A QoS-aware composition method supporting cross-platform service invocation in cloud environment

Lianyong Qi, Wanchun Dou, Xuyun Zhang, Jinjun Chen

PII: S0022-0000(11)00161-9
Reference: YJCSS 2572

To appear in: Journal of Computer and System Sciences

Received date: 22 August 2010
Revised date: 20 May 2011
Accepted date: 8 December 2011


This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
A QoS-aware composition method supporting cross-platform service invocation in cloud environment

Lianyong Qi\textsuperscript{1,2}, Wanchun Dou\textsuperscript{1,2,*}, Xuyun Zhang\textsuperscript{3}, Jinjun Chen\textsuperscript{4}

\textsuperscript{1} State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
\textsuperscript{2} Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China
\textsuperscript{3} Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
\textsuperscript{4} School of Systems, Management and Leadership, University of Technology Sydney, Australia

Highlights

▶ We model service composition as a constraint satisfaction problem. ▶ Little time is needed for determining whether a composition problem has a solution. ▶ Decreasing quality precision will improve service composition efficiency.
A QoS-Aware Composition Method Supporting Cross-Platform Service Invocation in Cloud Environment

Lianyong Qi¹², Wanchun Dou¹²,*, Xuyun Zhang³ and Jinjun Chen⁴

1. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
2. Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China
3. Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
4. School of Systems, Management and Leadership, University of Technology Sydney, Australia

Abstract: With the increasing popularity of cloud computing technologies, more and more service composition processes are enacted and executed in cloud environment. Compared with the various and approximately infinite application requirements from end users, the web services held by a cloud platform is usually limited. Therefore, it is often a challenging effort to develop a service composition, in such a situation that only part of the functional qualified candidate services could be found inside a cloud platform. In this situation, the absent services will be invoked in a cross-platform way outside the cloud platform. In view of this challenge, a QoS-aware composition method is investigated for supporting cross-platform service invocation in cloud environment. Furthermore, some experiments are deployed to evaluate the method presented in this paper.

Keywords: service composition, cloud environment, QoS, cross-platform service invocation

1. Introduction

In the past decades, web service technique has brought unprecedented opportunities for organizations to establish more agile and versatile collaborations with each other [1]. From the perspective of realization, web service is often encapsulated with application functionality and information resources, and accessed through advertised programmatic interfaces enabled by certain Internet protocols. Essentially, web service is an autonomous software system identified by URIs on Internet, which could be advertised, located, and invoked through messages encoded according to XML-based standards (e.g., SOAP, WSDL and UDDI) [2]. Generally, a newly generated service may be advertised in terms of both its functionality and non-functional attributes, i.e., QoS (quality of service, QoS) properties. The functionality indicates “what thing a service can do”, while the QoS properties specify “how the performance of a service is”. Since a single service is often limited in its functionality, to compose multiple services into a composite one, i.e., WSC (web service composition, WSC) is considered as a promising way to satisfy the more complex requirements of an end user.

Recently, a new trend of cloud computing has been observed and studied in both academy and industry domains. Essentially, cloud computing paradigm is a natural evolution of service computing towards a more flexible, dynamic, scalable and business-oriented environment [3]. As a key delivery platform in the field of service computing, cloud computing achieves a promising way for resources sharing via Internet [4]. By abstracting various computing resources into web services that could be invoked, cloud computing achieves the long-held dream of computing as a utility [5], which shifts the way that people design and use IT resources significantly.

As more and more service resources come into practice in cloud environment, more and more
service composition processes are enacted and executed inside a cloud platform. However, the web services held by a cloud platform is usually limited, compared with the various and approximately infinite application requirements from end users. Therefore, for certain task nodes of a service composition process, functional qualified candidate services could be found inside the requested cloud platform; while for some other task nodes, functional qualified candidate services are absent inside the requested cloud platform. In the former case, for a task node, its number of functional qualified candidate services is usually small. While in the latter case, for a task node, an outsourcing process is often employed to select an optimal service from the Internet; in this situation, for this task node, its number of functional qualified candidate services is usually large, because there are so many services that share similar functionality on the Internet [1, 6]. Therefore, if an outsourcing process is employed by a service composition process in cloud environment, a huge number of functional qualified composite solutions are usually available; in this situation, it is usually hard to determine the QoS-optimal composite solution within limited period, which blocks the popularity of service composition in cloud environment.

The remainder of the paper is organized as follows: the motivation of our paper is further specified in Section2. Section3 introduces the preliminary knowledge and related hypotheses for QoS-aware WSC. In Section4, a decision-making method is brought forth, to determine whether a QoS-aware WSC problem has a QoS qualified composite solution. A novel QoS-aware WSC method, i.e., LOEM (Local Optimization and Enumeration Method, LOEM) is put forward in Section5. The effectiveness and efficiencies of LOEM are demonstrated by an experiment in Section6. Evaluation is presented in Section7. Section8 summarizes the paper and draws directions for further research.

2. Motivation

In this section, a service composition scenario in cloud environment, which is adapted from the multimedia delivery application in [6], is presented to further clarify the motivation of our paper. As demonstrated in Fig.1, a smartphone end user requests multimedia data from an elastic cloud platform, where three tasks are referred in the form of service composition: transcoding task for transforming the various formats of multimedia data into those formats that could be supported by the smartphone; compression task for adapting the size of multimedia content to the wireless link of smartphone; payment task for calculating the delivered amount of data and paying the bill.

For the three tasks in Fig.1, their sets of functional qualified candidate services inside the elastic cloud platform are denoted by pool_{tra}, pool_{com} and pool_{pay}, respectively. As indicated in Fig.1, for transcoding task, its set of functional qualified candidate services inside the elastic cloud platform is not empty, i.e., pool_{tra}≠∅. Therefore, inside the elastic cloud platform, a candidate service could be selected for executing transcoding task; besides, the services held by an elastic cloud platform is usually limited, so the size of pool_{tra}, i.e., | pool_{tra} | is usually small. While for compression task, no functional qualified candidate service could be found inside the elastic cloud platform, i.e., pool_{com}=∅. In this situation, if the elastic cloud platform still expects to complete the whole service composition process, a service outsourcing process would be employed, to seek for functional qualified candidate services for compression task from Internet, after which a new set of functional qualified candidate services for compression task, i.e., Pool_{com} is obtained. As the services on Internet are nearly infinite, the number of services that could execute compression task,
i.e., $|Pool_{com}|$ is usually large. In the same way, for $payment$ task, its set of functional qualified candidate services inside the elastic cloud platform is empty, i.e., $pool_{pay} = \emptyset$. Therefore, a service outsourcing process is employed, after which a new set of functional qualified candidate services for $payment$ task, i.e., $Pool_{pay}$ is obtained. Similar to the case of $compression$ task, the number of services that could execute $payment$ task, i.e., $|Pool_{pay}|$ is usually large.

With the above analyses, service composition process is transformed as follows: from a small service set $pool_{tra}$, a candidate web service is selected for executing $transcoding$ task; from a large service set $Pool_{com}$, a candidate web service is selected for executing $compression$ task; from a large service set $Pool_{pay}$, a candidate web service is selected for executing $payment$ task. Therefore, totally $|pool_{tra} |*|Pool_{com} |*|Pool_{pay}|$ functional qualified composite solutions are obtained, each of which could execute the service composition process. As $|Pool_{com}|$ and $|Pool_{pay}|$ are usually large, $|pool_{tra} |*|Pool_{com} |*|Pool_{pay}|$ is large. In this situation, it is usually hard to determine the QoS-optimal composite solution from the $|pool_{tra} |*|Pool_{com} |*|Pool_{pay}|$ ones within limited period, while satisfying the end user’s QoS constraints. Especially when a quick response is required, the traditional QoS-aware service composition methods, e.g., Global [2] and Hybrid [6] exhibit their disadvantages, which bring a great challenge to seek for more efficient methods. In view of this challenge, in this paper, a QoS-aware composition method is investigated for supporting cross-platform service invocation in cloud environment. Concretely, a decision-making method for composite solution discovery and a novel QoS-aware service composition method, i.e., $LOEM$ are put forward, aiming at finding a QoS near-to-optimal composite solution with less time cost, when the number of composite solutions is large.
3. Preliminary Knowledge and Hypotheses

3.1 Preliminary Knowledge of QoS-aware WSC

In this subsection, some preliminary knowledge is introduced to clarify the process of QoS-aware WSC. Generally, for describing a QoS-aware WSC problem, the following concepts are necessary: task set $\mathbf{TK}$, service pool set $\mathbf{POOL}$, QoS criterion set $\mathbf{Crit}$, QoS constraint value set $\mathbf{Cons}$ and weight value set $\mathbf{Wgt}$. Next, we will introduce these five concepts respectively.

