International Journal of Innovative Computing, Information and Control
Volume **12**, Number **6**, December **2016**

ICIC International ©2016 ISSN 1349-4198
pp. 2053–2066

DECOMPOSITION OF GLOBAL CONSTRAINTS FOR QOS-AWARE WEB SERVICE COMPOSITION

Hengzhou Ye[1,2], Taoshen Li[3] and Chao Jing[2]

[1]College of Electrical Engineering
[3]School of Information and Engineering
Guangxi University
No. 100, Daxue Street, Xixiangtang District, Nanning 530004, P. R. China
{ 2002018; jingchao }@glut.edu.cn

[2]College of Information Science and Engineering
Guilin University of Technology
No. 12, Jiangan Street, Qixing District, Guilin 541000, P. R. China
tshli@gxu.edu.cn

Received May 2016; revised September 2016

Abstract. *Web service composition is a key technology required for the implementation of Service-Oriented Architecture (SOA). Since there are many web services with the same functionalities and different Quality of Service (QoS), there is significant interest in QoS-aware web service composition. This paper proposes a novel Global QoS Constraint Decomposition (GCD) approach for QoS-aware web service composition that includes global decomposition and local optimal selection. The GCD model and greedy algorithm have a weak connection with the QoS values of the candidate web services. Hence, the proposed approach can be adopted for dynamic scenarios. Finally, experiments were performed to compare this approach to previous work. The results indicate that our approach shows improved utility and time complexity.*
Keywords: Web service composition, QoS-aware, Global QoS constraints decomposition, Greedy algorithm

1. **Introduction.** Service-Oriented Architecture (SOA) is an emerging paradigm for the development of distributed Internet applications. In an SOA environment, a system is usually described as a composition of web services (WSs) that are loosely coupled, standard-based, and platform-independent [1]. Due to the limitation of individual WS functionalities, it is often necessary to combine a set of WSs to satisfy the complex requirements of users. Additionally, there are WSs that provide the same functionalities but different Quality of Service (QoS), so it is a computational challenge to determine an optimal set of WSs that meet the constraints and preferences of the user, namely QoS-aware Web Service Composition (QSC). Since QoS of WSs can frequently change in a dynamic environment, this problem can be quite challenging. An additional level of complexity is presented because exceptions may occur during the execution of a particular composition scheme, such as a violation of Service Level Agreement or the unavailability of some WSs. To address these potential complications, it is crucial to devise a low-complexity approach to find an optimal replaceable WSs composition.

Most approaches to the QSC problem can be generalized as a local optimization approach or a global optimization approach [2]. The first approach determines an optimal WS for an individual task among a set of candidate WSs. This approach has low time complexity, but it cannot guarantee that the solution meets the global constraints [3]. The second approach is computationally expensive [3,4] because it attempts to select the

optimal composition in order to provide the best utility within the global constraints. Therefore, there is a need to design and implement a low-complexity algorithm that combines the advantages of local and global approaches to meet global constraints.

Here, we propose a strategy of web service composition based on GCD. It is a two stage algorithm: decomposition of global constraints and local selection. During the first stage, the global constraints are decomposed into a set of local constraints that serve as the conservative upper/lower bounds for each task. In this way, the satisfaction of local constraints guarantees the eventual satisfaction of the global constraints. The GCD model and its solution are only weakly associated with the values of QoS criteria for candidate services. When these values are changed, the local constraints obtained by the original process of GCD remain effective. Therefore, when exceptions occur, it is only necessary to repeat the second stage of the process to find an optimal substitution with linear time complexity.

Compared to the previous approaches, the contributions of this work are summarized as follows.

- We proposed an improved model for constraint decomposition with respect to global constraints. This model adopted a more reasonable objective function that allows application to a normal workflow by importing the labelled tree.
- We presented a low-complexity, greedy algorithm with the model of constraints decomposition in accordance with global constraints. The algorithm has few connections to the properties of QoS in candidate services allowing separation of the process of decomposition of global constraints and local selection.

The remainder of this paper is structured as follows. Related work is presented and discussed in Section 2. Section 3 formulates the problem and statement. Section 4 details the proposed algorithm. Section 5 describes the performance and evaluations. Finally, we present our conclusions.

2. **Related Work.** Generally, the QSC problem is an NP-hard problem that is equivalent to the Multidimensional Multiple choice Knapsack Problem (MMKP) [3,4]. There have been many attempts to solve this problem. These approaches can be classified as ones for which the predefined workflow is known and ones for which the predefined workflow is unknown, referring to WS discovery and semantic reconciliation [5,6]. In our work, we focus on the first approach.

In [2], two approaches are described and compared: local selection and Integer Programming (IP) based global selection. For the local selection, the chosen service is based on local optimum, which becomes sub-optimal in the global view. This approach cannot address the problem of global constraints. Some greedy strategy-based methods [3,7] have similar features. The IP-based selection approach gains better performance without violation of global constraints. However, the time complexity becomes exponential, especially for a dynamic scenario. Gabrel et al. [4] converted the problem of QSC into a Mixed Integer Linear Programming (MILP) problem, and the number of variables and constraints are polynomial. However, the solution is time-consuming, particularly under dynamic conditions. This strategy can only be used for linear objective functions.

