS information, this paper uses the service dependency graph $G = (V, E)$ to model the Web services set.

In G, the node set V represents a set of services. Each service W_i is represented as a node in G. The directed edge set E represents the set of service matches. The set satisfies: $\forall e_k \in E$, $e_k = (v_n, v_m, tag_k)$, in which v_n and v_m respectively represent the service nodes corresponding to the start and the end of the edge. The tag_k satisfies the following relationship:

1. $tag_k \subseteq v_n.O$
2. $tag_k \subseteq v_m.I$

It is necessary to point out that when there is a service request R, the entrance service node Start and the exit service node End are dynamically generated in the graph. The two service nodes satisfy the following relationship:

Table 1. Service node.

Service name	Input parameters	Output parameters	QoS value
v_1	*A*, *B*, *C*	*D*	900
v_2	*A*, *B*	*E*, *F*	100
v_3	*C*, *E*	*H*	200
v_4	*C*, *F*	*G*	500
v_5	*L*, *J*	*D*	600
v_6	*K*	*H*	500
v_7	*H*	*D*	200
v_8	*G*	*H*	500

1. $Start.I = \varnothing; Start.O = R.I$
2. $End.I = R.O; End.O = \varnothing$

According to the above definition, if the service set is the service described in Table 1, the service request is R = <I, O>, in which the request input parameter is I = {A, B, C} and the request output parameter is O = {D}. The service dependency graph constructed is shown in Fig. 1.

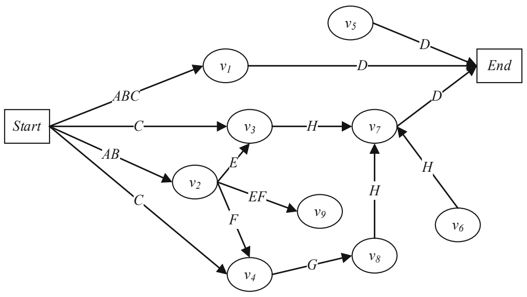


Fig. 1. Service dependency graph.

2.2 QWSC-K Algorithm Structure and Process

The Top-k QoS-optimal service composition algorithm QWSC-K [15] under static environment mainly includes three modules: the service filtering module, the obtaining the combined path sequence module and the combined path sequence conversion module. Firstly, in the service filtering module, an effective hierarchy filtering algorithm is adopted to filter the initial service set. The search space of the graph and the time complexity of the entire algorithm can be reduced greatly. Secondly, in the obtaining the combined path sequence module, the filtered service candidate set is used to construct a service dependency graph. Then traverse the service dependency graph. In the traversal process, the Top-k composition path information sequences associated with each service node are calculated and saved. The sequences are sorted according to global QoS values. Finally, in the combined path sequence conversion module, the Top-k composition path information sequences at the exit service node End are converted into the solution of the final Top-k service composition.

3 DQWSC-K Algorithm Description

The DQWSC-K algorithm mainly include four modules: dynamic service acquisition and classification module, service composition intermediate result acquisition module, service dependency graph updating module and incremental adaptive updating module. The algorithm framework is shown in Fig. 2.

