A Decomposition-based Method for QoS-aware Web Service Composition with Large-scale Composition Structure

Lianyong Qi[1,2], Xiaona Xia[1], Jiancheng Ni[1],Chunmei Ma[1], Yanxue Luo[1]
[1] Computer Science College
Qufu Normal University, P.R. China
[2] State Key Lab. for Novel Software Technology
Nanjing University, P.R. China
Email: lianyongqi@gmail.com

***Abstract*—Web service composition (WSC), has been considered as a promising way for integrating various distributed computing resources for complex application requirements. However, for a QoS-aware WSC problem with large composition structure, much computation time is necessary to determine the QoS-optimal composite solution, which challenges the service composition applications in large-scale collaboration environment. In view of this challenge, a Decomposition-based service Composition Method, named DCM, is introduced in this paper. Firstly, the proposed DCM method decomposes the large-scale composition structure into many small-scale composition segments, through mixed integer programming. Then for each small-scale composition segment, find a QoS-optimal composite solution with less time cost. Through experiments, we demonstrate that the execution efficiency of DCM outperforms the present service composition methods, especially for the WSC problems with large-scale composition structure.**

Keywords-web service composition; QoS; Decomposition; mixed integer programming

I. INTRODUCTION

In recent years, the web service technology has gained more and more attention and is becoming the de facto standard for integrating various distributed computing resources for complex application requirements [1-3]. Through functional encapsulation, a web service could be advertised by the service provider, and invoked by an end user via the pre-provided accessible interfaces. The easy-to-use property of web service brings more convenience for both service providers and end users, and greatly benefits the flexible integration of cross-domain applications.

However, the function that a single web service could provide is usually limited, compared to the complex computing requirements from end users. Therefore, to compose various component services into a more powerful composite service (WSC) has been considered as a promising way to satisfy the end users' complex requirements. For example, more and more WSC instances are deployed in the popular areas, e.g., Scientific Workflow, Electronic Commerce and Multimedia Delivery applications [4]. However, as there are many web services that share similar functionality, the candidate composite solutions for a WSC problem are multiple, not unique. In this situation, quality of service (QoS) could be recruited as an important discriminating factor, because a WSC process is usually accompanied with various QoS constraints from end users. For example, a smart-phone end user may expect that the total *latency time* of the composite multimedia service not exceed 2 seconds. Therefore, for a WSC problem, the next question is to find a QoS-optimal composite solution from the huge amount of candidates, while considering the various global QoS constraints from end users (QoS-aware web service composition).

Many researchers concentrate on this hot research topic and put forward some valuable methods for the QoS-aware WSC problems [4-9]. However, these proposed methods cannot work very well because of inefficiency. This is due to the fact that QoS-aware WSC is inherently a NP-hard problem [6], so it is usually time-consuming to find the QoS-optimal composite solution. Especially for the WSC problem with large-scale composition structure (i.e., the WSC process consists of many component tasks. For example, an e-Science composite process with dozens of tasks), the traditional service composition methods may fail in delivering satisfactory results within limited time period. Hence, it is of great challenge to study more efficient composition method. In view of this challenge, a novel Decomposition-based service composition method, named DCM, is proposed. DCM is based on mixed integer programming, which is an optimization approach that has a set of goal function that should be maximized or minimized and a set of constraint conditions that should be satisfied.

The remainder of the paper is organized as follows. In Section II, we put forward our motivation via a large-scale service composition example, and introduce the necessary preliminary knowledge. A Decomposition-based service composition method, named DCM, is proposed in Section III. Through the experiments in Section IV, we demonstrate the feasibility and efficiency of DCM in solving the WSC problems with large-scale composition structure. In SectionV, the proposed DCM method is evaluated, and finally we summarize the paper and point out our future research directions in SectionVI.