1. $\mathbf{TK} = \{t_1, \ldots, t_n\}$ is a task in a service composition schema and $n$ is the number of tasks. Considering the example in Fig.1, each node in the figure denotes a single task, e.g., Compression is a task that is responsible for compressing data information.

2. $\mathbf{POOL} = \{\mathbf{pool}_1, \ldots, \mathbf{pool}_l, \ldots, \mathbf{pool}_n\}$. $\mathbf{pool}_l (1 \leq l \leq n)$ is a service pool, which is comprised of the available services that could execute task $t_i$. Namely, $\mathbf{pool}_l = \{w_{s_1}^l, \ldots, w_{s_l}^l, \ldots, w_{s_l}^l\}$, where $w_{s_l}^l (1 \leq l \leq l)$ denotes the $k$-th candidate service in $\mathbf{pool}_l$ and $l$ is the number of candidate services in $\mathbf{pool}_l$. Please note that, $\mathbf{pool}_l (1 \leq l \leq n)$ may be a service pool inside an elastic cloud platform, or a service pool obtained from the Internet through the outsourcing process. Here, we do not distinguish these two different situations. For the convenient discussions, it is provided that $l$ candidate services are available in each $\mathbf{pool}_l (1 \leq l \leq n)$. Namely, there are $l$ available candidate services for each task $t_i (1 \leq i \leq n)$.

3. $\mathbf{Crit} = \{c_1, \ldots, c_j, \ldots, c_M\}$. $c_j (1 \leq j \leq M)$ is a QoS criterion of web service and $M$ is the number of QoS criteria that a service holds. For a service $w_s$, its QoS value over criterion $c_j$ could be denoted by $w_{s,c_j}$, which is regarded as a fixed value in this paper, e.g., $w_{s,duration} = 1s$.

4. $\mathbf{Cons} = \{\mathbf{cons}_1, \ldots, \mathbf{cons}_j, \ldots, \mathbf{cons}_m\}$. $\mathbf{cons}_j (1 \leq j \leq m)$ is a global constraint value over QoS criterion $c_j$ required by an end user, and $m$ is the number of QoS constraints. In this paper, it is assumed that $\mathbf{cons}_j$ is represented by a value domain $[7, 8]$, e.g., the duration constraint of an end user is: $\mathbf{cons}_{dura} \leq 3s$, i.e., $\mathbf{cons}_{dura} = [0s, 3s]$. For a service $w_s$ if $w_{s,c_j} \in \mathbf{cons}_j$, it can be concluded that service $w_s$ is qualified over criterion $c_j$. Besides, set $\mathbf{Cons} = \{\mathbf{cons}_1, \ldots, \mathbf{cons}_j, \ldots, \mathbf{cons}_m\} (1 \leq j \leq n)$ is employed to represent a set of sub-constraint values for task $t_i$, where $\mathbf{cons}_j$ is utilized to denote the constraint value over $c_j$ of task $t_i$.

5. $\mathbf{Wgt} = \{w_1, \ldots, w_j, \ldots, w_m\}$. $w_j (1 \leq j \leq m)$ is the weight value for criterion $c_j$ that is stressed by an end user, where $0 \leq w_j \leq 1$ and $\sum_j w_j = 1$ hold. Generally, $\mathbf{Wgt}$ plays a key role in service optimization selection for QoS-aware WSC.

For simplicity, it is assumed that $m = M$ holds (In the following discussions, the expressions are unified with “$m$”). Namely, for each QoS criterion $c_j (1 \leq j \leq m)$, there is a corresponding constraint value $\mathbf{cons}_j$ stressed by an end user. With the above five concepts, generally, QoS-aware WSC could be defined as below.

**Definition1:** QoS-aware WSC. QoS-aware web service composition, i.e., QoS-aware WSC could be represented by a five-tuple $\mathbf{QWSC(\mathbf{TK}, \mathbf{POOL}, \mathbf{Crit}, \mathbf{Cons}, \mathbf{Wgt})}$.

Taking advantage of Def.1, a general QoS-aware WSC process could be described as follows: For each task $t_i$ in $\mathbf{TK}$, select a service $w_{s_i}^l$ from $\mathbf{pool}_l$ (in $\mathbf{POOL}$) to execute $t_i$, then a functional qualified composite solution is derived. This phase is called *combination*. Afterwards, all the functional qualified composite solutions are evaluated, to determine whether their composite QoS
value over \( c_j \) (in \( \text{Crit} \)) could satisfy an end user’s QoS constraint value \( \text{cons}_j \) (in \( \text{Cons} \)), which is called constraints filter. Finally, from all the QoS qualified composite solutions, a QoS-optimal one is selected based on the weight value set \( \text{Wgt} \) through certain evaluation methods, e.g., SAW (Simple Additive Weighting, SAW) technique [2, 9], which is called optimization. After the above three phases, the obtained QoS-optimal composite solution would be returned to the end user, for the final execution of service composition process.

3.2 Related Hypotheses

To facilitate the subsequent discussions, some related hypotheses recruited in our QoS-aware service composition method are declared in this subsection.

**Hypothesis1:** Various composition models, e.g., sequential, parallel, alternative and loops are often present in an identical service composition problem. However, in this paper, only the sequential model is discussed, e.g., the case in Fig.1, as other composition models could be transformed into the sequential model by mature techniques [6, 10].

**Hypothesis2:** In the sequential composition model, the aggregation types of different QoS criteria are often varied. In this paper, the widely accepted aggregation types as well as aggregation functions introduced in [2] are employed, which are shown in Table1, where \( CS \) denotes a composite solution and \( ws_i \) represents a service selected for task \( t_i \) \((1 \leq i \leq n)\).

**Hypothesis3:** The end user’s QoS constraint over criterion \( c_j \) \((1 \leq j \leq m)\), i.e., \( \text{cons}_j \) is represented by a value domain \([\text{cons}_{j}^{\text{low}}, \text{cons}_{j}^{\text{upp}}]\). For the value of a negative QoS criterion \( c_j \), the smaller the better, so its constraint value \( \text{cons}_j \) could be abbreviated with a value domain \([0, \text{cons}_{j}^{\text{upp}}]\); while for the value of a positive QoS criterion \( c_j \), the larger the better, so its constraint value \( \text{cons}_j \) could be abbreviated with a value domain \([\text{cons}_{j}^{\text{low}}, \infty)\).

<table>
<thead>
<tr>
<th>Aggregation type</th>
<th>QoS criterion</th>
<th>Aggregation function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summation</td>
<td>price, duration</td>
<td>( CS.c_j = \sum_{i=1}^{n} ws_i . c_j )</td>
</tr>
<tr>
<td>Average</td>
<td>reputation</td>
<td>( CS.c_j = \frac{1}{n} \sum_{i=1}^{n} ws_i . c_j )</td>
</tr>
<tr>
<td>Multiplication</td>
<td>availability, success rate</td>
<td>( CS.c_j = \prod_{i=1}^{n} ws_i . c_j )</td>
</tr>
</tbody>
</table>

4. A Decision-Making Method for Solution Discovery in Service Composition Development

In a QoS-aware service composition problem, a set of QoS constraint values, i.e., \( \text{Cons} \) are often proposed by an end user, to specify his/her QoS expectation about the composite solutions. However, in certain circumstances, the QoS constraint values proposed by an end user are too rigid to find a QoS qualified composite solution. In this situation, service selection and composition are failed and much time is wasted on solution discovery. Therefore, before
introducing our QoS-aware composition method, a decision-making method should be developed first, to determine whether a WSC problem has a qualified composite solution that satisfies an end user’s QoS constraints. If the answer is “yes”, then our proposed composition method in Section5 would be employed to select a QoS-optimal composite solution, from all the QoS qualified ones; otherwise, a failure message is returned to the end user, as no QoS qualified composite solution is present for the QoS-aware WSC problem. Next, we will introduce how to design such a decision-making method for composite solution discovery.

