<table>
<thead>
<tr>
<th>Author</th>
<th>Zhang, Mingwei; Liu, Chengfei; Yu, Jian; Zhu, Zhiliang; Zhang, Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>A correlation context-aware approach for composite service selection</td>
</tr>
<tr>
<td>Year</td>
<td>2013</td>
</tr>
<tr>
<td>Journal</td>
<td>Concurrency Computation: Practice and Experience</td>
</tr>
<tr>
<td>Volume</td>
<td>25</td>
</tr>
<tr>
<td>Issue</td>
<td>13</td>
</tr>
<tr>
<td>Pages</td>
<td>1909-1927</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/1959.3/311374">http://hdl.handle.net/1959.3/311374</a></td>
</tr>
</tbody>
</table>

Copyright © 2013 John Wiley & Sons, Ltd. This is the peer reviewed version of the following article: Zhang, Mingwei; Liu, Chengfei; Yu, Jian; Zhu, Zhiliang and Zhang, Bin 2013, 'A correlation context-aware approach for composite service selection', Concurrency Computation: Practice and Experience, Vol. 25, no. 13 (Sep 2013), pp. 1909-1927, which has been published in final form at http://dx.doi.org/10.1002/cpe.2988. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Self-Archiving.

This is the author’s version of the work, posted here with the permission of the publisher for your personal use. No further distribution is permitted. You may also be able to access the published version from your library.

The definitive version is available at: http://dx.doi.org/10.1002/cpe.2988
A correlation context-aware approach for composite service selection

Mingwei Zhang1,*†, Chengfei Liu2, Jian Yu2, Zhiliang Zhu1 and Bin Zhang1

1Software College, Northeastern University, Shenyang, China
2Faculty of ICT, Swinburne University of Technology, Melbourne, Australia

SUMMARY

Composite service selection is one of the core research issues in Web service composition. Because of the complex service correlation context, candidate services may perform differently when being used with other services. Presently, most service selection approaches ignore this issue, which makes the selected composite services less efficient than expected. To solve this problem, a service correlation context-aware composite service selection approach is proposed on the basis of the concept of single-entry single-exit (SESE) region. The general process of our approach is as follows: (1) mining the SESE patterns that are frequently used together in the set of efficiently executed instances of a composite service; (2) dividing the process model of the composite service into SESE regions and generating the candidate SESE pattern set of each region, using the discovered SESE pattern set; and (3) optimizing composite service selection globally on the basis of QoS using divided regions as selection units and their candidate pattern sets as candidate service sets. Because SESE patterns are testified by large amount of efficiently executed instances, they have higher quality than the results of independent selection of services in an SESE region. Experimental results demonstrated that our approach can improve the quality of selected composite services effectively in the correlation context. Concurrency and Computation: Practice and Experience, 2012.© 2012 Wiley Periodicals, Inc.

Received 24 July 2012; Accepted 13 December 2012

KEY WORDS: Web services; composite service selection; service correlation; SESE patterns

1. INTRODUCTION

Along with the rapid development of service computing technologies, the number of available Web services is also on the rise. For example, there are more than 20,000 Web services registered on Seekda (http://webservices.seekda.com/). On the other hand, service composition as the main paradigm for developing service-oriented application is also gaining much attention [1]. Frequently, within a composite service, the functionality of a component service can be satisfied by more than one Web services. Consequently, how to select the optimal composite services among those providing the same functionality becomes an important research issue. [11–18]

Recently, there have been many research works about composite service selection, and most of them are QoS-based selection approaches. In theory, these approaches can select a composite service with best quality on the premise of meeting user’s constraints. In the selection process, they suppose that each candidate service is independent with other services in a composite service and the quality of the candidate service is fixed. However, in a real execution environment of composite services, there

*Correspondence to: Mingwei Zhang, Software College, Northeastern University, Shenyang 110819, China.
†E-mail: zhangmw@mail.neu.edu.cn

Copyright © 2012 John Wiley & Sons, Ltd.
may be complex correlations among candidate services that influence the quality of each other. For example, in a ‘Travel Planning’ composite service, the hotel service and the airline service in the same alliance may give a customer a combined better price if the customer chooses to use them together. Current QoS-based service selection approaches almost all ignore this issue, which makes the QoS attributes of candidate services used during selection deviate from real-time QoS attributes because of the existence of service correlation context. The consequence is that the information used in service selection is inaccurate and the selected composite services may not have the quality as expected.