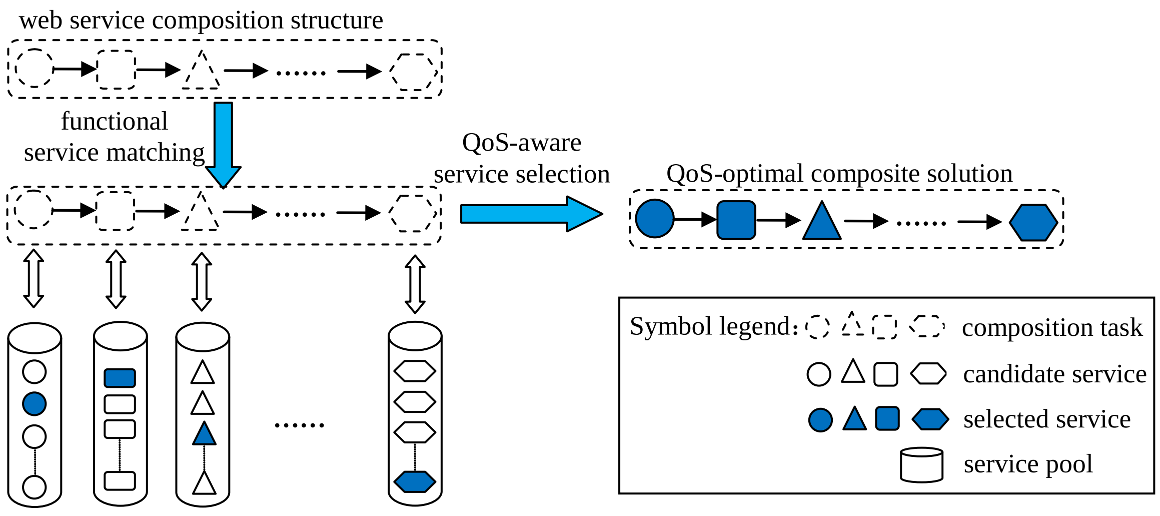
web service composition structure
functional service matching
QoS-aware service selection
QoS-optimal composite solution
......
Symbol legend:
composition task
candidate service
selected service
service pool

Figure1. An example of QoS-aware web service composition process

II. Preliminary Knowledge and Motivation

In order to facilitate the further discussion, some preliminary knowledge of QoS-aware WSC problem, is introduced here. Concretely, some basic concepts are listed as below.

1. ***TK***={tk_1, …, tk_i, …, tk_n}. tk_i ($1 \le i \le n$) denotes a composition task consisted in the service composition structure.
2. ***CR***={cr_1, …, cr_j, …, cr_m}. cr_j ($1 \le j \le m$) is a QoS criterion of web service and *m* is the number of QoS constraints requested by an end user. Commonly, for a web service *ws*, its quality value over QoS criterion cr_j could be denoted by $ws.cr_j$.
3. ***CST***$_{global}$={cst_1, …, cst_j, …, cst_m}. cst_j ($1 \le j \le m$) is a global QoS constraint over criterion $cr_j \in$ ***CR*** by an end-user and *m* is the number of QoS constraints.
4. ***Pool**$_i$*={ ws_i^1, …, ws_i^k, …, ws_i^l } is a service pool corresponding to composition task $tk_i \in$ ***TK***. Namely, each service $ws_i^k \in$ ***Pool**$_i$* ($1 \le k \le l$) could execute task tk_i($1 \le i \le n$), and *l* is number of functional qualified candidate services for task tk_i.
5. ***WGT*** = {wgt_1, …, wgt_j, …, wgt_m} is the weight value set for different QoS criteria. $wgt_j \in$ ***WGT*** ($1 \le j \le m$) is the weight value for QoS criterion cr_j. Generally, set ***WGT*** could be available from the end-user's preferences for different QoS criteria.
6. ***CompS*** = { $ws_1^{k_1}$, …, $ws_i^{k_i}$, …, $ws_n^{k_n}$ } is a functional qualified composite solution, where each component service $ws_i^{k_i}$ is the k_i-th candidate in ***Pool**$_i$* of task tk_i($1 \le i \le n$).