Essentially, determining whether a QoS-aware WSC problem has a QoS qualified composite solution is a 0-1 CSP (Constraint Satisfaction Problems), whose time complexity is exponential. Therefore, the traditional decision-making method that traverse all the composite solutions is inappropriate, in certain situations that require a quick response. In view of this challenge, in this section, two simple but quick decision-making manners, i.e., **Max-Min manner** and **Local Optimization manner** are introduced to realize “partial” (not “all”) decision-making objective. In other words, through these two decision-making manners, partial QoS-aware WSC problems that has no QoS qualified composite solution are determined quickly and excluded beforehand, to reduce the unnecessary time cost. The two decision-making manners will be introduced in the following two subsections, respectively.

### 4.1 Max-Min manner

For the candidate services in pooli (corresponding to task ti, 1 ≤ i ≤ n), their minimal and maximal values over QoS criterion cj (1 ≤ j ≤ m), i.e., min'j and max'j could be calculated by statistic technique. Based on min'j, max'j and consj (i.e., [0, consjupp] or [consjlow, ∞)), a QoS-aware WSC problem with no QoS qualified composite solution could be quickly determined. According to different aggregation types in Table1, the minimal and maximal composite value over criterion cj (1 ≤ j ≤ m), i.e., MINj and MAXj could be calculated based on the following formulas.

\[
MIN_j = \sum_{i=1}^{n} \text{min}'_j \text{ or } \frac{1}{n} \sum_{i=1}^{n} \text{min}'_j \text{ or } \prod_{i=1}^{n} \text{min}'_j
\]

\[
MAX_j = \sum_{i=1}^{n} \text{max}'_j \text{ or } \frac{1}{n} \sum_{i=1}^{n} \text{max}'_j \text{ or } \prod_{i=1}^{n} \text{max}'_j
\]

For negative criterion cj, MINj and MAXj denote the best and worst values that a composite solution can achieve, respectively; while for positive criterion cj, MINj and MAXj denote the worst and best values of a composite solution, respectively.

- If MINj > consjupp holds for any one negative criterion cj or MAXj < consjlow holds for any one positive criterion cj, a conclusion could be drawn that the QoS-aware WSC problem has no QoS qualified composite solution, according to the end user’s QoS constraints.
- Otherwise, if MAXj ≤ consjupp holds for each negative criterion cj and MINj ≥ consjlow holds for each positive criterion cj, then it can be inferred that the QoS-aware WSC problem has at least one QoS qualified composite solution, according to the end user’s QoS constraints.

### 4.2 Local Optimization manner

In this subsection, a novel **Local Optimization manner** is introduced, to determine whether a QoS-aware WSC problem has a QoS qualified web service composite solution. For evaluating the quality of a web service composite solution, a widely accepted view is to calculate its aggregated
value of utility function [2]. In this subsection, the proposed Local Optimization manner is based on the decomposition of the composite solution’s utility function, which is of positive significance for the QoS criteria with aggregation types of “Summation” (e.g., price and duration) and “Average” (e.g., reputation).

Without loss of generality, in the rest of this subsection, $m$ negative QoS criteria $c_j$ ($1 \leq j \leq m$) are considered for illustration purpose. According to the widely used SAW technique, for a composite solution $CS$, its utility value $u_{CS}$ could be calculated by (1).

$$u_{CS} = \sum_{j=1}^{m} w_j \cdot \frac{\text{MAX}_j - CS.c_j}{\text{MAX}_j - \text{MIN}_j}$$

Where $w_j$ is the weight value of QoS criterion $c_j$ ($1 \leq j \leq m$), $CS.c_j$ is the composite value over criterion $c_j$ in composite solution $CS$, $\text{MAX}_j$ and $\text{MIN}_j$ denote the maximal and minimal composite values over criterion $c_j$ of all composite solutions respectively. Please note that $\text{MAX}_j = \sum_{i}^{n} \text{MAX}_{j_i}$ (for the QoS criteria with aggregation types of “Summation”) or $(1/n) \cdot \sum_{i}^{n} \text{MAX}_{j_i}$ (for the QoS criteria with aggregation types of “Average”) holds, where $\text{MAX}_{j_i}$ is the maximal value over QoS criterion $c_j$ of services in pool$_i$ (for task $t_i$), i.e., $\text{MAX}_{j_i} = \max_i (w_{j_i} c_{j_i})$ ($1 \leq i \leq l$) holds; $\text{MIN}_j = \sum_{i}^{n} \text{MIN}_{j_i}$ (for the QoS criteria with aggregation types of “Summation”) or $(1/n) \cdot \sum_{i}^{n} \text{MIN}_{j_i}$ (for the QoS criteria with aggregation types of “Average”) holds, where $\text{MIN}_{j_i}$ is the minimal value over QoS criterion $c_j$ of services in pool$_i$ (for task $t_i$), i.e., $\text{MIN}_{j_i} = \min_i (w_{j_i} c_{j_i})$ ($1 \leq i \leq l$) holds. Besides, $CS.c_j$ is the composite value over $c_j$ of composite solution $CS$, i.e., $CS.c_j = \sum_{i}^{n} w_{j_i} c_{j_i}$ (for the QoS criteria with aggregation types of “Summation”) or $(1/n) \cdot \sum_{i}^{n} w_{j_i} c_{j_i}$ (for the QoS criteria with aggregation types of “Average”). Based on the above analyses, the utility function formula of composite solution $CS$, i.e., $u_{CS}$ could be decomposed according to (2). Here, without loss of generality, the $m$ negative QoS criteria $c_j$ ($1 \leq j \leq m$) are all assumed to be with aggregation type of “Summation” (for the QoS criteria $c_j$ ($1 \leq j \leq m$) with aggregation type of “Average”, similar decomposition process is also available).

$$u_{CS} = \sum_{j=1}^{m} w_j \cdot \frac{\sum_{i}^{n} \text{MAX}_{j_i} - \sum_{i}^{n} w_{j_i} c_{j_i}}{\text{MAX}_j - \text{MIN}_j}$$

$$= \sum_{j=1}^{m} w_j \cdot \frac{\sum_{i}^{n} \text{MAX}_{j_i} - \sum_{i}^{n} w_{j_i} c_{j_i}}{\text{MAX}_j - \text{MIN}_j}$$

$$= \sum_{j=1}^{m} \frac{w_j \cdot (\text{MAX}_j - \sum_{i}^{n} w_{j_i} c_{j_i})}{\text{MAX}_j - \text{MIN}_j}$$

$$= \sum_{j=1}^{m} \frac{w_j \cdot (\text{MAX}_j - \sum_{i}^{n} w_{j_i} c_{j_i})}{\text{MAX}_j - \text{MIN}_j}$$