Intelligent algorithms have been widely adopted to solve the problems of QSC. In [8], a Genetic Programming (GP)-based algorithm was proposed to find an approximate optimal service composition solution in a distributed service environment. In this scenario, the performance of an atomic WS is determined for the users' location. The authors in [9] employed a penalty-based Genetic Algorithm (GA) to address dependency and conflict among WSs. Researchers in [10] devised a GA that takes the execution time, price, transactional property, stability, and penalty-factor into account when searching for the

best solution. In [11], they presented a composition model that includes consideration of the QoS and features in a Cloud network and used GA to solve the problem. To improve the efficiency and speed of convergence, some GA-based algorithms are used as a population diversity measurement [12] or an adaptive crossover strategy [13]. Additionally, Simulated Annealing (SA) [14], Harmony Search (HS) [15], Hill-Climbing (HC) [16], and particle swarm optimization [17-19] strategies have been adopted to address these problems. Although these algorithms can find optimal solutions, they have poor time complexity and stability, and if service failure occurs during execution, they are not able to quickly find a good replacement.

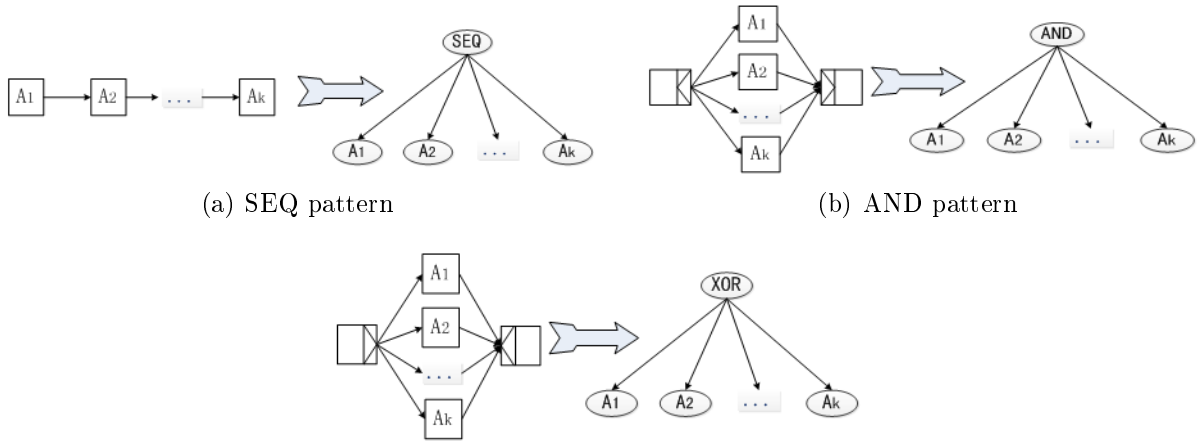
For the QSC problem, a global constraint decomposition-based method can overcome these shortcomings. In [20], based upon user preferences, the global QoS constraint was broken down into local constraints. Also, incorporating fuzzy logic, an adaptive particle swarm optimization algorithm can be exploited to select the best quality ruler. The process of decomposition relied upon an empirical utility function rather than astrict mathematic model. In [21], a Quality Constraint Decomposition (QCD) model was established to support the generic workflow. However, these approaches cannot satisfy global constraints. Since the optimization of the global utility of a composite service requires optimizing the local utilities of all tasks, and the global constraints can be verified by analysis of the local constraints for each task, [22] utilized a decomposition-based algorithm. However, this algorithm cannot leverage local selection alone to quickly find an alternative solution after an exception occurs in a dynamic scenario. In [23], a methodology consisting of Constraint Decomposition Phase and Service Selection Phase is proposed to select the best available service combination for a given workflow. A new and unique method based on local selection and a new aggregation function which can assure 100% guarantee for successful execution of each alternate path of an OR pattern are its main contributions. However, the proposal is only suitable for decomposing constraints to a simple combinational workflow.

There are two previous publications that are closely related to our work [24] and [25]. In [24], they analyzed the shortcomings of literature [25] and detailed an approach that meets the global constraint. However, this model still has several issues. First, the GCD model's objective function must be improved. GCD balances the need to provide each task with as much services as possible to satisfy local constraints with the need for an even distribution of the number of services for each task. By meeting both these objectives, GCD can maximize the WS compositions. Second, the complexity of the Culture Genetic Algorithm (CGA) [22] is proportional to the number of tasks, the number of services, and the number of QoS criteria and therefore performs more slowly with increased problem complexity. An approximate algorithm for GCD model that can rapidly find a proper solution would be highly useful. Third, in the previous works, they have focused on only a sequential pattern. However, it is essential to also allow patterns of parallel and exclusive choice in the workflow. By adding these two patterns into the model, we present an enhanced global QoS constraint decomposition model, and devise a greedy algorithm to increase the speed of the GCD model.

3. **Problem Formulation and Statement.**

3.1. **Workflow model.**

A workflow is used to describe the process of assessing and utilizing the functionalities of different WSs to satisfy the users. For a given workflow, a task has a set of WSs that share functionality and a pattern (e.g., sequence, parallel, exclusive choice, or loop) that represents the temporal relationship between different tasks [4]. Since the loop pattern can be converted into a sequential pattern [27,28], we therefore



(a) SEQ pattern (b) AND pattern

(c) XOR pattern

FIGURE 1. Illustration of the patterns and resulting labelled tree

consider the following three patterns: sequence (SEQ), parallel (AND), and exclusive choice (XOR).

There are alternate methods that can be used to describe a workflow, such as Petri Net [26], labelled tree [21], or numbered graph [4]. Here, we use the labelled tree method to describe the workflow. Assume that a workflow includes only one of the patterns (SEQ, AND, or XOR); it can be labelled as a 2-layer labelled tree. Each leaf node is labelled to represent a task and the root is labelled to represent a pattern, as shown in Figure 1.

A general workflow consists of the above three patterns that are recursively concatenated and interlaced. This can be represented as a labelled tree with the following features:

(1) The number of leaf nodes is equal to the number of tasks in the workflow, and each leaf node corresponds to a task;
(2) Each internal node represents a pattern. The number of internal nodes is less than the number of leaf nodes. Thus, the total number of nodes is less than $2 * n$, where n is the number of tasks;
(3) Each internal node has at least two children. Meanwhile, each node has a sole parent, except for the root node.