For example, as in Fig. 1, assume that there are four tasks in a composition process, then ***TK***={tk_1, tk_2, tk_3, tk_4}. Assume that for each task node, there are four candidate services; Then, for task node tk_i ($1 \le i \le 4$), its service pool ***Pool**$_i$* ={ ws_i^1, …, ws_i^4 }. If we only consider two QoS criteria: *latency* and *reputation*, whose constraints are respectively *latency*<1s and *reputation*>98%, then set ***CR***={*latency*, *reputation*} and ***CST**$_{global}$*={*latency* ∈ (0s, 1s), *reputation* ∈ (98%, 100%)}. If ***WGT*** = {wgt_{la}, wgt_{re}} where wgt_{la} = 0.7 and wgt_{re} = 0.3, then it means that *latency* is more important than *reputation* for the end user. Consider the example in Fig. 1, set { ws_1^2, ws_2^1, ws_3^3, ws_4^4 } is a composite solution that belongs to set ***CompS***.

Next, with the introduced basic concepts and the example in Fig. 1, we can clarify our motivation more clearly and formally. As there are *n* composition tasks {tk_1, …, tk_n} in set ***TK***, and for each task tk_i ($1 \le i \le n$) in ***TK***, there are *l* functional qualified candidates { ws_i^1, …, ws_i^l } in service pool ***Pool**$_i$*, the total number of functional qualified composite solutions is l^n. Next, our goal is to find a QoS-optimal composite solution ***CompS*** from all the l^n ones, while considering the end user's global QoS constraints ***CST**$_{global}$* and weight value set ***WGT***.

However, for a WSC problem with large-scale composition structure, the number of composition tasks (i.e., *n*) is usually large, which may lead to much time cost in order to find a QoS-optimal composite solution and disappoints the end user. In view of this, we put forward a composition method DCM, in the next section. DCM can decompose the large-scale composition structure into several small-scale ones, through which the composition efficiency could be improved significantly.

TABLE I. AGGREGATION TYPES AND AGGREGATION FUNCTIONS OF QOS CRITERIA

Aggregation type	Criterion	Aggregation function
Summation	price, duration	$CompS.cr_j = \sum_{i=1}^{n} ws_i.cr_j$
Average	reputation	$CompS.cr_j = \frac{1}{n}\sum_{i=1}^{n} ws_i.cr_j$
Multiplication	availability, success rate	$CompS.cr_j = \prod_{i=1}^{n} ws_i.cr_j$

III. A SERVICE COMPOSITION METHOD: DCM

In this section, a service composition method DCM is proposed, to improve the efficiency of WSC problems with large-scale composition structure.

A. *Hypotheses*

For the convenience of further discussions, some hypotheses are declared firstly.

Hypothesis1: Only negative QoS criteria (i.e., the smaller its value is, the better it is for user) are considered, e.g., *composition time*, *composition price*, as positive criteria could be transformed into negative ones by multiplying -1.

Hypothesis2: Only the sequential composition model illustrated in Fig. 1 is discussed, as other composition models (e.g., parallel, alternative and loops) could be transformed into the sequential model by present mature unfolding techniques [4-5].

Hypothesis3: In the sequential composition model, the aggregation types of various QoS criteria are different. In this paper, the common aggregation types and aggregation functions introduced in [4] are employed as in Table 1, where *CompS* denotes a composite solution and *n* is the number of tasks in composition structure.

B. *A QoS-aware service composition Method:* DCM

The main idea of our proposed DCM is: Firstly, the large-scale composition structure with *n* tasks is decomposed into $\lceil n/k \rceil$ small-scale composition segments, each with *k* ($1 \le k \le n$) tasks. Secondly, calculate the segment QoS constraints for each composition segment, through mixed integer programming. Thirdly, for each composition segment, find a QoS-optimal composite solution that satisfies the segment QoS constraints. Next, we will introduce these three steps of proposed DCM method separately.