$$= \sum_{j=1}^{m} \frac{w_j \cdot (\text{MAX}_j - \sum_{i}^{n} w_{j_i} c_{j_i})}{\text{MAX}_j - \text{MIN}_j} + \ldots + \sum_{j=1}^{m} \frac{w_j \cdot (\text{MAX}_j - \sum_{i}^{n} w_{j_i} c_{j_i})}{\text{MAX}_j - \text{MIN}_j}$$
As can be seen from (2), the aggregated utility value of composite solution $CS_{optimal}$ could be decomposed into the sum of $n$ local utility values, each of which could be regarded as the evaluation basis of local service selection over task $t_i$ ($1 \leq i \leq n$). Namely, for each task $t_i$ ($1 \leq i \leq n$), a QoS-optimal candidate service, i.e., $ws_i$ will be selected by local selection formula $\sum_{j=1}^{n} w_j \times (\frac{\max_{j} - ws_j}{\max_{j} - \min_{j}})$; afterwards, the selected $n$ candidate services, i.e., $\{ws_1, \ldots, ws_n\}$ would compose a global QoS-optimal composite solution, if an end user’s QoS constraint values are not considered. For the $m$ negative QoS criteria $c_j$ ($1 \leq j \leq m$) with “Average” aggregation type, similar analyses could also be done, while the local selection formula over task $t_i$ ($1 \leq i \leq n$) turns to be $(1/n) \sum_{j=1}^{n} w_j \times (\frac{\max_{j} - ws_j}{\max_{j} - \min_{j}})$. Please note that, in the $m$ negative QoS criteria, if partial criteria are with “Summation” aggregation type while the remainders are with “Average” aggregation type, then the decomposition process in (2) still holds. This is because in the decomposition process for the QoS criteria with “Average” aggregation type, there is a “$1/n$” in the numerator and denominator, which could be eliminated simultaneously.

With the above analyses, Local Optimization manner could be launched as below. For each task $t_i$ ($1 \leq i \leq n$), a QoS-optimal service $ws_i$ is selected from $pool_i$, based on local utility formula $\sum_{j=1}^{n} w_j \times (\frac{\max_{j} - ws_j}{\max_{j} - \min_{j}})$ or $(1/n) \sum_{j=1}^{n} w_j \times (\frac{\max_{j} - ws_j}{\max_{j} - \min_{j}})$. Then a composite solution $CS_{optimal}\{ws_1, \ldots, ws_n\}$ is derived, which could be employed for the decision-making of composite solution discovery. For each QoS criterion $c_j$ ($1 \leq j \leq m$), calculate the aggregated value $CS_{optimal}\cdot c_j$ of composite solution $CS_{optimal}$.

- If $CS_{optimal}\cdot c_j \in cons_j$ holds for each QoS criteria $c_j$ ($1 \leq j \leq m$), then a conclusion could be drawn that a QoS-aware WSC problem has no QoS qualified composite solution, according to the user’s QoS constraints. The above conclusion could be proved by apagoge.

**Proof.** Without loss of generality, here, we consider two negative QoS criteria: $c_1$ and $c_2$, and two tasks: $t_1$ and $t_2$, for example. Assume $CS_{optimal}\{ws_1, ws_2\}$ is derived according to the Local Optimization manner, and $CS_{optimal}\cdot c_1 = ws_1 \cdot c_1 + ws_2 \cdot c_1 > cons_1^{\text{up}}$, $CS_{optimal}\cdot c_2 = ws_1 \cdot c_2 + ws_2 \cdot c_2 > cons_2^{\text{up}}$. If there is a QoS qualified composite solution $CS_{q}\{ws_1^q, ws_2^q\}$, where $CS_{q}\cdot c_1 = ws_1^q \cdot c_1 + ws_2^q \cdot c_1 \leq cons_1^{\text{up}}$ and $CS_{q}\cdot c_2 = ws_1^q \cdot c_2 + ws_2^q \cdot c_2 \leq cons_2^{\text{up}}$ hold, then the following formulas are available according to the utility function in (1):

$$u_{CS_{optimal} - c_1} = w_1 \times (\frac{\max_1}{\max_1 - \min_1} + w_2 \times (\frac{\max_2}{\max_2 - \min_2})) - (w_1 \times (\frac{CS_{optimal}\cdot c_1}{\max_1 - \min_1} + w_2 \times (\frac{CS_{optimal}\cdot c_1}{\max_2 - \min_2}))$$

$$u_{CS_{optimal} - c_2} = (w_1 \times (\frac{\max_1}{\max_1 - \min_1} + w_2 \times (\frac{\max_2}{\max_2 - \min_2})) - (w_1 \times (\frac{CS_{q}\cdot c_1}{\max_1 - \min_1} + w_2 \times (\frac{CS_{q}\cdot c_2}{\max_2 - \min_2})).$$

On one hand, as $CS_{optimal}\cdot c_1 > CS_{q}\cdot c_1$ and $CS_{optimal}\cdot c_2 > CS_{q}\cdot c_2$ hold, $u_{CS_{optimal} - c_1} < u_{CS_{q}}$ holds. On the other hand, according to (2), $u_{CS_{optimal} - c_1} = u_{CS_{q}} + u_{1}^{c_1}$ and $u_{CS_{optimal} - c_2} = u_{CS_{q}} + u_{2}^{c_2}$ hold, where $u_1^{c_1}$ denotes the utility value of $ws_1$. As $u_1^{c_1}$ and $u_2^{c_1}$ are local QoS-optimal services for task $t_1$ and $t_2$ respectively, $u_1^{c_1} \geq u_1^{c_1}$ and $u_2^{c_1} \geq u_2^{c_1}$ hold, i.e., $u_{CS_{optimal} - c_1} \geq u_{CS_{q}}$ holds. Consequently, a contradiction is raised as $u_{CS_{optimal} - c_1} < u_{CS_{q}}$ and $u_{CS_{optimal} - c_2} \geq u_{CS_{q}}$ hold simultaneously, by which our conclusion is proven.
If $CS_{optimal} \in cons_j$ holds for all QoS criteria $c_j(1 \leq j \leq m)$, then a conclusion could be drawn that the QoS-aware WSC problem has at least one QoS qualified composite solution, according to an end user’s QoS constraints. In this situation, no QoS constraints are violated, so local optimality leads to global optimality, and the optimal solution is $CS_{optimal} = \{ w_{nws}^1, \ldots, w_{nws}^k \}$.

Next, the time complexity of the two manners proposed for decision-making will be analyzed.

Let $|TK| = n$, $|Crit| = m$ and $|pool| = l$ ($1 \leq i \leq n$), then the time complexity of the Max-Min manner is $O(2^n \cdot m \cdot n \cdot l \cdot g + 2^n \cdot m \cdot n) = O(m \cdot n \cdot l \cdot g)$; while for the Local Optimization manner, its time complexity is $O(2^n \cdot m \cdot n \cdot l \cdot g + 2^n \cdot m \cdot n + n \cdot (m \cdot l \cdot g) + m \cdot n + m) = O(m \cdot n \cdot g)$. Besides, according to the former analyses, the Max-Min manner could be employed for the decision-making of QoS-aware WSC problems with various aggregation types (“Summation”, “Average” and “Multiplication”). While the Local Optimization manner could be adopted, only when the aggregation types of QoS criteria are “Summation” and “Average”.

After the decision-making of composite solution discovery, partial QoS-aware WSC problems with no QoS qualified solution would be determined; in this situation, generally, a failure message is returned to the end user, to ask for relaxing his/her rigid QoS constraints for another service selection. While for the WSC problems with QoS qualified composite solutions, a QoS-aware service composition method named LOEM will be introduced in the next section, to pursue a QoS near-to-optimal composite solution with less time cost.

5. A QoS-Aware Service Composition Method: LOEM

In this section, a novel QoS-aware service composition method, i.e., LOEM will be introduced. The main theory behind LOEM is: for each task $t_i(1 \leq i \leq n)$ referred in a service composition process, filter its $l$ candidate services into $h$ ($h << l$) promising ones by local optimizing selection; after that, with the obtained $h$ promising web services for each task, enumerate all the possible web service composite solutions to pursue a QoS near-to-optimal one. The four steps of LOEM are listed in Fig. 2.

**Step1:** For each QoS criterion $c_j(1 \leq j \leq m)$ of each task $t_i(1 \leq i \leq n)$, the minimal and maximal values over $c_j$ of candidate services in $pool_i$, i.e., $\min_j^t$ and $\max_j^t$ are calculated.