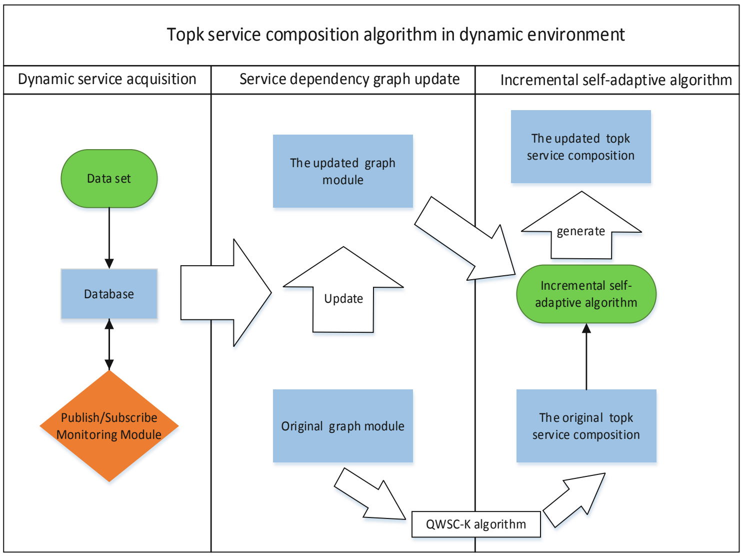
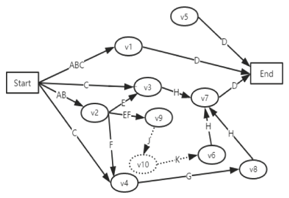
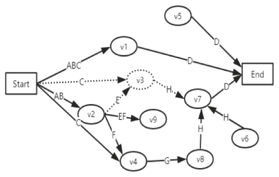
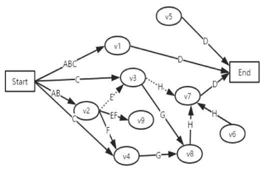
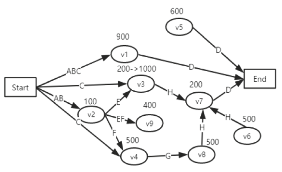

Topk service composition algorithm in dynamic environment
Dynamic service acquisition
Service dependency graph update
Incremental self-adaptive algorithm
Data set
Database
Publish/Subscribe Monitoring Module
The updated graph module
Update
Original graph module
QWSC-K algorithm
The updated topk service composition
generate
Incremental self-adaptive algorithm
The original topk service composition

Fig. 2. Top-k service composition algorithm framework in dynamic environment.

3.1 Dynamic Service Acquisition and Classification Module

In accordance with the method described in literature [16], this paper receives events related to dynamic services from the publish/subscribe network, and uses soapUI to monitor and obtain Web services and their QoS. For the received dynamic services, this paper divides them into four categories according to different processing strategies: (1) new services: the services newly added to the service set; (2) failed services: the services that need to be deleted in the service set; (3) interface change services: the services existing in the original service set, but their input and output parameter set changes; (4) the services with changed QoS. As shown in Table 2, compared with Fig. 1, v10 in Table 2 (1) is a new service, v3 in Table 2 (2) is a failed service, v3 in Table 2 (3) is an interface change service, and v3 in Table 2 (4) is a service with changed QoS value from 200 to 1000.

Table 2. Dynamic services.

Dynamic services	Example
(1)New services	
(2)Failed services	
(3)Interface change services	
(4)Services with changed QoS	

3.2 Service Composition Intermediate Result Acquisition Module

In the process of running the QWSC-K algorithm, the effective service candidate set was obtained by using the hierarchical filtering algorithm. Then, according to the effective service candidate set, the combined path traversal algorithm is used to obtain the parameter source table and the combined path sequence information source table generated during the traversal process. These two tables are the intermediate result of service combination.

However, the intermediate state results obtained by the QWSC-K algorithm may be incomplete for new services and interface change services. The reason is that the new service and the interface change service could lead to a previously inaccessible service node becoming accessible, which may eventually lead to generating a new service composition. So when traversing the graph, the state of every reachable service node need to be recorded in order to check whether the original unreachable node becomes reachable due to the appearance of the new services or interface change services. However, in reality, the QWSC-K algorithm will directly filter out the previously unreachable nodes during the filtering phase, so new service combinations cannot be generated. Therefore, for the new services and the interface change services, the QWSC-K algorithm needs to be improved to remove the filtering stage, so that the intermediate states of all reachable service can be obtained.

The improved QWSC-K algorithm (IQWSC-K) is similar to the QWSC-K algorithm. After the filtering phase is removed, the parameter source table is obtained through the entire service set instead of the original valid service list. At the same time, the service dependency graph is directly constructed from the original service set.