(1) Step1: Decompose the large-scale composition structure into $\lceil n/k \rceil$ small-scale composition segments.

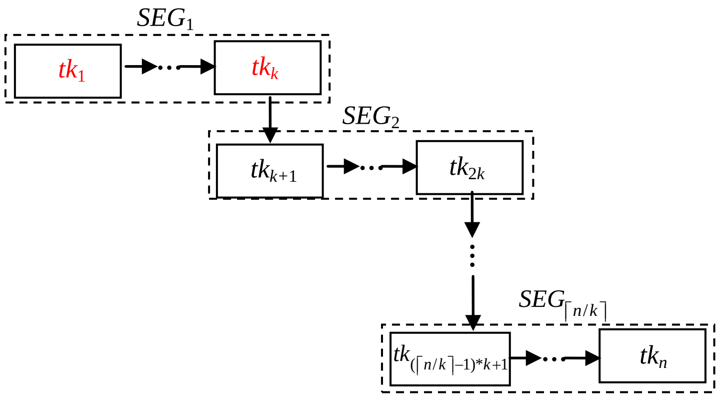

Figure2. Decompose the large-scale composition structure into small-scale composition segments

As the composition structure is large-scale (i.e, *n* is large), the needed time cost for finding a QoS-optimal composite solution may disappoints the end user. Therefore, we firstly decompose the large-scale composition structure into $\lceil n/k \rceil$ small-scale composition segments, each with *k* ($1 \le k \le n$) tasks. For example, as illustrated in Fig. 2, the composition structure with *n* tasks is decomposed into $\lceil n/k \rceil$ composition segments, which are denoted by $SEG_1=(tk_1, \ldots, tk_k)$, $SEG_2=(tk_{k+1}, \ldots, tk_{2k})$, …, $SEG_{\lceil n/k \rceil}=(tk_{(\lceil n/k \rceil -1)*k+1}, \ldots, tk_n)$.

(2) Step2: Decompose the global QoS constraints CST_{global} into $\lceil n/k \rceil$ segment QoS constraints via mixed integer programming.

After achieving $\lceil n/k \rceil$ small-scale composition segments, in this step, we calculate the segment QoS constraints for each composition segment, by decomposing the global QoS constraints CST_{global} via mixed integer programming technique.

Firstly, by the mathematical statistic technique, we calculate the minimal and maximal values over QoS criterion cr_j ($1 \le j \le m$) of candidate services for each task tk_i ($1 \le i \le n$), which are denoted by min_i^j and max_i^j respectively. Then a value range [min_i^j , max_i^j] is achieved, which depicts the QoS criterion cr_j's value distribution of task tk_i's candidate services. Then, we calculate the value distribution of candidates of each composition segment SEG_p ($1 \le p \le \lceil n/k \rceil$) over QoS criterion cr_j, which is denoted by [MIN_p^j , MAX_p^j], according to the aggregation functions introduced in TABLE Ⅰ. Afterwards, range [MIN_p^j , MAX_p^j] is discretized into *d* ($d \ge 2$) discrete value with interval dis_p^j =(MAX_p^j - MIN_p^j)/(d-1), i.e., { ${}^1Bound_p^j$,…, ${}^dBound_p^j$ }, where ${}^1Bound_p^j$ = MIN_p^j and ${}^dBound_p^j$ = MAX_p^j . Here, each discrete value ${}^xBound_p^j$ ($1 \le x \le d$) corresponds to the composition segment SEG_p's QoS constraint [0, ${}^xBound_p^j$] over QoS criterion

cr_j. Our final object of this step is to find an appropriate x value for each QoS criterion cr_j ($1 \le j \le m$) of each segment SEG_p ($1 \le p \le \lceil n/k \rceil$).