**Step2:** For each QoS criterion $c_j(1 \leq j \leq m)$ of each task $t_i(1 \leq i \leq n)$, $d$ quality levels, i.e., $\{q_1^j, \ldots, q_d^j\}$ are obtained by discretizing the value domain $[\min_j^t, \max_j^t]$ into $d$ values. Each quality level corresponds to a QoS constraint value over $c_j$ of $t_i$, which plays a key role in local optimizing selection over task $t_i$.

**Step3:** For each task $t_i(1 \leq i \leq n)$, $h$ promising services are derived by local optimizing selection.

**Step4:** With the $h$ promising candidate services for each task $t_i(1 \leq i \leq n)$, a MIP (Mixed Integer Programming, MIP) problem is derived to enumerate all the possible composite solutions. Finally, a QoS near-to-optimal composite solution, i.e., $CS_{n-t-o} = \{ w_{nws}^1, \ldots, w_{nws}^k \}$ is selected for the final execution of service composition.
(1) **Step1:** Determine the minimal and maximal values of each QoS criterion for each task.

For each task \( t_i (1 \leq i \leq n) \), repeat the following operations: for each QoS criterion \( c_j (1 \leq j \leq m) \), the minimal and maximal values over \( c_j \) of candidate services in service pool \( \text{pool}_i \) (corresponds to task \( t_i \)), i.e., \( \text{min}^i_j \) and \( \text{max}^i_j \) should be determined first. Generally, this step could be achieved by statistical processing.

(2) **Step2:** Discretize the value domain \( [\text{min}^i_j, \text{max}^i_j] \) into \( d \) quality levels.

For each QoS criterion \( c_j (1 \leq j \leq m) \) of any task \( t_i (1 \leq i \leq n) \), \( d \) quality levels are available by discretizing the value domain \( [\text{min}^i_j, \text{max}^i_j] \) into \( d \) concrete values, i.e., \( \{q^1_j, \ldots, q^d_j\} \), where \( \text{min}^i_j \leq q^1_j \leq \ldots \leq q^d_j \leq \text{max}^i_j \) holds. This step is similar with the division process in [6]. However, in this paper, the average idea is recruited for calculating the value of \( q^d_j \) (\( 1 \leq d \leq d \)) for simplicity. In other words, the following equations hold in the discretization process:

\[
q^1_j = \text{min}^i_j, \quad q^2_j = \text{min}^i_j + \frac{\text{max}^i_j - \text{min}^i_j}{d - 1}, \quad q^d_j = \text{max}^i_j.
\]

Take task \( t_1 \) and four QoS criteria: negative price and duration, positive availability and reputation as an instance. Here, for simplicity, the units of various QoS criteria are omitted. Suppose that the four value domains derived in Step1 are \([20, 80],[0, 10],[0.8, 0.85],[0.7, 0.85]\) respectively, and \( d=4 \) holds, then four quality levels are derived for each of the four QoS criteria, which are shown in Table2.

<table>
<thead>
<tr>
<th>Quality level</th>
<th>price</th>
<th>duration</th>
<th>availability</th>
<th>reputation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q^1_j )</td>
<td>20</td>
<td>10</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>( q^2_j )</td>
<td>40</td>
<td>20</td>
<td>0.6</td>
<td>0.75</td>
</tr>
<tr>
<td>( q^3_j )</td>
<td>60</td>
<td>30</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>( q^4_j )</td>
<td>80</td>
<td>40</td>
<td>0.8</td>
<td>0.85</td>
</tr>
</tbody>
</table>

(3) **Step3:** Find \( h \) promising candidate services for task \( t_i (1 \leq i \leq n) \) by local optimizing selection.

In this step, with the derived quality levels in Step2, a local optimizing selection process is recruited, to filter the \( l \) candidate services of task \( t_i \) into \( h (h<l) \) promising ones. According to the previous analyses, each quality level \( q^{\hat{i}}_j \) (\( 1 \leq \hat{i} \leq d \)) corresponds to a local constraint value \( \text{cons}^i_j = [0, q^{\hat{i}}_j] \) (\( \text{cons}^i_j = [q^{\hat{i}}_j, \infty] \)), over a negative (positive) QoS criterion \( c_j (1 \leq j \leq m) \) of task \( t_i (1 \leq i \leq n) \). So for a negative QoS criterion \( c_j \), the smaller the value of \( q^{\hat{i}}_j \) is, the better the quality of services that satisfy constraint \([0, \text{cons}^i_j]\) will be; while for a positive QoS criterion \( c_j \), the larger the value of \( q^{\hat{i}}_j \) is, the better the quality of services that satisfy constraint \([q^{\hat{i}}_j, \infty]\) will be. Therefore, the local optimizing selection process could be clarified as follows.

For each task \( t_i (1 \leq i \leq n) \), determine its local QoS constraint value set \( \text{Cons}^i = \{\text{cons}^1_j, \ldots, \text{cons}^i_j, \ldots, \text{cons}^m_j\} \). Initially, \( \text{cons}^1_j = [0, q^1_j] \) holds for each negative QoS criterion \( c_j \); while for each positive QoS criterion \( c_j \), \( \text{cons}^i_j = [q^d_j, \infty] \) holds. Let us consider the example in Table2, the initial constraint values for task \( t_1 \) is \( \text{Cons}^1 = [[0, 20], [0, 10], [0.8, \infty], [0.85, \infty]] \). Then according to \( \text{Cons}^i = [[0, 20], [0, 10], [0.8, \infty], [0.85, \infty]] \), local service selection is performed in service pool \( \text{pool}_1 \). Here, the utility value of a candidate service \( w^i_q \), i.e., \( u^i_q \) could be calculated by the utility functions in (3)-(5), and finally a candidate service with the largest utility value is selected.
\[ u_{k}^{+} = \sum_{j=1}^{n} w_{j}^{k} \] 
\[ \frac{\max_{j=1}^{n} - ws_{j}^{k} \cdot c_{j}}{\max_{j=1}^{n} - \min_{j=1}^{n}} + \frac{1}{\sum_{j=1}^{n} w_{j}^{k}} \] 
\[ u_{k}^{-} = \sum_{j=1}^{n} w_{j}^{k} \] 
\[ \frac{\min_{j=1}^{n} - ws_{j}^{k} \cdot c_{j}}{\max_{j=1}^{n} - \min_{j=1}^{n}} + \frac{1}{\sum_{j=1}^{n} w_{j}^{k}} \] 
\[ v_{k}^{+} = u_{k}^{+} + u_{k}^{-} \]

where \( w_{j} \) represents the weight value of QoS criterion \( c_{j} \); \( u_{k}^{+} \), \( u_{k}^{-} \), and \( v_{k}^{+} \) denote the utility values of negative and positive QoS criteria of service \( ws^{k} \) in \( \text{pool} \), respectively; \( u^{k} \) is the total utility value of \( ws^{k} \); here, \( m \) QoS criteria are classified into the following four categories: \( c_{j_{1}} (c_{j_{2}}) \) is a negative (positive) QoS criterion with aggregation type of “Summation” or “Multiplication”, while \( c_{j_{1}} (c_{j_{2}}) \) is a negative (positive) criterion with aggregation type of “Average”; \( ws^{k} \cdot c_{j} \) represents the value over QoS criterion \( c_{j} \) of candidate service \( ws^{k} \); \( max_{j} \) and \( min_{j} \) denote the largest and smallest values over QoS criterion \( c_{j} \) of candidate services in \( \text{pool} \), respectively; while \( MAX_{j} \) and \( MIN_{j} \) represent the largest and smallest composite values over QoS criterion \( c_{j} \), whose computation manners are specified in Table 3 according to the aggregation type of QoS criterion \( c_{j} \). Here, for each task \( t_{i} \), if the result of local service selection is empty, a relaxation process is executed for all constraint values in set \( \text{Cons}_{i} \), i.e., \( \text{cons}_{j} \) (1 ≤ \( j \) ≤ \( m \)) simultaneously for re-selection. Here, for any negative criterion \( c_{j} \), relaxation means that \( \text{cons}_{j} \) is altered from \([0, q_{j}^{-}] \) to \([0, q_{j}^{-} + 1] \) (1 ≤ \( z < d \)); while for any positive criterion \( c_{j} \), relaxation means \( \text{cons}_{j} \) is varied from \([q_{j}^{-}, \infty) \) to \([q_{j}^{-} - 1, \infty) \) (1 < \( z \leq d \)). Considering the example in Step 2, constraint value set \{[0, 40], [0, 20], [0.7, \infty), [0.8, \infty)\} would be regarded as a relaxation of the initial constraint value set \( \text{Cons}_{1} \). After relaxation, an updated constraint value set is derived. After that, with the updated constraint value set, repeat the local service selection process.