Figure 2 illustrates the transformation of a workflow into a labelled tree. This transformation is performed from the inside to the outside. First, A2, A3, and A4 in an SEQ pattern can be transformed to a sub-tree with the root of SEQ (labelled as ST2) and A6, A7, and A8 forming an XOR pattern that can be described as a sub-tree with the root of XOR (labelled as ST6). Then, A5, ST6, and A9 are used to construct an SEQ

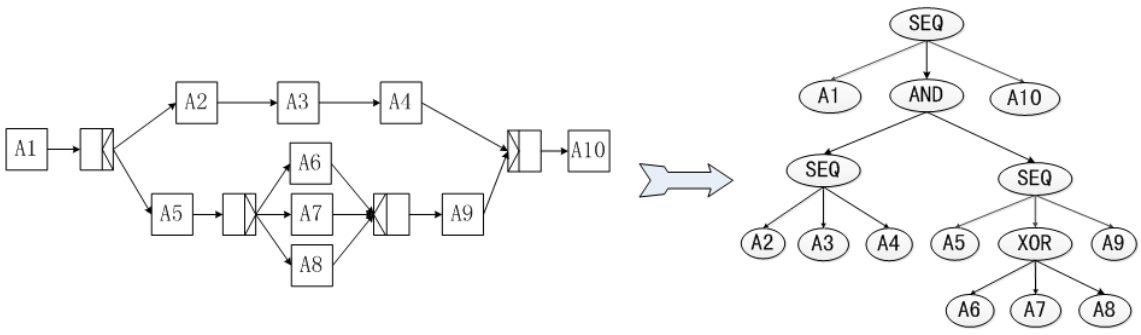


FIGURE 2. Illustration of generating a labelled tree based on a workflow

pattern which corresponds to an SEQ sub-tree (labelled as ST5). Next, ST2 and ST5 can be transformed to an AND sub-tree (labelled as ST25). Finally, transforming A1, ST25, and A10 completes the transformation and results in a labelled tree.

3.2. **Modeling of QoS criteria.** For each task in the workflow, there are a set of WSs. These WSs must be evaluated based on their QoS criteria. Previous studies [26,29-31] have shown that QoS criteria can include execution price, response, availability, throughput, reliability, reputation, and others. As mentioned in [4], depending on the way the composite WS performance is calculated, there are three kinds of QoS criteria: sum-type (e.g., price), min/max-type (e.g., response time), and product-type (e.g., reliability). The aggregating formulas for each workflow pattern are presented in Table 1.

TABLE 1. Quality-aggregated formulas with different patterns

Aggregation type	**Example**	**SEQ**	**AND**	**XOR**
sum-type	price	$\sum_{i=1}^{n} q(x_i)$	$\sum_{i=1}^{n} q(x_i)$	$\frac{1}{n}\sum_{i=1}^{n} q(x_i)$
min/max-type	response time	$\sum_{i=1}^{n} q(x_i)$	$\max\{q(x_1), \ldots, q(x_n)\}$	$\min\{q(x_1), \ldots, q(x_n)\}$
product-type	reliability	$\prod_{i=1}^{n} q(x_i)$	$\prod_{i=1}^{n} q(x_i)$	$\frac{1}{n}\sum_{i=1}^{n} q(x_i)$

In Table 1, x_i, $\forall i \in [1, n]$ represents the ith component in combined service $X = (x_1, x_2, \ldots, x_n)$, and $q(x_i)$ is the quality value of x_i.

The QoS criteria can be positive or negative, and positive attributes can be easily transformed into negative attributes by multiplying by -1. For the sake of simplicity, here we only address negative criteria [25].

3.3. **Problem statement.** There are n tasks $T = \{T_1, T_2, \ldots, T_n\}$ and k constraints $c = \{c_1, c_2, \ldots, c_k\}$ in the QSC problem. Each task T_i, $i \in [1, n]$ has m number of candidate WSs $s_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$. The objective of the GCD process is to decompose each global QoS constraint c_i into a set of local constraints c_{r1}, c_{r2}, $\ldots$, c_{rn}, where n is the number of tasks in the workflow. These local constraints should ensure that if candidate services for each task are able to satisfy the local QoS constraints, the combined WS by these candidate services can meet the global QoS constraints.

To improve performance, GCD attempts to maximize the number of candidate WSs that satisfy the local constraints. It can be defined as Equation (1):

$$F(t) = \prod_{i=1}^{n} num(T_i) \tag{1}$$

where $num(T_i)$ denotes the number of candidate WSs for T_i ($i \in [1, n]$) that meets the local constraints. The value of $num(T_i)$ can be calculated using Equation (2):

$$num(T_i) = \#\{s | s \in s_i \wedge q(s, r) \leq c_{ri}\} \tag{2}$$

where $\#S$ denotes the number of elements in S and $q(s, r)$ denotes the rth QoS criterion for service s.

Hence, the problem of GCD can be formulated as an optimization problem as follows:

$$\text{Max } F(T)$$
$$f(rt, r) \leq c_r, \quad r = 1, 2, \ldots, k$$

$$\min\left(q_{ij}^r\right) \le c_{ri} \le \max\left(q_{ij}^r\right), \quad i = 1, 2, \ldots, n, \quad r = 1, 2, \ldots, k$$

where c_{ri} $(r = 1, 2, \ldots, k;\ i = 1, 2, \ldots, n)$ are decision variables; rt is the root node of labelled tree for the given workflow; $f(rt, r)$ represents the upper bound of the rth QoS criterion for rt; q_i^r denotes the rth QoS criterion for candidate service s_{ij}; and $\min(q_i^r)$ and $\max(q_i^r)$ stand for the minimum and maximum of the rth QoS criterion for T_i, respectively.