Because of the complexity of the correlation context of a composite service, it is difficult to acquire its full information and to model its internal interaction in correlation context-aware service selection using a white box analysis approach. In this paper, we consider the correlation context for composite service selection using a black box analysis approach. In this approach, we do not care about the internal interaction of service correlation. Instead, we analyze the external behaviors that it may induce and their influences on composite service selection. Then, the related knowledge patterns are discovered to optimize service selection.

Because of the existence of the service correlation context, some candidate services become more efficient and some become less efficient if they are used together. The core idea of our approach is that the candidate service sets that are frequently used together in the set of efficiently executed instances, in the form of single-entry single-exit (SESE) patterns, are usually more efficient in the service correlation context, so applying them in composite service selection can improve the quality of the selected results. The general process of our approach is described as follows. First, the set of efficiently executed instances is extracted from the composite service log repository, and candidate services that are frequently used together are mined from it. On this basis, the composite service process model is divided into SESE regions [2, 3], and each region that has given amount of mined candidate service sets (called SESE patterns) can be chosen. Finally, regarding the divided SESE regions as selection units and the corresponding SESE patterns of each region as its candidate services, QoS-based global optimizing selection is carried out for the composite service. The main contribution of this paper includes the following. First, on the basis of analyzing the service correlation context existing in a composite service and its influences on service selection using an example, we propose a service correlation context-aware service selection approach. Second, we present a black box analysis method for conducting correlation context-aware composite service selection. In the selection process, considering the service correlation context by using historical information and adopting the techniques of SESE decomposition and pattern mining, this approach can improve the quality of the composite service selection effectively. Moreover, the proposed ‘SESE pattern’ concept can be applied to other optimization approaches for composite services, such as online adaptation and process model improvement.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the concept of service correlation context using an example and demonstrates our general research idea. Section 4 explains some core concepts used in our approach. Section 5 presents the whole process of the proposed correlation context-aware service selection approach in detail. Section 6 gives the experimental result, and finally we conclude our work in Section 7.

2. RELATED WORK

There have been some studies on Web service correlations in the service composition field. Kim and Jain [4] presented a task dependence-based approach for Web service composition driven by business rules. This research analyzed what kinds of dependence exist among the business rules and tasks and how dependence among business rules and tasks affects Web service composition. Xu et al. [5] analyzed semantic associations between Web services and proposed a dynamic semantic association-based service composition approach. Lin and Arpinar [6] presented a novel technique for discovering semantic relations between pre-condition and post-condition of different
services based on ontological descriptions and used pre-condition and post-condition of Web services to enable their automatic composition. In [7–9], the authors analyzed the Web service correlations from the view point of transactions, discussed transactional properties of composite services, and proposed approaches for increasing resource availability and flexible compensation. Kreshnik et al. [10] analyzed the correlation of messages and presented a delta algorithm to discover conversations and service protocols. Such Web service relations are part of the service correlation context. However, the research about correlation context-aware composite service selection is relatively new.

Composite service selection approaches have been extensively studied recently, and most of them are based on QoS. Zeng and Benatallah [11] presented a global planning approach to selecting an optimal execution plan by means of integer programming. Ardagna and Pernici [12] modeled the composite service selection as a mixed integer linear problem where both local and global constraints were taken into account. Yu and Lin [13] discussed selection algorithms for multiple QoS attributes using an increasing utility function of the QoS. Korkmaz and Krunz [14] used the theoretical properties of the nonlinear cost function and proposed a heuristic algorithm to minimize both the nonlinear cost function and the primary cost function. Canfora et al. [15] adopted a quite different strategy for optimal selection based on genetic algorithms. Zheng et al. [16] proposed a general solution to calculate the QoS for composite services with complex structures and showed that QoS-based service selection can be conducted on the basis of the proposed QoS calculation method. Godse et al. [17] outlined a method of monitoring and extrapolating service performance and used the same method for automated service selection. Xu and Jennings [18] presented a service selection algorithm that takes into account time-sensitive intra-provider and inter-provider discounts, and it can minimize the expected cost of the service aggregator by offering a composite service within a specified time interval. Although here we do not exhaust the body of work on service selection, to the best of our knowledge, none of them considered the existence of the service correlation context. They assume that the QoS of Web services is fixed and not influenced by other services. However, this assumption is not true in most cases.