Next, we utilize the mixed integer programming to determine the best segment constraint [0, ${}^{x}Bound_p^j$] for each QoS criterion cr_j of each composition segment SEG_p. According to the mixed integer programming, this optimization problem could be formalized into the following question:

$$\text{Maximize} \prod_{p=1}^{\lceil n/k \rceil} \prod_{j=1}^{m} Utility({}^{x}Bound_p^j) \quad (1)$$

$$\text{Subject to} \prod_{p=1}^{\lceil n/k \rceil} {}^{x}Bound_p^j \le cst_j \quad (2)$$

$$\frac{1}{\lceil n/k \rceil} \sum_{p=1}^{\lceil n/k \rceil} {}^{x}Bound_p^j \le cst_j \quad (3)$$

$$\sum_{p=1}^{\lceil n/k \rceil} {}^{x}Bound_p^j \le cst_j \quad (4)$$

$$x \in \{1, 2, \ldots, d\} \quad (5)$$

Here, $Utility({}^{x}Bound_p^j)$ in (1) denotes the "goodness" of utilizing [0, ${}^{x}Bound_p^j$] as segment SEG_p's QoS constraint over criterion cr_j, whose computation manner could be found in [5], so we will not discuss it here. The left parts of (2)-(4) respectively denote the aggregated segment QoS constraints, corresponding to the three aggregation types in TABLE Ⅰ, which should not exceed the end user's global QoS constraint. The constraint condition in (5) means that variable x has d possible values. After solving the above optimization problem with mixed integer programming, we can derive the best segment QoS constraints [0, ${}^{x}Bound_p^j$]($1 \le j \le m$) for each composition segment SEG_p ($1 \le p \le \lceil n/k \rceil$).

(3) Step3: For each composition segment, find the QoS-optimal composite solution that satisfies segment QoS constraints.

In Step1, we obtain $\lceil n/k \rceil$ small-scale composition segments { SEG_1, SEG_2,…, $SEG_{\lceil n/k \rceil}$ }, each with k ($1 \le k \le n$) tasks. Through which, we successfully transform a large-scale WSC problem into $\lceil n/k \rceil$ small-scale WSC sub-problems. And in Step2, for each WSC sub-problem, its QoS constraints are also obtained. So in Step3, we only need to solve these QoS constrained WSC sub-problems. Next, we formalize the WSC sub-problem (corresponding to a composition segment) as follows:

$$\text{Maximize } Utility(SEG_p) \quad (6)$$

Subject to

$$\prod_{q=(p-1)*k+1}^{p*k} \left(\sum_{z=1}^{l} ws_q^z.cr_j * x_q^z\right) \le {}^{x}Bound_p^j \quad (7)$$

$$\frac{1}{\lceil n/k \rceil}\left(\sum_{q=(p-1)*k+1}^{p*k} \sum_{z=1}^{l} ws_q^z.cr_j * x_q^z\right) \le {}^{x}Bound_p^j \quad (8)$$

$$\sum_{q=(p-1)*k+1}^{p*k} \sum_{z=1}^{l} ws_q^z.cr_j * x_q^z \le {}^{x}Bound_p^j \quad (9)$$

$$\sum_{z=1}^{l} x_q^z = 1 \quad x \in \{1, 2, \ldots, d\} \quad (10)$$

$$x_q^z \in \{0, 1\} \quad (11)$$

Here, $Utility(SEG_p)$ in (6) denotes the utility value of composite solution for SEG_p. The left parts of (7)-(9) respectively denote the aggregated value for composite solution of segment SEG_p, over QoS criterion cr_j, which should not exceed the calculated segment QoS constraints, according to the three aggregation types in TABLE Ⅰ. The constraint conditions in (10)-(11) mean that for a segment task, only one candidate service would be selected, to form a composite solution. After solving the above mixed integer programming problem, a QoS-optimal composite solution *CompS-p* could be achieved for composition segment SEG_p. And finally, the set {*CompS*-1, …, *CompS*-$\lceil n/k \rceil$ } makes up the QoS-optimal composite solution, for the original WSC problem with large-scale composition structure.