4. **Algorithm Design.** To resolve the problem of QSC using global constraint decomposition, the proposed algorithm is divided into two steps: a GCD process and local optimal selection. QSC is a combinatorial optimization problem. These types of problems can often be solved using intelligence algorithms [9-20], but these often have a high degree of complexity and instability. Additionally, because the objective function Equation (1) is strongly associated with the QoS values of candidate WSs, these algorithms cannot be used in a dynamic scenario. As an alternative, we propose a greedy algorithm as part of the process of GCD to reduce the time complexity and allow low coupling to the process of local optimal selection.

4.1. **Constraint conditions for GCD.** For each global QoS constraint, there is a corresponding constraint condition in the GCD model. We need to formulate the $f(rt, r)$ using the local constraints c_{ri} $\{r = 1, 2, \ldots, k,\ i = 1, 2, \ldots, n\}$.

For any node v in labelled tree, when the rth QoS criterion is sum-type (e.g., price), the upper bound of the rth QoS criterion for v, denoted as $f(v, r)$, can be calculated using Equation (3):

$$f(v,r) = \begin{cases} c_{r,v.task} & type(v) == LEAF \\ \sum_{cv \in v.child} f(cv, r) & type(v) == \text{SEQ or AND} \\ \dfrac{1}{|v.child|} \sum_{cv \in v.child} f(cv, r) & type(v) == \text{XOR} \end{cases} \tag{3}$$

where $v.task$ represents the task that is associated with the node v, if and only if v is a leaf node; $v.child$ denotes the set of v's directed children nodes; $type(v)$ is the type of v.

Similarly, when the rth QoS criterion is min/max-type (e.g., response time), $f(v, r)$ can be determined by Equation (4):

$$f(v,r) = \begin{cases} c_{r,v.task} & type(v) == LEAF \\ \sum_{cv \in v.child} f(cv, r) & type(v) == \text{SEQ} \\ \max\limits_{cv \in v.child} f(cv, r) & type(v) == \text{AND} \\ \min\limits_{cv \in v.child} f(cv, r) & type(v) == \text{XOR} \end{cases} \tag{4}$$

When the rth QoS criterion is product-type (e.g., reliability), then the $f(v, r)$ can be determined by Equation (5):

$$f(v,r) = \begin{cases} c_{r,v.task} & type(v) == LEAF \\ \prod_{cv \in v.child} f(cv, r) & type(v) == \text{SEQ or AND} \\ \dfrac{1}{|v.child|} \sum_{cv \in v.child} f(cv, r) & type(v) == \text{XOR} \end{cases} \tag{5}$$

Using Equations (3) to (5) as appropriate, the $f(rt, r)$ is calculated via recursion. Thus, all the nodes in the labelled tree will be visited only one time. Because the number of nodes is less than $2 * n$, the time complexity is bounded by $O(n)$.

4.2. **Greedy algorithm for GCD.** To include as many possible feasible composition schemes, a greedy algorithm is used according to the following criteria: (1) the local constraints are relaxed as much as possible; (2) for each task, the number of WSs meeting the local constraint should be uniformly distributed. Here, we use a greedy algorithm to initially search for a solution that meets all the constraints of the GCD model. Next, the initial solution is adjusted in three phases. First, for the fairness of all tasks, all QoS criteria for each task will be simultaneously and repeatedly incremented a step-length

Algorithm 1: Find a suitable QCD solution

input:
levelNum: the number of quality level;
q_{ij}^r: the value of QoS attributes the rth attribute of s_{ij};

output:
Solution[k][n]: solution[r][i] represents the local constraint for the rth QoS attribute of the ith task, where $\forall r \in [1,k]$, $\forall i \in [1,n]$

Step 1: set array delta, where delta[r][i] represents the step length for the rth QoS attribute of the ith task.
Step 2: Init solution[r][i] = $\min\left(q_{ij}^r\right)$;
Init delta[r][i] = $\left(\max\left(q_{ij}^r\right) - \min\left(q_{ij}^r\right)\right)$ /**levelNum**;
Step 3: **if**(!**isFit**(solution)) then
Output("cannot find suitable solution");
End if
Step 4: **while** (**isFit**(solution)) do
Solution[r][i] += delta[r][i];
End while
Solution[r][i] −= delta[r][i];
Step 5: changed = true;
While (changed) do
For each task i do
Solution[r][i] += delta[r][i];
If (**isFit**(solution)) then
Changed = true;
Else
Solution[r][i] −= delta[r][i];
End if
End for
End while
Step 6: changed = true;
While (changed) do
For each task i and each attribute k do
Solution[r][i] += delta[r][i];
If (**isFit**(solution)) then
Changed = true;
Else
Solution[r][i] −= delta[r][i];
End if
End for
End while
Step 7: **Output** solution;

within the global constraint. The step length is determined by the number of quality level **levelNum**, and is calculated as:

$$\Delta_i^k = \left(\max\left(q_{ij}^r\right) - \min\left(q_{ij}^r\right)\right) / \mathbf{levelNum}$$

where $\max\left(q_{ij}^r\right)$ and $\min\left(q_{ij}^r\right)$ define the maximum and minimum possible value of the rth criterion for task T_i respectively. The second step is the tuning of all QoS criterion single tasks. The last step is adjustment of the single QoS criterion for each single task. The purpose of these last two steps is to maintain as many candidate WSs as possible. The main workflow of this algorithm is detailed as Algorithm 1.