In the Web service mining field, Dustdar et al. introduced the term of Web services interaction mining (WSIM) and presented three levels of abstraction on which WSIM could be performed: Web service operations, interactions [19], and workflows [20], namely mining respectively for only one single Web service, for one service and its neighbors, and for the entire workflow. Walid Gaaloul et al. [21, 22] proposed a set of mining techniques to discover composite service transactional behavior from an event-based log and used a set of rules to improve its reliability. Zheng et al. [23, 24] presented a service mining framework for exploring interesting compositions of existing Web services that might give some hints to composite service designers. The aforementioned researches mainly apply the mining results to process analysis and management. In this paper, the mining results are used in service correlation context-aware composite service selection where frequently used and tightly correlated concrete service patterns are discovered and used for service selection.

3. MOTIVATING EXAMPLE AND PROPOSED RESEARCH IDEAS

3.1. Motivating example

There are complex correlations among candidate services in a composite service. Because this complex service correlation context, some candidate services become more efficient and some become less efficient if they are used together. But, most present QoS based composite service selection approaches ignore this issue, so as not be able to improve the quality of selected composite services effectively.

To illustrate the above issue, let us have a look at a composite service example Travel Planning shown in Figure 1. The service is composed of the component service set \{WS_1, WS_2, WS_3, WS_4\}.
WS_5, WS_6, WS_7, WS_8}. In this composite service, tourist attraction searching can be carried out at the same time as airplane ticket reserving and hotel reserving, and attraction searching can be repeated until a satisfying result is reached. Then, the travel route between the selected attraction and the selected hotel is chosen. Finally, the distance of this route is calculated to determine whether to rent a taxi or a bicycle. To facilitate explaining, we assume that the candidate service set of WS_1 is \{American Airlines, United Airlines\}, the candidate service set of WS_2 is \{Hotel Pennsylvania, Wellington Hotel\}, and the candidate service set of WS_3 is \{Bank of America, Citibank\}.

First, the business logic of candidate services may depend on each other. In Figure 1, as a candidate service of WS_1, American Airlines makes a flexible pricing policy for sale promotion: a customer can have a cheaper airplane ticket if Visa card is used for payment, or a hotel belonging to Hotel Alliance is booked while reserving the airplane ticket. However, only some of the candidate services of the payment service WS_3 use Visa card. For example, Bank of America uses Visa card, but Citibank uses MasterCard. Similarly, only some of the candidate services of the hotel service WS_2 belong to Hotel Alliance. For example, Hotel Pennsylvania belongs to the Hotel Alliance but Wellington Hotel does not. Therefore, the price of the service American Airlines depends on the selection results of WS_2 and WS_3. In addition, a candidate service may change its policy such as security level or resource allocation priority according to whether another candidate service is used together. Clearly, there are complex business correlations among these candidate services.

Second, data passing may be needed between candidate services in one composite service. Considering the distance calculation service WS_6 in Figure 1, supposing that the required input ‘travel route’ of this service must be expressed in a given specific form, if some candidate services of the travel route selecting service WS_5 cannot produce the output data that meets this requirement, then extra work needs to be carried out such as adding a service to convert the format of the selected travel route. Consequently, the data passing between candidate services will no doubt make candidate services correlated with each other.

Last, the environments of candidate services may have mutual constraints. Considering one of the candidate payment services Citibank, because of network environment and others, the response time will be shorter and the operation will be more secure if the payment receiver is also a client of Citibank, and the situation will be the opposite if the receiver is not a client of Citibank. In this example, American Airlines and Hotel Pennsylvania are the clients of Bank of America. United Airlines and Wellington Hotel are the clients of Citibank. Therefore, the payment services Citibank will be more efficient if it is used together with the airplane service United Airlines and the hotel service Wellington Hotel than with other services. This situation will be more obvious for a time-sensitive composite service. Furthermore, considering an information management composite service, the performance of the data query service and the data processing service will inevitably correlate with the selection result of the data storage service. Consequently, there are complex environment correlations among candidate services in a composite service.
On the basis of the aforementioned discussion, it is obvious that there are complex correlations among candidate services on the business, data, and environment aspects, which makes some services become more efficient and some services less efficient if they are used together. On the basis of this observation, we conclude that it is necessary to conduct research into service correlation context-aware composite service selection in order to improve the performance of composite services.

3.2. Research ideas

Our approach to service correlation context-aware composite service selection is briefly described as follows. First, the composite service execution information is recorded into logs, and the process model of the composite service is decomposed to generate the process structure tree that uses SESE regions to represent the composite service. On this basis, SESE patterns frequently used in past efficiently executed instances of the composite service are mined. Finally, the discovered SESE patterns are used to divide the process model of the composite service into SESE regions, and optimal composite service selection is carried out regarding the divided regions as selection units.