For the failed services and the services with changed QoS, the emergence of these two types of dynamic services will not lead to the original unreachable service node reachable. Therefore, the QWSC-K algorithm can be directly used to filter and calculate the intermediate state result.

3.3 Service Dependency Graph Update Module

It can be seen from Sect. 3.1 that four types of dynamic services will have an impact on the structure or attributes of the service dependency graph. According to the type of dynamic service, the following strategies are used to update the service dependency graph respectively. First of all, for the new service, the corresponding node is added in the service dependency graph. And according to the matching relationship between this node and other nodes, the corresponding edge is added in the service dependency graph. Then, the precursor service node of the new service node is obtained according to the saved parameter source table. And the Top-k composite path sequence associated with the new node is obtained from the precursor service node and saved to the composite path information source table. Secondly, for the failed service, the corresponding node and its associated edges are removed from the service dependency graph. And the table items associated with the node are removed from the composite path information source table and parameter source table. Thirdly, for the service with changed QoS, the QoS values of the corresponding nodes in the service dependency graph are updated. And the Top-k composite path sequence associated with this node is found from the composite path information source table to update its global QoS value. Finally, for the interface change service, it is handled in two steps: (1) Delete the original service in the service dependency graph; (2) Handle the service as a new service. Thus, in the manner described above, we have transformed the four dynamic services into the three dynamic services for processing.

3.4 Incremental Adaptive Update Module

At present, for Top-k service composition algorithm, when dynamic services arrive, the service composition engine reruns the service composition process and generates a new Top-k service composition scheme. However, this approach is less efficient in a dynamic environment with real-time change. Through the analysis of the existing QWSC-K algorithm, it is found that in the process of generating service composition, each service node saves the composite path sequence from the starting node to the service node. This also shows that for each service node, only the change of service state in the precursor service set will have a direct impact on it. Thus, the generation of each dynamic service can only affect its associated subsequent services, while the state of its precursor services does not change. So, in the process of service composition, the service nodes affected by the dynamic service are updated in turn according to the state of the dynamic service and the intermediate results of the saved service nodes. After that, a new Top-k service composition scheme can be obtained.

When the dynamic service is received, according to specific categories of dynamic service, incremental adaptive algorithm (IA_Alg) takes the corresponding processing strategy and generates new Top-k service composition. The specific steps of IA_Alg algorithm are as follows:

(1) Analyze the impact of dynamic services and judge whether dynamic services may cause changes in the intermediate state of other services. Specifically, for failed services, interface change services and services with changed QoS, determine whether this dynamic service exists in the saved composite path information list. If it exists, it indicates that the new service may affect the state of the subsequent service nodes. If it does not exist, then the dynamic service will not affect other service nodes. For a new service, determine whether it can trigger a new service node. If so, it means that the service may generate a new service combination. It needs to continue to consider the influence of subsequent nodes. If not, it means that there is no influence.

(2) Update the status of the affected service. First, the priority queue PQ is used to store the dynamic services identified in the first step that may affect the state of other services. Then the services in PQ are popped up by loop and processed accordingly. In PQ, service nodes are saved in order from small to large according to their global QoS value in order to avoid repeated update of service state.

For different dynamic service types, the specific process of updating the service state is different. For the new service or the interface change service, it can make the original unreachable service become the reachable service. Therefore, on the one hand, for each service popped up from the PQ, it is necessary to determine whether it enables other service nodes to change from the unreachable state to the reachable state based on its output parameters. That is, each output parameter of the dynamic service need to be traversed to see if it can trigger a new service node. If it can trigger a new service node, the new service node needs to be added to PQ. On the other hand, it is necessary to determine whether these dynamic services have an impact on the Top-k composite path sequence stored in the subsequent service. Then update the state of their subsequent service nodes and add the affected subsequent services to the PQ. For the failed service or the service with changed QoS, it is only necessary to determine whether they have an impact on the Top-k combined path sequence stored in subsequent services. If so, directly update the status of the subsequent service nodes and add the affected subsequent services to PQ.