IV. Experiment

In this section, a group of experiments are enacted and executed to validate the effectiveness and efficiency of proposed DCM method, especially in dealing with the WSC problems with large composition structure.

A. *Experiment Configuration*

In this paper, the experiment is based on the updated *QWS Dataset* from Dr. Eyhab Al-Masri [5], which is widely used in the service computing domain. The experiments were deployed and executed on a Lenovo machine with Intel Pentium 2.40 GHz processors and 1.00 GB RAM. The machine is running under Windows XP and C++ integration environment.

B. *Experiment Results and Analyses*

In order to evaluate the feasibility and efficiency of DCM in dealing with WSC problems with large-scale composition structure, the following four test profiles are

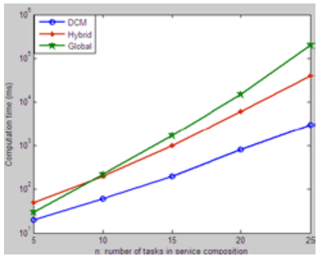

Figure3. Performance comparison w.r.t. number of tasks

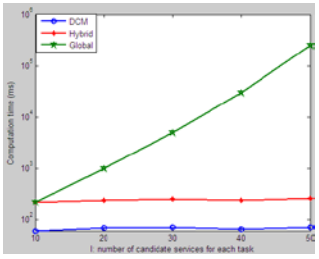

Figure4. Performance comparison w.r.t. number of candidates

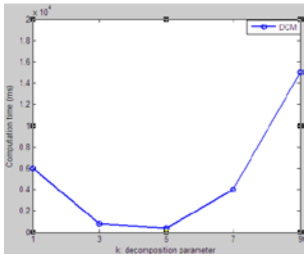

Figure5. Performance comparison w.r.t. parameter *k*

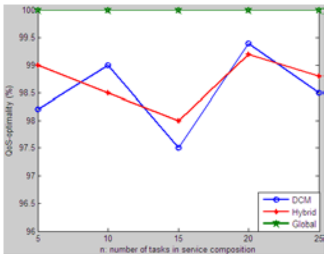

Figure6. QoS optimality w.r.t. number of tasks

designed and compared with another two present methods: *Global* [6] and *Hybrid* [5]. In the experiments, the parameter *d* is equal to 5 for both *Hybrid* and DCM, and five QoS constraints are employed for all test cases.

Profile1: performance comparison of DCM, *Global* and *Hybrid* with different task number (i.e., *n*)

In this profile, the computation time of three methods are tested and compared, for finding a QoS-optimal composite solution. In the experiment setting, the number of tasks in composition structure (i.e., *n*) is varied from 5 to 25. And for each task, there are 10 candidate services, i.e., *l*=10. The parameter *k* in DCM is equal to 3. The experiment comparison results are demonstrated in Fig. 3. As can be seen from Fig. 3, the computation time of DCM outperforms the other two methods when *n* is increased. Especially when the value of *n* is large, the proposed DCM method exhibits its significant advantages, in dealing with WSC problems.

Profile2: performance comparison of DCM, *Global* and *Hybrid* with different candidates (i.e., *l*)

In this profile, the computation time of three methods is compared when the number of candidate services changes. In the experiment setting, the number of tasks in composition structure, i.e., *n*=10. And for each task, the number of candidate services (i.e., *l*), is varied from 10 to 50. The parameter *k* in DCM is equal to 3. The experiment comparison results are demonstrated in Fig. 4. As in Fig. 4, the computation time of *Global* increases significantly when *l* arises. However, DCM and *Hybrid* exhibit similar change trends, whose computation time both stay nearly unchanged with the increase of *l*.