In Algorithm 1, the function **isFit** is used to decide whether a potential solution satisfies the global QoS constraints and determines the time complexity for Algorithm 1. We suppose that the number of loops for Step 4 to Step 6 is n_1, n_2 and n_3 ($n_1 + n_2 + n_3 \leq$ **levelNum**), respectively. The number of the function **isFit** carried out is:

$$n_1 + n * n_2 + n * k * n_3 \leq n * k * \mathbf{levelNum}$$

where n and k represent the number of tasks and the number of QoS criteria that meet the global constraints, respectively. Thus, the time complexity of Algorithm 1 is less than $O(n * k * \mathbf{levelNum})$.

4.3. **Local optimal selection.** Previous reports [30-34] discussed how to determine the QoS value of a WS. As mentioned in [23], it is also important to consider commercial agreements or historical contact information that exist between the current candidate WS and its upstream service as part of the QoS value assessment for the candidate WSs. The GCD process is connected with a workflow and QoS model, but it is independent from the number and the QoS values of the WSs. Therefore, the QoS values of the WSs can be determined by local optimal selection as detailed in Algorithm 2.

Algorithm 2: Select the best end-to-end composite service using a local selection strategy

input: the local constraints for each tasks determined by GCD
output: a labelled tree representing end-to-end composite service

Step 1: **For** each task do
Set utility = -1, selected WS = null;
For each ws associated with task T do
determine the QoS values of ws according to the commercial agreements, the historical contact information, or from its description file;
If (ws satisfies the local constraints for t) and (U(ws)>utility) then
Utility = U(ws);
selected WS = ws;
end if
end for
if (selected WS == null) Output "No available solution";
end for
Step 2: for each leaf node in LT, let the selected WS represent the corresponding task;
for each internal node in LT, calculate its utility;
Setp 3: **for** each node in LT with a depth-first traversal do
If ($type(v)$ == XOR) clip all the children except the child with best utility;
End for
Output LT;

In Algorithm 2, after determining the WSs' QoS values, the utilities of the WSs that satisfy the local constraints obtained by the GCD process are then evaluated using Equation (6).

$$U(s) = \sum_{r=1}^{k} \frac{\max(q_i^r, s) - q(r, s)}{\max(q_i^r, s) - \min(q_i^r, s)} \cdot w_r \tag{6}$$

where $\max(q_i^r, s)$ is the maximum value and $\min(q_i^r, s)$ is the minimum value that can be expected for the rth QoS criterion of the task T_i; $q(r, s)$ denotes the value of the rth criterion of WSs; $w_r \in (0, 1)$ and $\sum_{r=1}^{k} w_r = 1$. For each task, the WS with the best utility is selected (Step 1).

Note that there may be XOR patterns and several end-to-end composited services within the workflow and the optimal solution should be selected. In Step 2, for each internal node in the labelled tree, the utility is calculated as described in [21]. In Step 3, with a depth-first traversal, for each XOR node in the labelled tree, the child with the best utility is retained.

The time complexity of Step 1 is $O(nm)$, and the time complexity of Step 2 and Step 3 are $O(n)$, where n and m represent the number of tasks and the number of WSs for each task, respectively. The time complexity of Algorithm 2 is bounded by $O(nm)$.

5. **Experiments.** The experiments were conducted on a Lenovo PC with Intel (R) Core (TM) i5-2430M, with a 2.4 GHz Intel Xeon dual-core processor and 4 GB RAM, under Windows 7 and Java 7. The workflows were randomly generated with SEQ, XOR, and AND patterns by varying the number of tasks. For the number of AND and XOR patterns, each accounts for one quarter. As in reference [4], the goal was to minimize the cost (execution price) and satisfy the two QoS constraints of execution (response) time and reliability. The WS registry was generated as done previously [4]. For a given number of tasks and a set number of WSs per task, 20 instances were randomly generated in order to compute average value.

5.1. **Utility and time complexity.** We evaluated the utility and time complexity of our approach (GS_GCD) compared to MILP [4] and CGA_QCD [21]. The value of **levelNum** was set to 20; the number of initial individuals was set to 200; the iteration time was set to 200; and other parameters were assigned the same values as used by [21].

As shown in Figure 3, we used the MILP results as a reference. When the number of WSs was set to 300 and the number of tasks ranged from 40 to 200, the execution price of

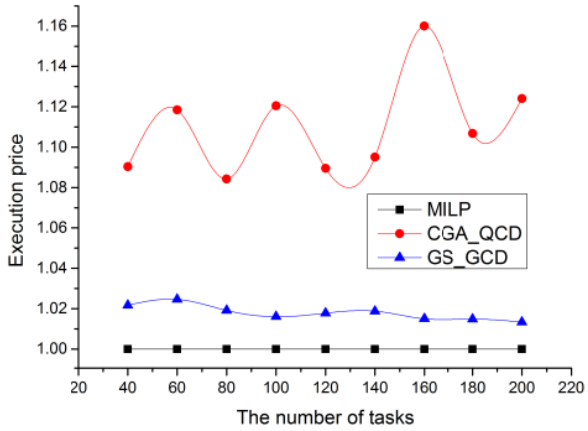


FIGURE 3. Execution price for the three models for different numbers of tasks

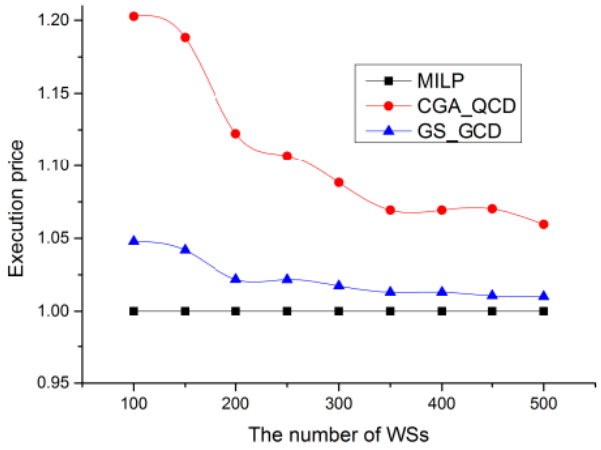


FIGURE 4. Execution price for the three models for different numbers of WSs

GS_GCD and CGA_QCD approaches increased by 2% and 10%, respectively. In Figure 4, when the number of tasks was set to 100 and execution price over a range of WSs was evaluated, we found that the execution price of GS_GCD increased 5% at lower WSs but decreased as the number of WSs increased. From Figure 3 and Figure 4, we can conclude that the utility of GS_GCD was slightly lower than that of MILP, but it significantly outperformed the CGA_QCD model.