1) Otherwise, a candidate service with the largest utility value is returned for task \( t_{i} \), denoted by \( ws_{i}^{k_{i}} \). Afterwards, a relaxation process is performed for each constraint value \( \text{cons}_{j} \in \text{Cons}_{i} \), and an updated constraint value set is obtained. According to the updated constraint value set, next service selection is performed, during which the candidate services that satisfy the prior constraint value set will not be considered. Considering the example in Step 2, if a candidate \( ws^{k_{i}} \)
is qualified according to the initial constraint value set \( \text{Cons}^1 \{ [0, 20], [0, 10], [0.8, \infty], [0.85, \infty] \} \), then \( ws \) will not be considered when performing the second service selection according to the updated constraint value set \{ [0, 40], [0, 20], [0.7, \infty], [0.8, \infty] \}.

For each task \( t_i (1 \leq i \leq n) \), repeat the above local selection process until \( h (h \geq 1) \) local-optimal candidate services are achieved, or no relaxation is available for any constraint value in \( \text{Cons}^1 \). Thus, at most \( h \) candidate services are derived for each task \( t_i \), i.e., \( P_{-pool} \{ w_{i}^{k_1}, \ldots, w_{i}^{k_h} \} \). The derived \( h \) candidate services are qualified according to the relatively rigid QoS constraint values, so in this paper, they are regarded as promising.

(4) Step4: With the \( h \) (at most \( h \)) promising candidate services for each task \( t_i (1 \leq i \leq n) \), enumerate all the composite solutions to pursue a QoS near-to-optimal one.

After the previous three steps, for each task \( t_i (1 \leq i \leq n) \), its number of functional qualified candidate services is decreased from \( l \) to \( h \) \((h < l)\). Thus the original QoS-aware WSC problem is transformed into the following new one: for each task \( t_i (1 \leq i \leq n) \), there are at most \( h \) functional qualified composite solutions available at most; finally, a composite solution, which satisfies the end user’s global QoS constraints \( \text{Cons} \) and achieves the maximal utility value, is regarded as a QoS near-to-optimal composite solution for the original QoS-aware WSC problem. In this step, the QoS-aware WSC problem is transformed into a MIP one, as the linear programming method is very effective when the size of problem is small [6]. In the MIP problem depicted in (6)-(13), the employed variable is \( x_i^k \), which is binary. If a candidate service \( w_{i}^{k} \) is selected for the final service composition, \( x_i^k = 1 \); otherwise, \( x_i^k = 0 \).

Maximize \( \sum_{j=1}^{n} w_j \cdot \frac{\text{MAX}_j - \text{CS}_j}{\text{MAX}_j - \text{MIN}_j} + \sum_{j=1}^{n} w_j \cdot \frac{\text{CS}_j - \text{MIN}_j}{\text{MAX}_j - \text{MIN}_j} \) \hspace{1cm} (6)

Subject to
\[
\text{CS}_j \leq w_j \cdot \text{cons}_j^{upp} \quad \text{(7)}
\]
\[
\text{CS}_j \geq w_j \cdot \text{cons}_j^{low} \quad \text{(8)}
\]
\[
\text{CS}_j = \sum_{k=1}^{h} \sum_{j=1}^{n} w_{i}^{k_j} \cdot \text{c}_j \cdot x_i^k \quad \text{or} \quad \frac{1}{h}\sum_{k=1}^{h} \sum_{j=1}^{n} w_{i}^{k_j} \cdot \text{c}_j \cdot x_i^k \quad \text{or} \quad \prod_{k=1}^{h} (w_{i}^{k_j} \cdot \text{c}_j \cdot x_i^k) \quad \text{(9)}
\]
\[
\sum_{j=1}^{n} x_i^k = 1, x_i^k \in \{0, 1\} \quad \text{(10)}
\]
\[
\sum_{j=1}^{n} w_j = 1, w_j \in \{0, 1\} \quad \text{(11)}
\]

Where \( c_j \) and \( c_j^+ \) denote the negative and positive QoS criteria respectively; \( \text{CS} \) is a composite solution, whose criterion value \( \text{CS}_c \) could be calculated based on the aggregation type of \( c_j \) in Table1; \( \text{MAX}_j \) and \( \text{MIN}_j \) represent the maximal and minimal composite values over QoS criterion \( c_j \), which are specified in Table3. Please note that the composite utility value in (6) is calculated by the SAW technique [9]; while (7)-(11) formalize a group of constraint conditions that should be satisfied in the MIP problem.

After solving the derived MIP problem with any MIP solver, a QoS near-to-optimal composite solution \( \text{CS}_{n-to-o} \{ w_{1}^{k}, \ldots, w_{n}^{k} \} \) is achieved, which is regarded as the final composite solution to the original QoS-aware WSC problem in this paper. The pseudocode of our proposed LOEM is specified as below. With regards to the meanings of \( \text{TK}, \text{POOL}, \text{Crit}, \text{Cons}, \text{Wgt} \), please refer to Def.1 in Section3.
LOCAL OPTIMIZATION AND ENUMERATION (TK, POOL, Crit, Cons, Wgt)

1: for \( i \leftarrow 1 \) to \( n \)
2:   do for \( j \leftarrow 1 \) to \( m \)
3:     do calculate \( \min_j^i \) and \( \max_j^i \) by statistic technique
4:       DISCRETIZATION (\( [\min_j^i, \max_j^i] \), \( d \))
5: for \( i \leftarrow 1 \) to \( n \)
6:   do for \( j \leftarrow 1 \) to \( m \)
7:     do if \( c_j \) is negative
8:       then \( \text{Cons}_i^j = q_j^i \)
9:       else \( \text{Cons}_i^j = q_j^i \)
10: an initial sub-constraint value set for \( t_i \) is derived, i.e., \( \text{Cons}_i^j \{ \text{Cons}_1^j, \ldots, \text{Cons}_m^j \} \)
11: for \( i \leftarrow 1 \) to \( n \)
12:   do for \( p \leftarrow 1 \) to \( d \)
13:     do RELAX (\( \text{Cons}_i^j \), \( p \))
14:     LOCAL SELECTION (\( Wgt, \text{Cons}_i^j, \text{pool}_i, p \))
15:       if \( |P\text{-pool}_i|=h \)
16:         then break
17: MIXED INTEGER PROGRAMMING (\( Wgt, \text{Cons}_i, \text{P\text{-pool}}_i \))
18:    solve the MIP problem and a near-to-optimal composite solution: \( CS_{n-k} \{ \text{ws}_1^k, \ldots, \text{ws}_n^k \} \) is returned.

(1) Function DISCRETIZATION (\( [\min, \max] \), \( d \)) is employed to discretize the value domain \( [\min, \max] \) into \( d \) quality levels, whose pseudocode is specified as below.

DISCRETIZATION (\( [\min, \max] \), \( d \))
1: for \( z \leftarrow 1 \) to \( d \)
2:   do \( q_z^i = \min^i + (z-1) \times (\max^i - \min^i) / (d-1) \)
3: return \( q_1^i, \ldots, q_d^i \)

(2) Function RELAX (\( \text{Cons}_i^j \), \( p \)) is employed to relax QoS constraint \( \text{Cons}_i^j \) with \( p \) step-sizes, whose pseudocode is specified as below.