As shown in Figure 2, the general process of our approach can be divided into four relatively independent stages: the log recording stage, the process structure tree generating stage, the SESE pattern mining stage, and the composite service selecting stage.

At the log recording stage, the service correlation context-aware composite service selection is carried out on the basis of previous execution experience of the composite service. Therefore, the service composition system needs to record its execution information into logs to be prepared to do follow-up mining.

At the process structure tree generating stage, the process model of the composite service is decomposed into canonical SESE regions that are organized into a nested process structure tree. This tree is the data structure basis of our approach. The SESE pattern mining stage is the core stage of the whole correlation context-aware selection process. On the basis of execution log repository and the process structure tree of the composite service, the frequently used SESE patterns (concrete service sets) are mined to do composite service selection in the service correlation context.

Finally, on the basis of the mined SESE patterns, the process of the composite service is divided into SESE regions, and each of them has more than a given amount of candidate SESE patterns that can be chosen. Regarding the divided regions as selection units and the corresponding SESE pattern sets as candidate service sets, correlation context-aware service selection can be carried out on the basis of QoS. Because SESE patterns are testified by many efficiently executed instances, their quality as selected composited services is usually higher than those selected services by doing selection independently for each service in the corresponding SESE region. As such, we can obtain a composite service with better QoS attributes.

In the next section, we explain some basic concepts used in our approach.
4. PRELIMINARIES

In our approach, SESE patterns are used in correlation context-aware selection, and SESE pattern mining is the most important step in the whole selection process. This section will first review the concepts of ‘composite service’ and ‘SESE region’ and then give the key concept of SESE pattern.

Definition 1 (Composite service)
A composite service is a directed graph \( G = (N, E) \), where a node \( n \in N \) is exactly one of the following: a start node, a stop node, a service, a fork, a join, a decision, or a merge, such that

1. there is exactly one start node and exactly one stop node; the start node has no incoming edges and exactly one outgoing edge, whereas the stop node has exactly one incoming edge but no outgoing edges;
2. each fork and each decision has exactly one incoming edge and two or more outgoing edges, whereas each join and each merge has exactly one outgoing edge and two or more incoming edges; each service has exactly one incoming and exactly one outgoing edge; and
3. each node \( n \in N \) is on a path from the start node to the stop node.

It follows from the definition that no node is directly connected to itself. According to the concept, the Travel Planning composite service is shown as Figure 1.

The correlation context-aware selection is carried out on the basis of the idea of dividing a composite service into many consecutive units, and each unit contains some component services. When selection is being performed, all component services in one unit are selected as a whole. For QoS-based global optimal selection algorithms, the QoS value of the composite service is calculated according to the QoS of each component service. In our approach, the divided units act as component service, so the composite service QoS is calculated on the basis of the QoS of the units. To facilitate calculation, each divided unit is limited to only have one entry edge and one exit edge, that is, SESE region. Here, we in fact make a trade-off between correlation context awareness and global optimal selection to obtain the best composite service instance in the service correlation context as much as possible.

Definition 2 (SESE region)
Let \( G = (N, E) \) be a composite service. An SESE region \( F = (N', E') \) is a nonempty subservice of \( G \), that is, \( N' \subseteq N \), \( E' = E \cap (N' \times N') \) such that there exist edges \( e, e' \in E \), with \( E \cap ((N \setminus N') \times N') = \{e\} \) and \( E \cap (N' \times (N \setminus N')) = \{e'\} \). \( e \) and \( e' \) are called the entry and the exit edge of \( F \), respectively.

It can be known from the aforementioned definition that each SESE region \( F \) of the composite service \( G \) only has one entry edge \( e \) and one exit edge \( e' \). In addition, the loop structure is considered in this definition, and every cycle in \( G \) containing \( e \) also contains \( e' \) and vice versa [2, 3]. SESE region is a concept at the process schema level composed of abstract services, whereas SESE pattern is a concept at the level of executed instances that are composed of concrete services. To simplify the discussion, we assume that all the composite services are acyclic when we define the concept of SESE pattern. If a composite service contains cycles, we replace all loop sequences with the most frequent service or service set. The related concepts of SESE pattern will be given next.