The results in Figure 5 and Figure 6 show that the time complexity of GS_GCD is less than MILP and CGA_QCD for the number of tasks and the number of WSs. The results also show that the growth of WSs has little impact on the time for the GS_GCD model but results in strong increases in computation time for the MILP and CGA_QCD models. This is because the complexity of GS_GCD is dominated by the GCD process which is largely independent of the number of WSs. However, the scale of the MILP model is approximately linear with the number of the WSs. For the CGA_QCD model, although the QCD component is independent of the number of WSs, the CGA algorithm required to solve this model is closely related resulting in a strong dependence overall.

In summary, for both utility and time complexity, our results indicate that the GS_GCD model significantly outperforms the CGA_GCD model. Although the performance of the GS-GCD model is similar to that of MILP, we propose the GS_GCD is superior to

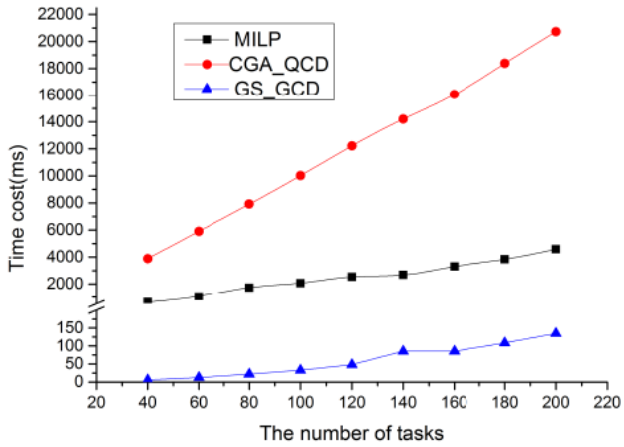


FIGURE 5. Computational time requirements for the three models for different numbers of tasks

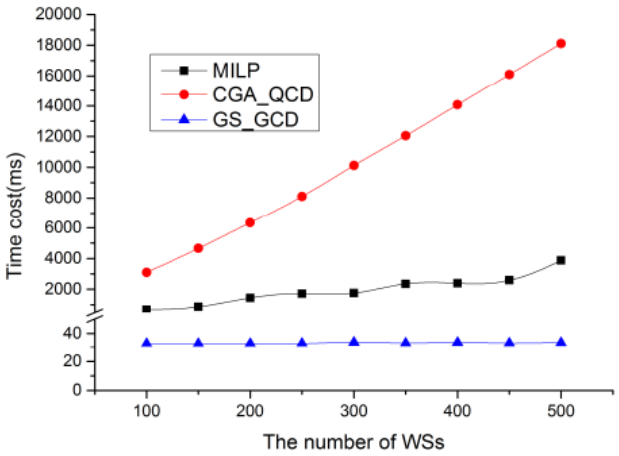


FIGURE 6. Computational time requirements for the three models for different numbers of WSs

MILP for the following reasons: (1) the objective function of MILP must be linear; (2) MILP requires a much higher time complexity than GS_GCD; and (3) once service failure occurs during execution, it is difficult to find an optimal replaceable service that can still guarantee the global QoS constraints.

5.2. **The impact of levelNum.** This experiment was designed to determine the influence of **levelNum** on our model. In this experiment, the value of **levelNum** ranged from 5 to 40.

In terms of execution price, according to Figure 7 and Figure 8, when the value of **levelNum** is equal or greater than 10, the value of **levelNum** has a slight influence. As the number of tasks or WSs increased, there was less of an effect.

We next determined the costs for different values of **levelNum**. As shown in Figure 9 and Figure 10, the time cost linearly increased as the value of **levelNum** increased.

The experimental results show that the proposed method is not sensitive to the value of **levelNum** only if **levelNum** ≥ 10. Thus, the model has good stability.

6. **Conclusions.** This paper proposes a global QoS constraint decomposition-based approach to analyze workflow-based QSC. To decompose the global QoS constraints, a set of local constraints for each task was first obtained. Then, using these local constraints,

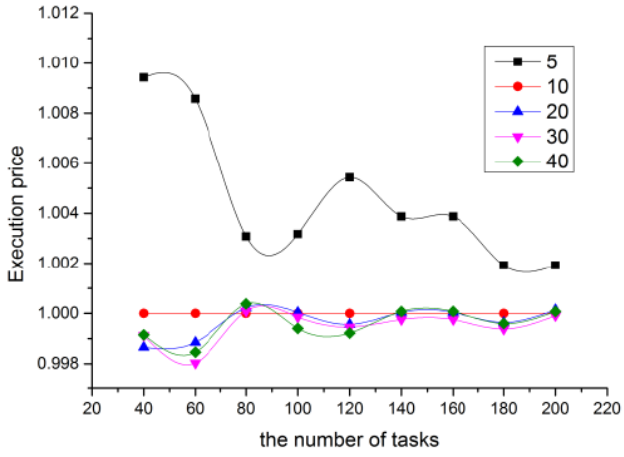


FIGURE 7. Execution price in terms of different values of **levelNum** (5, 10, 20, 30, or 40) over a range of number of tasks