Profile*3: performance comparison of DCM with different values of parameter *k

In our proposed DCM method, we improve the composition efficiency, by transforming a large-scale WSC problem with *n* tasks into $\lceil n/k \rceil$ small-scale WSC sub-problems. Therefore, *k* is an important parameter in DCM. In this profile, we observe the performance change of DCM with the increase of *k*. In the experiment setting, the number of tasks in composition structure, i.e., *n*=20. And for each task, the number of candidate services, i.e., *l* is equal to 10, while the value of parameter *k* is varied from 1 to 9, with interval of 2. The experiment comparison results are demonstrated in Fig. 5. As can be seen from Fig. 5, the computation time of DCM decreases firstly with the arise of the *k*, this is because when *k* grows, the computation workload for achieving segment QoS constraints reduces. While when *k* grows further, the computation workload for finding segment QoS-optimal composite solution increase, so the computation time arises correspondingly.

Profile*4: QoS optimality comparison of DCM, *Global* and *Hybrid

Execution time is an important evaluation item for QoS-aware web service composition, however, the QoS optimality of obtained composite solution should also be considered. In this profile, we test the QoS optimality of three methods. In the experiment setting, the number of candidate services for each task, i.e., *l* is fixed to 10, *n* is varied from 5 to 25, the employed parameter *k* (the task length of each composition segment) is equal to $\lceil n/3 \rceil$. The experiment results are shown in Fig. 6. As in Fig. 6, the *Global* method is the best in terms of QoS optimality of obtained composite solution. And the QoS optimality of DCM is almost the same as that of *Hybrid*, both close to 100%. Therefore, DCM can generate a QoS near-to-optimal composite solution, while ensures better composition efficiency.

V. EVALUATION

In this section, we will analyze the time complexity of DCM. A comparison with related work is also presented, to show the advantages of our proposed DCM method.

A. *Complexity Analysis*

In this subsection, the time complexity of the three steps of DCM will be analyzed separately, after which we obtain the total time complexity of DCM.

(1)Step1: Decomposition of large-scale composition structure into small-scale ones

With the determined *k* value in DCM, we can decompose the large-scale composition structure with n tasks into $\lceil n/k \rceil$ small-scale composition segments, whose time complexity is O(1).

(2)Step2: Determine the segment QoS constraints for all composition segments

A MIP problem is achieved, for determining the segment QoS constraints for each composition segment. Therefore, its time complexity is O($2^{\lceil n/k \rceil * m * d}$), where *n* is the task number, *k* is the decomposition parameter employed in DCM, *m* is the number of QoS constraints, and *d* is the discretization parameter.

(3)Step3: Determine the QoS-optimal composite solution

The QoS-optimal sub-composition solutions of $\lceil n / k \rceil$ composition segments make up the QoS-optimal composite solution, for the original WSC problem with large-scale composition structure. For each composition segment, the time complexity is O(2^{k*m*d}), for determining the QoS-optimal sub-composition solution. Therefore, the time complexity of Step3 is O($\lceil n / k \rceil$ *2^{k*m*d}).

According to the above analysis, a conclusion could be made that the time complexity of our proposed DCM is O($2^{\lceil n/k \rceil * m * d}$ +$\lceil n / k \rceil$ *2^{k*m*d}) = O($2^{\lceil n/k \rceil * m * d}$), as *k* is a fixed value in DCM. Therefore, DCM outperforms *Global* and *Hybrid* methods, whose time complexity are O(2^{n*l}) and O(2^{n*m*d}) respectively, when dealing with WSC problems with large-scale composition structure.