(3) After all the services in the priority queue PQ are processed, determine whether the composite path sequence of the terminating service node End is updated. If the sequence is updated, the path sequence transformation algorithm will convert the new Top-k composite path sequence into the solution of the final service combination. If not, the original solution of the Top-k service composition is returned.

4 Experiment

4.1 Experimental Procedure

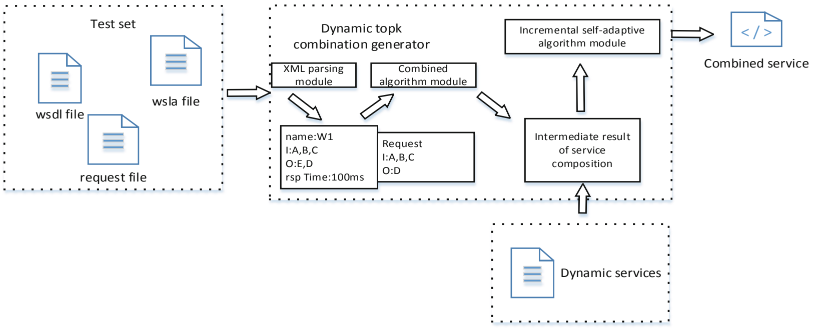

Test set
wsdl file
wsla file
request file
Dynamic topk combination generator
XML parsing module
Combined algorithm module
name:W1
I:A,B,C
O:E,D
rsp Time:100ms
Request
I:A,B,C
O:D
Incremental self-adaptive algorithm module
Intermediate result of service composition
Dynamic services
Combined service

Fig. 3. Dynamic environment experiment system.

The experimental test set uses the data set generated by the WSBen tool, which contains 5 test sets with different number of services (200–10000). In the experiment, we also generate dynamic services in a random way for each test set. There are three types of files on each test set: the WSDL file describes the input and output parameters of the service; the WSLA file describes QoS information of the service; the request file describes the user's request. The whole experimental system is shown in Fig. 3.

This paper mainly selects QWSC-K algorithm based on re-execution as the comparison object of DQWSC-K algorithm. This paper mainly adopts the following two evaluation indexes: query response time and accuracy.

4.2 Query Response Time Evaluation

Because the intermediate results of the service composition obtained are different for the different dynamic service types in the DQWSC-K algorithm, the intermediate results of the service composition obtained for new services and interface change services are different from those for failed services and services with changed QoS. Therefore it is necessary to conduct experimental verification for these two situations. In order to verify the time performance of the algorithm, this paper conducts two sets of experiments: the same k value, the same number of dynamic services, and different service set sizes; the same k value, the same service set size, and different numbers of dynamic service. The specific experimental conditions are as follows:

Experiment 1: The influence of the same k value, the same service set size and different dynamic service quantity on the algorithm.

Figure 4 shows that when the dynamic service is a new service or an interface change service, k value is 5 and service size n = 1000, different dynamic service quantity has an impact on the execution time of DQWSC-K algorithm and QWSC-K algorithm. Figure 5 shows the impact of dynamic service as a failed service or a service with changes QoS on the execution time of DQWSC-K algorithm and QWSC-K algorithm under the same condition. It can be seen from Fig. 4 and Fig. 5 that the execution time of both DQWSC-K algorithm and QWSC-K algorithm increases with the increase of the number of dynamic services. But the execution time of QWSC-K algorithm increases with large amplitude, while that of DQWSC-K algorithm is smaller. This is because for QWSC-K algorithm, each addition of a dynamic service means an increase in the time to search the entire service space, the time consumption increases with large amplitude. In contrast, DQWSC-K algorithm can avoid re-search and consume less time, so the increase amplitude is lower.