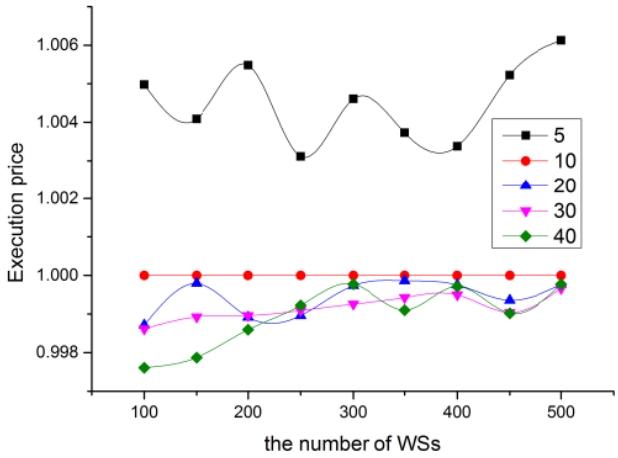


FIGURE 8. Execution price in terms of different values of **levelNum** (5, 10, 20, 30, or 40) over a range of number of WSs

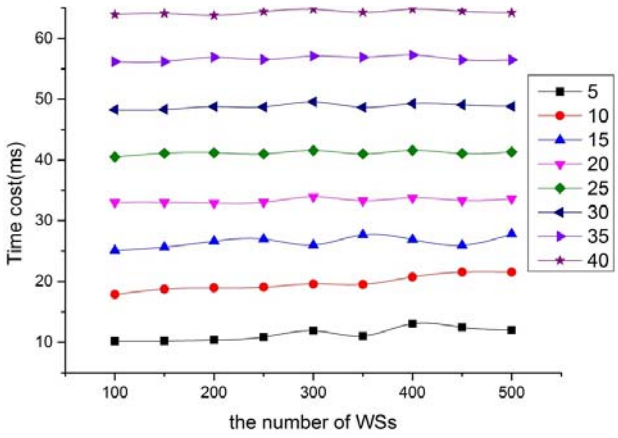


FIGURE 9. Time cost in terms of different values of **levelNum** (5, 10, 15, 20, 25, 30, 35, or 40) over a range of number of WSs

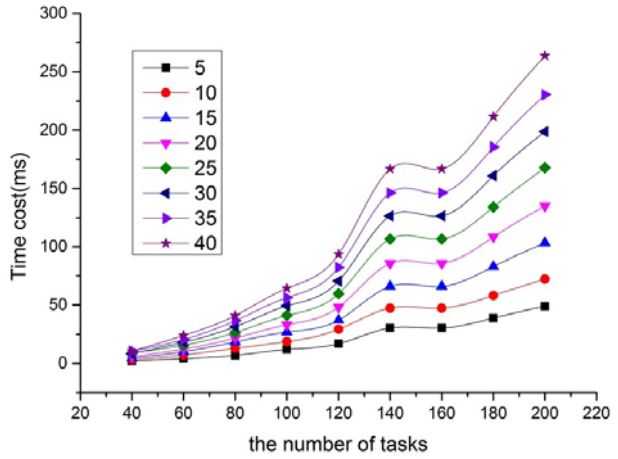


FIGURE 10. Time cost in terms of different values of **levelNum** (5, 10, 15, 20, 25, 30, 35, or 40) and the number of tasks

a local optimal selection process was carried out to ensure the suitability of the global constraints. In this proposed approach, the GCD model and solution method are nearly

independent from the WSs. This ensures that the local constraints obtained by the original GCD process are valid even if the QoS values of WSs change. Therefore, when an exception occurs during running time, an appropriate replacement or composition service can be quickly found via a local optimization selecting process. Finally, the experimental results demonstrated that this approach outperforms current approaches.

This paper does not consider user preferences for constraints, and is not able to include constraints that are expressed in uncertain form. Expansion of the model to allow this flexibility is the goal of future studies.

Acknowledgment. This work was supported by the Natural Science Foundation of Guangxi (No. 2014GXNSFBA118269, No. 2015GXNSFBA139260) and the National Natural Science Foundation of China (No. 51365010).

REFERENCES

[1] M. P. Papazoglou and W. J. Heuvel, Service oriented architectures: Approaches, technologies and research issues, *The VLDB Journal*, vol.16, no.3, pp.389-415, 2007.

[2] L. Z. Zeng, B. Benatallah, M. Dumaset et al., QoS-aware middleware for web services composition, *IEEE Trans. Software Engineering*, vol.30, no.5, pp.311-327, 2004.

[3] T. Yu, Y. Zhang and K. J. Lin, Efficient algorithms for web services selection with end-to-end QoS constraints, *ACM Trans. the Web*, vol.1, pp.1-26, 2007.

[4] V. Gabrel, M. Manouvrier and C. Murat, Web services composition: Complexity and models, *Discrete Applied Mathematics*, vol.196, no.1, pp.100-114, 2015.

[5] G. B. Zou, Q. Lv, Y. X. Chen et al., QoS-aware dynamic composition of web services using numerical temporal planning, *IEEE Trans. Services Computing*, vol.7, no.1, pp.18-31, 2014.

[6] P. W. Wang, Z. J. Ding, C. J. Jiang et al., Constraint-aware approach to web service composition, *IEEE Trans. Systems, Man, and Cybernetics: System*, vol.44, no.6, pp.770-784, 2014.

[7] Y. S. Luo, Y. Qi, D. Hou et al., A novel heuristic algorithm for QoS-aware end-to-end service composition, *Computer Communications*, vol.34, no.9, pp.1137-1144, 2011.