RELAX (\( \text{Cons}_i^j \), \( p \))
1: for \( \forall \text{Cons}_i^j \in \text{Cons}_i \)
2:   do if \( c_j \) is negative
3:     then \( \text{Cons}_i^j = p^i \times (\max - \min) / (d-1) \)
4:     else \( \text{Cons}_i^j = -p^i \times (\max - \min) / (d-1) \)
5: return updated \( \text{Cons}_i^j \)

(3) Function LOCAL SELECTION (\( Wgt, \text{Cons}_i^j, \text{pool}_i, p \)) is employed to perform the \( p \)-th local service selection in \( \text{pool}_i \), according to \( Wgt \) and \( \text{Cons}_i^j \), whose pseudocode is specified as below.

LOCAL SELECTION (\( Wgt, \text{Cons}_i^j, \text{pool}_i, p \))
1: for \( \forall \text{ws} \in \text{pool}_i \)
2:   do for \( \forall \text{Cons}_i^j \in (\text{RELAX}(\text{Cons}_i^j, p) - \text{RELAX}(\text{Cons}_i^j, p-1)) \)
3:     do if \( \text{ws}.c_j \notin \text{Cons}_i^j \)
4:       then break
5: calculate \( \text{ws} \)'s utility value based on (3)-(5)
6: add the candidate service with the maximal utility value into a promising pool \( P\text{-pool}_i \).
(4) Function MIXED INTEGER PROGRAMMING (Wgt, Cons, P-pool) is employed to transform the WSC problem into a MIP problem, based on Wgt, Cons and web services in P-pool, whose pseudocode is specified as below.

MIXED INTEGER PROGRAMMING (Wgt, Cons, P-pool)

1:  a MIP problem is derived according to (6)-(11).

6. Experiment

In this section, some experiments are deployed to validate our hypothesis: LOEM can achieve better composition efficiency than traditional QoS-aware composition methods, e.g., Global method [2] and Hybrid method [6], while a QoS near-to-optimal composite solution is derived. Correlated comparison analyses are also demonstrated after the experiments.

6.1 Experiment Deployment

In our experiments, the data are generated randomly in a range [0, 100], to simulate the QoS properties of employed service resources, e.g., reputation and price. 100 sets of QoS constraint value set Cons are also randomly generated, to simulate the actual service composition process in the example scenario of Fig.1, and their average experiment results are adopted finally. In this experiment, we only focus on the WSC problems with QoS qualified solutions. Namely, each constraint value set Cons ensures that at least a QoS qualified composite solution exists.

The experiments were conducted on a Founder N600 machine with Intel Pentium 2.80 GHz processor and 512 MB RAM. The machine is running under Windows XP (Service Pack 3) and JAVA 1.5. Each experiment was carried out 30 times, in order to reduce interferences in the form of outliers from the host’s operating system, and the average solving time was registered finally.

6.2 Experiment Results and Analyses

In order to evaluate the feasibility of LOEM, four evaluation profiles are tested and compared with Global and Hybrid methods. In the experiment, the employed parameter d is fixed to 10 for both Hybrid and LOEM, and four QoS constraints are employed for all test cases, i.e., m=4 holds.

Profile 1: performances of LOEM and other two methods

In Fig.3, the performances of three methods are compared with respect to the number of tasks (i.e., n), where h=6 holds for our proposed LOEM. In this experiment, the number of tasks, i.e., n varies from 5 to 25, and the number of candidate services, i.e., l for each task is fixed to 50. As can be seen from Fig.3, the computation time increases approximately exponentially when the number of tasks grows for all three methods; however, LOEM outperforms other two methods significantly. The performances with respect to the number of candidate services (i.e., l) are shown in Fig.4, where l varies from 50 to 500 and the number of tasks (i.e., n) is fixed to 5. As can be seen from Fig.4, the computation time of Global rises acutely when l is increased, while other two methods stay relatively stable. However, our proposed LOEM still outperforms Global and Hybrid.

Profile 2: optimality of LOEM and other two methods

In this profile, the number of promising candidates for each task, i.e., h varies from 2 to 6. As local optimizing selection is employed in LOEM, the derived composite solutions may not be the QoS-optimal one, but only a QoS near-to-optimal one. So in this profile, the optimality of LOEM
should be tested, by comparing the utility value of composite solutions generated by \textit{LOEM} and \textit{Global}, i.e., optimality = \( u_{\text{LOEM}} / u_{\text{Global}} \), where \( u_{\text{LOEM}} \) and \( u_{\text{Global}} \) could be calculated based on the utility function in (6) respectively. The optimality of \textit{LOEM} with respect to the number of tasks (i.e., \( n \)) is shown in Fig.5; while in Fig.6, the optimality of \textit{LOEM} with respect to the number of candidates (i.e., \( l \)) is demonstrated. As can be seen from the two figures, the optimality of \textit{LOEM} is approaching 100\% and only has a slight change with the increase of \( n \) and \( l \).

\textbf{Profile 3: performance of \textit{LOEM} with respect to the size of \( h \)}

As \( h \) (at most \( h \)) promising candidates are returned for each task, the size of \( h \) is a key factor for the efficiency of \textit{LOEM}, which is exposed in Fig.7. In the figure, the number of promising candidates, i.e., \( h \) varies from 1 to 6, and the number of tasks, i.e., \( n \) is altered from 5 to 25. The experiment results demonstrate that the computation time of \textit{LOEM} increases approximately exponentially with the rise of \( h \).

\textbf{Profile 4: failure rate of \textit{LOEM} with respect to the size of \( h \)}

As \textit{LOEM} does not traverse all the composite solutions, failure is inevitable. Namely, there exists the possibility that no solution can be found by \textit{LOEM}, in a QoS-aware WSC problem with a QoS qualified solution. This is due to the fact that the number of candidate services for each task is reduced from \( l \) to \( h \) by local optimizing selection. So \( h \) is a key factor for the failure rate of \textit{LOEM}, which is demonstrated in Fig.8. The experiment results show that the failure rate decreases acutely with the increase of \( h \). Another observation available from Fig.7-8 is that there is a tradeoff between the performance and failure rate for \textit{LOEM}.
7. Evaluation

In this section, we analyze the time complexity of LOEM introduced in Section 5 to evaluate the feasibility of our proposal. A comparison with related work is also presented. This is followed by discussions regarding the limitations and some possible extension of our work.

7.1 Complexity Analysis

In this subsection, the time complexity of the four steps of LOEM will be analyzed in sequence. Let $|\text{Cons}|=m$, $|\text{TK}|=n$, $|\text{pool}_i|=l$, $|P-\text{pool}_i|=h$ and $d$ is the number of quality levels discretized from value domain $[\text{min}_i', \text{max}_i']$.

1) Time complexity for determining the minimal and maximal values for each QoS criterion of each task

For each QoS criterion $c_j (1 \leq j \leq m)$ of each task $t_i (1 \leq i \leq n)$, a value domain $[\text{min}_i', \text{max}_i']$ could be obtained by the statistic processing, whose time complexity is $O(l)$. So the overall time complexity of this substep is $O(m*n*l)$.

2) Time complexity for discretizing the value domain $[\text{min}_i', \text{max}_i']$ into $d$ quality levels

For each value domain $[\text{min}_i', \text{max}_i']$ ($1 \leq j \leq m$, $1 \leq i \leq n$), the numbers of needed operations “+”, “-”, “*”, “/” are $d$, $3*d$, $d$ and $d$ respectively. Totally, $m*n$ value domains are present in the discretization step, so the time complexity is $O(m*n*(d+3*d+d+d))=O(6*m*n*d)=O(m*n*d)$.

3) Time complexity for finding $h$ promising candidate services for task $t_i$ ($1 \leq i \leq n$) by local optimizing selection

Here, only the negative QoS criteria are considered for simplicity. Local optimizing selection may return a null result if the selected local constraint set, i.e., $\text{Cons}_i'\{[0, q_{ij}'] | 1 \leq j \leq m\}$ for task $t_i$ ($1 \leq i \leq n$) are rigid. So the first qualified value of $q_{ij}'$ (namely, at least a service is qualified according to the constraints) for task $t_i$ could be $q_{ij}'$, or $q_{ij}^2$, …, or $q_{ij}^d$.