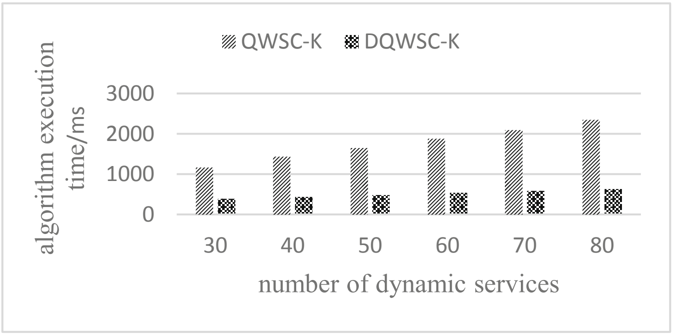


Fig. 4. Dynamic service is the new service or interface change service.

Experiment 2: The influence of the same k value, the same number of dynamic services, and different service set sizes on the algorithm.

Figure 6 shows that when the dynamic service is a new service or an interface change service, the value of k is 5 and the number of dynamic services is 10, different service set sizes have an impact on the execution time of DQWSC-K algorithm and QWSC-K algorithm. Figure 7 shows the impact of dynamic service as a failed service or a service with changed QoS on the execution time of DQWSC-K algorithm and QWSC-K algorithm under the same condition. As can be seen from Fig. 6 and Fig. 7, the execution time of the QWSC-K algorithm obviously depends on the size of the Web service set.

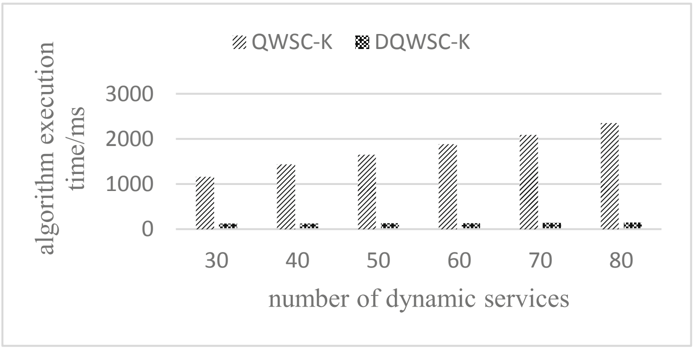


Fig. 5. Dynamic service is the failure service or service with changed QoS.

The more services there are, the longer the query execution time will be. This is because each time a dynamic service is generated, the QWSC-K algorithm needs to re-execute the query and search the entire service space, the execution time depends mainly on the number of services. But for the DQWSC-K algorithm, because the incremental adaption only needs to update the service state partially affected by the dynamic service, the execution time is lower than that of QWSC-K algorithm.

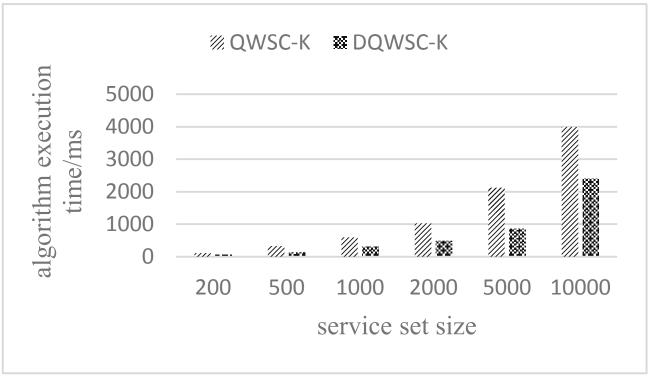


Fig. 6. Dynamic service is the new service or interface change service.

4.3 Accuracy Assessment

This section evaluates the accuracy of the two algorithms. It refers to whether the Top-k service combination returned by the two algorithms satisfies the query request and guarantees global QoS optimization. Because the QWSC-K algorithm has been proved theoretically and experimentally that the service composition it returns is globally optimal, by comparing whether the global QoS returned by QWSC-K algorithm is equal to the global QoS returned by DQWSC-K algorithm, we can determine whether the