[8] Y. Yu, H. Ma and M. J. Zhang, A genetic programming approach to distributed QoS-aware web service composition, *IEEE Congress on Evolutionary Computation*, pp.1840-1846, 2014.

[9] L. F. Ai and M. L. Tang, A penalty-based genetic algorithm for QoS-aware web service composition with inter-service dependencies and conflicts, *International Conference on Computational Intelligence for Modelling, Control and Automation*, Vienna, Austria, pp.547-562, 2008.

[10] Z. J. Ding, J. J. Liu, Y. Q. Sun et al., A transaction and QoS-aware service selection approach based on genetic algorithm, *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol.45, no.7, pp.1035-1046, 2015.

[11] D. D. Wang, Y. Yang and Z. Q. Mi, A genetic-based approach to web service composition in geo-distributed cloud environment, *Computers and Electrical Engineering*, vol.43, pp.129-141, 2015.

[12] C. W. Zhang, Adaptive genetic algorithm for QoS-aware service selection, *IEEE Workshops of International Conference on Advanced Information Networking and Applications*, pp.273-277, 2011.

[13] Y. Yu, H. Ma and M. J. Zhang, An adaptive genetic programming approach to QoS-aware web services composition, *IEEE Congress on Evolutionary Computation*, pp.1740-1747, 2013.

[14] A. E. Yilmaz and P. Karagoz, Improved genetic algorithm based approach for QoS aware web service composition, *IEEE International Conference on Web Services*, pp.463-470, 2014.

[15] N. Jafarpour and M. R. Khayyambashi, A new approach for QoS-aware web service composition based on harmony search algorithm, *Proc. of the 11th IEEE International Symposium on Web Systems Evolution*, pp.75-78, 2009.

[16] A. Klein, F. Ishikawa and S. Honiden, Efficient heuristic approach with improved time complexity for QoS-aware service composition, *IEEE International Conference on Web Services*, pp.436-443, 2011.

[17] H. Liu, F. R. Zhong, B. Ouyang et al., An approach for QoS-aware web service composition based on improved genetic algorithm, *International Conference on Web Information Systems and Mining*, vol.1, pp.123-128, 2010.

[18] J. X. Liao, Y. Liu, J. Y. Wang et al., Service composition based on niching particle swarm optimization in service overlay networks, *KSII Trans. Internet and Information Systems*, vol.6, no.4, pp.1106-1127, 2012.

[19] S. G. Wang, Q. B. Sun, H. Zou et al., Particle swarm optimization with skyline operator for fast cloud-based web service composition, *Mobile Networks & Applications*, vol.18, no.1, pp.116-121, 2013.

[20] S. G. Wang, Q. B. Sun and F. C. Yang, Web service dynamic selection by the decomposition of global QoS constraints, *Journal of Software*, vol.22, no.7, pp.1426-1439, 2011.

[21] F. Mardukhi, N. NematBakhsh, K. Zamanifar et al., QoS decomposition for service composition using genetic algorithm, *Applied Soft Computing*, vol.6, no.5, pp.3409-3421, 2013.

[22] S. X. Sun and J. Zhao, A decomposition-based approach for service composition with global QoS guarantees, *Information Sciences*, vol.199, pp.138-153, 2012.

[23] C. Surianarayanan, G. Ganapathy and M. S. Ramasamy, An approach for selecting best available services through a new method of decomposing QoS constraints, *Serv. Oriented Comput. Appl.*, vol.9, no.2, pp.107-138, 2015.

[24] Z. Z. Liu, X. Xue, J. Q. Shen et al., Web service dynamic composition based on decomposition of global QoS constraints, *Int. J. Adv. Manuf. Technol.*, vol.69, pp.2247-2260, 2013.

[25] M. Alrifai and T. Risse, Combining global optimization with local selection for efficient QoS-aware service composition, *Proc. of the 18th International Conference on World Wide Web*, pp.881-890, 2009.

[26] Y. H. Du, W. Tan and M. C. Zhou, Timed compatibility analysis of Web service composition: A modular approach based on Petri nets, *IEEE Trans. Automation Science and Engineering*, vol.11, no.2, pp.594-606, 2014.

[27] D. Ardagna and B. Pernici, Adaptive service composition in flexible processes, *IEEE Trans. Software Engineering*, vol.33, no.6, pp.369-384, 2007.

[28] Y. L. Liu, L. Wu and S. J. Liu, A novel QoS-aware service composition approach based on path decomposition, *IEEE Asia-Pacific Services Computing Conference*, pp.76-82, 2012.

[29] E. Al-Masri and Q. H. Mahmoud, QoS-based discovery and ranking of web services, *The 16th IEEE International Conference on Computer Communications and Networks*, pp.529-534, 2007.

[30] Y. L. Zhang, Z. B. Zheng and M. R. Lyu, WSExpress: A QoS-aware search engine for web services, *Proc. of the IEEE International Conference on Web Services*, pp.91-98, 2010.

[31] M. Almulla, H. Yahyaoui and K. Al-Matori, A new fuzzy hybrid technique for ranking real world web services, *Knowledge-Based Systems*, vol.77, pp.1-15, 2015.

[32] Z. B. Zheng, Y. L. Zhang and M. R. Lyu, Distributed QoS evaluation for real-world web services, *IEEE International Conference on Web Services*, pp.83-90, 2010.

[33] M. Rajeswari, G. Sambasivam, N. Balaji et al., Appraisal and analysis on various web service composition approaches based on QoS factors, *Journal of King Saud University – Computer and Information Sciences*, vol.26, pp.143-152, 2014.

[34] Z. B. Zheng, Y. L. Zhang and M. R. Lyu, Investigating QoS of real-world web services, *IEEE Trans. Services Computing*, vol.7, no.1, pp.32-39, 2014.