1) The best case. The best case means the first constraint value set $\text{Cons}_i'\{[0, q_{ij}'] | 1 \leq j \leq m\}$ is qualified for task $t_i$ ($1 \leq i \leq n$). So in this situation, local selection should be performed for $h$ times in order to find $h$ promising candidate services for each task $t_i$. As each local selection requires $m*l$ comparison operations, the time complexity is $O(m*l*h*n)$.
2) The worst case. The worst case means the last constraint value set $C_{\text{cons}}^t\{[0, q^t_j]|1 \leq j \leq m\}$ is qualified for task $t_i$ ($1 \leq i \leq n$), while the former $d-1$ constraint value sets, i.e., $C_{\text{cons}}^t\{[0, q^t_j]|1 \leq j \leq m\}(1 \leq d \leq d-1)$, all fail in finding a QoS-qualified candidate service. So in this situation, $d$ times of local selection are needed to find $h$ (here, only one candidate is promising, i.e., $h=1$) promising candidate services for each task $t_i$. As each local selection requires $m*l$ comparison operations, so the time complexity is $O(m*l*d*n)$.

(4) Time complexity for enumerating all the composite solutions to seek a near-to-optimal one

In the derived MIP problem, at most $h(1 \leq h \leq d)$ candidate services are available for each task $t_i$ ($1 \leq i \leq n$). So the number of employed binary variables $x_i^k$ is $n*h$. As the time complexity of MIP solvers is exponential, so the time complexity of this step is $O(2^{n*h})$.

Based on the former analyses, the total time complexity of our proposed $LOEM$ is $O(m*n*l+m*n*d+m*l*d*n+2^{n*h})= O(2^{n*h})$. So the time complexity of $LOEM$ is dominated by the step of mixed integer programming, whose complexity only depends on the number of tasks (i.e., $n$) and the number of selected promising candidates (i.e., $h$) for each task. So if the number of promising candidates $h$ satisfies: $h < m*d$ and $h < l$, we can ensure that the size of our MIP model is smaller than the size of models used in [2, 6] (whose numbers of decision variables are $n*m*d$ and $n*l$ respectively).

7.2 Related Work and Comparison Analyses

As a prominent way to improve IT practices, cloud computing has gained ever-increasing attention in both academic and industrial domains. In [5], three types of services for deploying a cloud computing platform are proposed: Infrastructure as a Service(IaaS), Platform as a Service(PaaS), and Software as a Service(SaaS). In [11], GVO (Guest Virtual Organization, GVO) is proposed to represent the owners of available computing resources. According to different strategies, a GVO may advertise its resources to the public directly or lend the resources to a cloud platform for renting purpose. An elastic cloud platform is proposed in [12], where the authors suggest that the services held by a cloud platform is limited compared with end users’ various application requirements, and an outsourcing process should be employed when the needed services are absent in a cloud platform.

Web service composition has become a promising way to fulfill the complex user requirements in cloud environment, and QoS-aware WSC is gaining more and more attention [1, 13, 14, 15, 16]. The QoS-aware WSC problem is proven to be NP-hard [17], so when the problem space is large, it is usually time consuming to derive a QoS-optimal service composite solution. In [2], a Global method is proposed to solve the QoS-aware WSC problem, which models the QoS-aware WSC problem as a MIP problem and decreases the time cost to some extent. In [13], a combinatorial model and a graph model are suggested to model the QoS-aware WSC problem: according to the combinatorial model, the QoS-aware WSC problem is modeled as a multi-dimension multi-choice 0-1 knapsack problem (i.e., MMKP); while according to the graph model, the QoS-aware WSC problem is modeled as a multi-constraint optimal path (i.e., MCOP) problem. However, the candidate space of the derived MIP problem is still large, especially when the number of tasks and the number of candidate services for each task are large. An ant colony optimization-based service composition method, i.e., MO-ACO is proposed in [14], to improve the composition efficiency. However, the above methods are usually inefficient for solving the QoS-aware WSC problem,
where numerous candidate services are available for each task involved in service composition. In [6], a Hybrid method is proposed to reduce the candidate space by transforming the global selection into local one, which improves the composition efficiency significantly; however, for transforming the global selection into local one, much time is still needed. In [18], a heuristic service composition method named SLOMIP is proposed, which is based on Skyline and local service selection. However, SLOMIP is suitable for the WSC problem with only “Summation” aggregation type. A genetic algorithm-based web service composition approach, named $A-G$, is proposed in [19], to improve the service composition efficiency; however, the choice of genetic operators (i.e., selection, mutation and recombination) has a big influence on the algorithm efficiency and correctness, which may lead to unstable service composition results.

In order to further improve the efficiency of WSC, in this paper, a QoS-aware composition method is investigated for supporting cross-platform service invocation in cloud environment. Concretely, two aspects of improvement are brought forth. Firstly, to reduce the unnecessary time cost, a decision-making method is proposed to quickly determine whether a QoS-aware WSC problem has a QoS qualified composite solution, and two manners are introduced to support this decision-making process, i.e., Max-Min manner and Local Optimization manner. If no QoS qualified solution exists, a QoS-aware WSC problem would be abandoned directly and a failure message is returned to the end user, so as to reduce the unnecessary time cost. Secondly, for the WSC problem with QoS qualified solution, a QoS-aware service composition method named LOEM is put forward to seek for a best solution. On one hand, LOEM filters the numerous candidate services for each task into fewer ones, which decreases the time complexity significantly; on the other hand, as the derived candidate services are promising, a QoS near-to-optimal composite solution could be achieved by enumeration. The experiment results demonstrate that our proposed LOEM outperforms the Global and Hybrid methods, in terms of computation time significantly. Specifically, when dealing with certain WSC problems that require quick response, LOEM exhibits its advantages.

7.3 Further Discussions

In this paper, a QoS-aware composition method is investigated for supporting cross-platform service invocation in cloud environment. Concretely, a decision-making method is brought forth, to determine whether a QoS-aware WSC problem has a QoS qualified composite solution; subsequently, a QoS-aware service composition method named LOEM is put forward, aiming at improving the composition efficiency when a large number of composite solutions are available in cloud environment. Although the obtained composite solution may not be the QoS-optimal one, a QoS near-to-optimal composite solution is derived with less time cost than the traditional QoS-aware composition methods.

However, in this paper, there still exist several shortcomings. First of all, determining whether a QoS-aware WSC problem has a QoS qualified composite solution is essentially a 0-1 CSP, whose time complexity is exponential, so only partial QoS-aware WSC problems could be determined in polynomial time by our proposed decision-making method. Besides, the time complexity of our proposed LOEM is still exponential, which usually cannot deliver a satisfactory result if an end user requires a real time response. Therefore, it is still an open research problem, to develop a more efficient heuristic QoS-aware service composition method with polynomial time complexity. These more complex situations will be investigated in our future work.
8. Conclusions

Due to the nearly unlimited services available in cloud environment, there is usually a gap between the low service composition efficiency and the end users’ quick response requirements. In this paper, a QoS-aware composition method is investigated for supporting cross-platform service invocation in cloud environment. Concretely, a decision-making method is suggested, to determine whether a QoS-aware WSC problem has a QoS qualified composite solution, so as to avoid unnecessary time cost. Subsequently, a QoS-aware service composition method, i.e., LOEM is proposed, aiming at improving the composition efficiency when numerous composite solutions are available. Through an experiment, we also demonstrate the feasibility of LOEM in dealing with the QoS-aware WSC problems. This QoS-aware service composition method could also be helpful for building a flexible and scalable cloud platform, which will be investigated as our future research topic.

ACKNOWLEDGEMENTS
This paper is partly supported by the National Science Foundation of China under Grant No. 61021062, 61073032 and 60736015, and Jiangsu Provincial NSF Project under Grants BK2008017.

REFERENCES