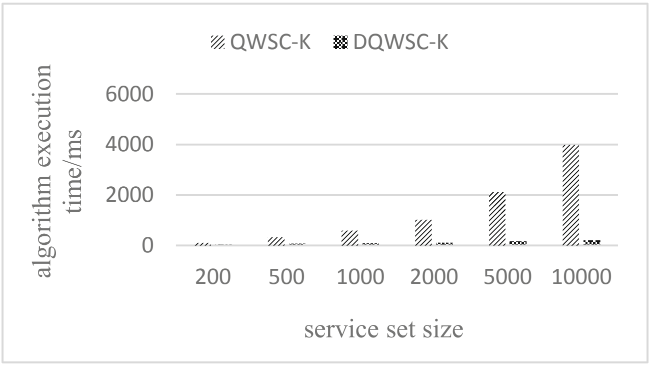


Fig. 7. Dynamic service is the failure service or service with changed QoS.

service combination returned by DQWSC-K algorithm is the global optimal QoS result. We respectively use two algorithms to conduct experiments based on the service dependency graph in Fig. 8: take the dynamic service as w_4, its QoS value changes from 10 to 30, and k is 3. The final experimental result is shown in Table 3.

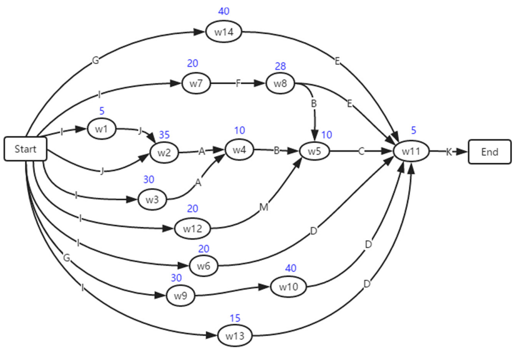
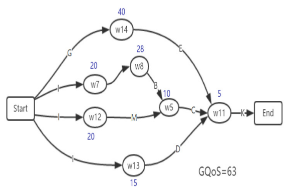
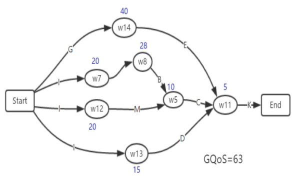
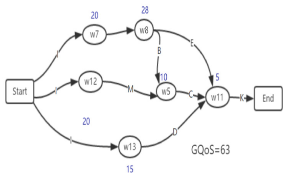
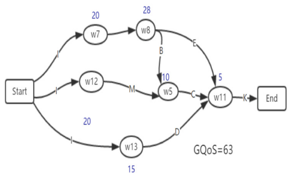
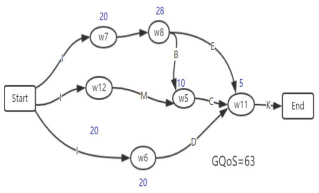
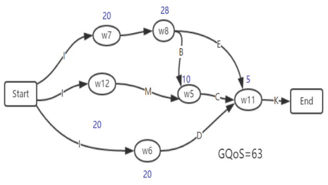


Fig. 8. Service dependency graph.

The result shows that the Top-k service combination results returned by QWSC-K algorithm and DQWSC-K algorithm are consistent and can be executed correctly. And the associated global QoS is also the same. This shows that the DQWSC-K algorithm can deal with the dynamic changing service environment and automatically update the service combination as required to ensure that the updated results still meet the query request and the global QoS is optimal.

Table 3. Updated service composition result table.

QWSC-K algorithm	DQWSC-K algorithm
 Start w14 w7 w8 w12 w5 w11 w13 End G I E B M C K D 40 28 20 10 5 15 GQoS=63	 Start w14 w7 w8 w12 w5 w11 w13 End G I E B M C K D 40 28 20 10 5 15 GQoS=63
 Start w7 w8 w12 w5 w11 w13 End I E B M C K D 20 28 10 5 15 GQoS=63	 Start w7 w8 w12 w5 w11 w13 End I E B M C K D 20 28 10 5 15 GQoS=63
 Start w7 w8 w12 w5 w11 w6 End I E B M C K D 20 28 10 5 GQoS=63	 Start w7 w8 w12 w5 w11 w6 End I E B M C K D 20 28 10 5 GQoS=63

5 Conclusion

This paper presents a Top-k QoS-optimal service composition algorithm under dynamic environment, which is based on the QWSC-K algorithm. This algorithm uses incremental adaptive algorithm to dynamically update the state of some affected services according to different service categories. The algorithm does not recalculate and search the entire service set space, which ensures the efficiency of the algorithm. The next step is to try to use the idea of distributed parallel or approximate computation to solve the Top-k QoS-optimal service combination problem in the dynamic service environment.