B. Related Work and Comparison Analysis

QoS-aware web service composition has been a hot research topic in the domain [1-3]. In [6], a *Global* method is introduced, where the QoS-aware WSC problem is modeled into a mixed integer programming one, whose time complexity is still large. To reduce the time complexity of WSC, many works focus on minimizing the number of candidate services for each task, such as the *Hybrid* method [5], the *LOEM* method [4], *Skyline* method [7], *MOCACO* method [8] and heuristic method [9]. However, their time complexity is still exponential with the task number, which may lead to much time cost when the composition task number is large. In this paper, we transform the large-scale WSC problem into several small-scale ones, and put forward a decomposition-based composition method DCM. Through experiments, we demonstrate the feasibility and efficiency of DCM in dealing with the WSC problems with large-scale composition structure.

VI. CONCLUSIONS

In this paper, a decomposition-based method, named DCM, has been presented for the WSC problems with large-scale composition structure. Through experiments, we demonstrate the feasibility of DCM. However, there are some shortcomings in the paper; for example, the employed parameter *k* is not assigned a concrete value, which will be studied as a future research topic.

ACKNOWLEDGMENT

This paper is supported by the Open Project of State Key Lab. for Novel Software Technology (No. KFKT2012B31), Natural Science Foundation of Shandong Province of China (No.ZR2012FQ011), SRI of SPED (No.J12LN06), DRF and UF (BSQD20110123, XJ201227) of QFNU.

REFERENCES

[1] Yuhong Yan, Min Chen, and Yubin Yang, “Anytime QoS optimization over the PlanGraph for web service composition,” Proceedings of ACM Symposium on Applied Computing (SAC 12), ACM Press, Mar. 2012, pp. 1968-1975, doi:10.1145/2245276.2232101.

[2] Marisol Garcia Valls, R. Fernández-Castro, Iria Estevez Ayres, Pablo Basanta Val, and Iago Rodriguez, “A Bounded-time Service Composition Algorithm for Distributed Real-time Systems,” Proceedings of IEEE International Conference on High Performance Computing and Communication (HPCC 12), IEEE Press, Jun. 2012, pp. 1413-1420, doi:10.1109/HPCC.2012.207.

[3] Wada Hiroshi, Suzuki Junichi, Yamano Yuji, and Oba Katsuya, “E3: A Multiobjective Optimization Framework for SLA-Aware Service Composition,” IEEE Transactions on Services Computing, vol. 5, Sep. 2012, pp. 358-372, doi: 10.1109/TSC.2011.6.

[4] Lianyong Qi, Ying Tang, Wanchun Dou, and Jinjun Chen, “Combining Local Optimization and Enumeration for QoS-Aware Web Service Composition,” Proceedings of IEEE International Conference on Web Services (ICWS 10), IEEE Press, Jul. 2010, pp. 34-41, doi: 10.1109/ICWS.2010.62.

[5] Mohammad Alrifai and Thomass Risse, “Combining global optimization with local selection for efficient QoS-aware service composition,” Proceedings of 18th International Conference on World WideWeb (WWW 09), ACM Press, Apr. 2009, pp. 881-890, doi: 10.1145/1526709.1526828.

[6] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang, “QoS-aware middleware for web services composition,” IEEE Transactions on Software Engineering, May 2004, vol. 30, pp. 311-327, doi: 10.1109/TSE.2004.11.

[7] Mohammad Alrifai, Dimitrios Skoutas, and Thomas Risse. “Selecting skyline services for QoS-based web service composition,” Proceedings of 19th International Conference on World Wide Web (WWW 10), ACM Press, Apr. 2010, pp. 11-20, doi: 10.1145/1772690.1772693.

[8] Wang Li and He Yan-xiang, “A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO,” Lecture Notes in Computer Science, vol.6082, Nov. 2011, pp. 79 - 86, doi:10.1007/978-3-642-13136-3_22.

[9] Adrian Klein, Fuyuki Ishikawa, and Shinichi Honiden, “Efficient Heuristic Approach with Improved Time Complexity for QoS-Aware Service Composition,” Proceedings of IEEE International Conference on Web Services (ICWS 11), IEEE Press, Jul. 2011, pp. 436-443, doi: 10.1109/ICWS.2011.60