Acknowledgement. This work was partly supported by Open Research Fund from State Key Laboratory of Smart Grid Protection and Control, China, Rapid Support Project (61406190120) and the National Key R&D Program of China (2018YFC0830200).

References

1. Deng, S., Huang, L., Tan, W.: Top-k automatic service composition: a parallel method for large-scale service sets. IEEE Trans. Autom. Sci. Eng. **11**(3), 891–905 (2014)

2. Tan, W., Fan, Y., Zhou, M.: A petri net-based method for compatibility analysis and composition of web services in business process execution language. IEEE Trans. Autom. Sci. Eng. **6**(1), 94–106 (2009)
3. Benouaret, K., Benslimane, D., Hadjali, A., et al.: Top-k web service compositions using fuzzy dominance relationship. In: 2011 IEEE International Conference on Services Computing (SCC), pp. 144–151. IEEE, Washington (2011)
4. Almulla, M.. Almatori, K., Yahyaoui, H.: A QoS-based fuzzy model for ranking real world web services. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 203–210. IEEE, Washington (2011)
5. Wang, X.L., Huang, S., Zhou, A.Y.: QoS-aware composite services retrieval. J. Comput. Sci. Technol. **21**(4), 547–558 (2006)
6. Jiang, W., Hu, S., Liu, Z.: Top k query for QoS-aware automatic service composition. IEEE Trans. Serv. Comput. **7**(4), 681–695 (2014)
7. Yan, Y., Poizat, P., Zhao, L.: Repair vs. recomposition for broken service compositions. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 152–166. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17358-5_11
8. Yan, Y., Poizat, P., Zhao, L.: Self-adaptive service composition through graphplan repair. In: 2010 IEEE International Conference on Web Services, pp. 624–627. IEEE, Miami (2010)
9. Kalasapur, S., Kumar, M., Shirazi, B.A.: Dynamic service composition in pervasive computing. IEEE Trans. Parallel Distrib. Syst. **18**(7), 907–918 (2007)
10. Wang, H., Wu, Q., Chen, X., et al.: Adaptive and dynamic service composition via multi-agent reinforcement learning. In: 2014 IEEE International Conference on Web Services, pp. 447–454. IEEE, Anchorage (2014)
11. Geyik, S.C., Szymanski, B.K., Zerfos, P.: Robust dynamic service composition in sensor networks. IEEE Trans. Serv. Comput. **6**(4), 560–572 (2013)
12. Feng, Y., Ngan, L.D., Kanagasabai, R.: Dynamic service composition with service-dependent QoS attributes. In: IEEE 20th International Conference on Web Services, pp. 10–17. IEEE Computer Society, Santa Clara (2013)
13. Groba, C., Clarke, S.: Opportunistic service composition in dynamic ad hoc environments. IEEE Trans. Serv. Comput. **7**(4), 642–653 (2014)
14. Lv, C., Jiang, W., Hu, S.: Adaptive method of composite service oriented to dynamic environment. Chin. J. Comput. **39**(2), 305–322 (2016)
15. Li, G., Wen, K., Wu, Y., Zhang, B.: Topk service composition algorithm based on optimal QoS. In: Sun, X., Pan, Z., Bertino, E. (eds.) Cloud Computing and Security, ICCC 2018, LNCS, vol. 11064, pp. 309–321. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00009-7_29
16. Yan, W., Hu, S., Muthusamy, V., et al.: Efficient event-based resource discovery. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, pp. 1–12. Association for Computing Machinery, New York (2009)