Definition 3 (Execution path)
An execution path of a composite service \( G \) is a sequence of abstract services \( P = <WS_1, WS_2, \ldots, WS_n> \), such that \( WS_1 \) is the initial service, \( WS_n \) is the final service, and for every service \( WS_i \) (1 < \( i < n \)),

1. \( WS_i \) is a direct successor of one of the services in the sequence \( <WS_1, \ldots, WS_{i-1}> \).
2. \( WS_i \) is not a direct successor of any of the services in the sequence \( <WS_{i+1}, \ldots, WS_n> \).
3. There is no service \( WS_j \) in \( <WS_1, \ldots, WS_{i-1}> \) such that \( WS_j \) and \( WS_i \) belong to two alternative branches of \( G \).
4. If \( WS_i \) belongs to one of the concurrent regions of \( G \), then all the services in this region are in the sequence \( <WS_1, WS_2, \ldots, WS_n> \).
This definition relies on the concept of a direct successor of a service. Roughly stated, an abstract service \( WS_j \) is a direct successor of another service \( WS_i \) if it is not necessary to execute other services to start the execution of \( WS_i \) after the execution of \( WS_j \) is finished. The literature [11] shows more details.

Definition 4 (Executed instance)
Given one execution of the composite service \( G, P = \langle WS_1, WS_2, \ldots, WS_n \rangle \) is its corresponding execution path. Then, for each abstract service \( WS_i \) in \( P \), there is a concrete service \( s_i \) executed successfully. All these concrete services constitute a sequence \( EI = \langle s_1, s_2, \ldots, s_n \rangle \), namely being an executed instance of the composite service \( G \).

Definition 5 (SESE pattern)
Given a composite service \( G = (N, E), S = \{ EI_1, EI_2, \ldots, EI_m \} \) is one executed instance set of \( G, F = (N', E') \) is an SESE region in \( G \), \( P = \langle WS_1, WS_2, \ldots, WS_n \rangle \) is one execution path of \( F \), and \( sp = \langle s_1, s_2, \ldots, s_k \rangle \) is the corresponding concrete service set of \( P \). The support of \( sp \) in the set \( S \) is defined as follows:

\[
support_s(sp) = \frac{|\{EI \mid EI \subseteq sp\}|}{|S|} (1 \leq l \leq m)
\]

\( support_s(sp) \) is the percentage of sequences containing \( sp \) in \( S \). Given the support threshold \( min_sup \in [0,1] \), if the support of \( sp \) is greater than \( min_sup \), then the concrete service set \( sp \) is a corresponding SESE pattern of the SESE region \( F \).

Generally speaking, an SESE pattern is a set of concrete services that are frequently executed together. Moreover, its corresponding abstract services must constitute an SESE region. The idea of our correlation context-aware selection is that those services in a frequently executed concrete service set appearing in efficiently executed instances are more context correlated and consequently a composite service that matches this kind of frequently executed concrete service sets is more likely to perform efficiently because it has been testified by many executed instances.

The discovered SESE patterns from executed instances can be applied to service selection. First, a composite service process is divided into several max frequent SESE regions, and each such region has a certain amount of candidate SESE patterns. Then, the composite service selection can be carried out regarding the max frequent SESE regions as selection units and their corresponding SESE patterns as candidate services. A max frequent SESE region is defined as follows.

Definition 6 (Max frequent SESE region)
Given a composite service \( G = (N, E), F = (N', E') \) is an SESE region in \( G \). If \( F \) has \( n \) candidate SESE patterns, where \( n \geq min_num \) and \( min_num \) is the threshold for the minimum number of patterns, then it is called frequent SESE region. If none of the SESE regions containing \( F \) is frequent SESE region, then \( F \) is a max frequent SESE region in \( G \).

5. COMPOSITE SERVICE SELECTION BASED ON SESE PATTERNS

In this section, the concrete techniques used in each stage of our approach are given in detail.

5.1. Log recording
The correlation context-aware selection is carried out on the basis of the past execution information of the composite service in our work. So, a log recording architecture for the service composition system is needed to be able to record the execution information. In addition, the contents of the log should be analyzed for the purpose of SESE pattern mining.

For the research of Web service logging, there have been a few works. Zheng et al. [16] proposed a classification of Web service logs according to their contained information and analyzed how to...
acquire the corresponding information for each log level. Cruz et al. [25] presented a Web services logging architecture WSLogA based on SOAP intermediaries, which captures comprehensive services usage information. Ringelstein and Staab [26] presented a solution, in the form of an architecture, a formalization and an implemented prototype for logging and collecting logs in service-oriented and cross-organization systems. But, it is difficult to record process information of composite services for all the aforementioned works. In our work, we choose to record logs on the client side. By embedding log recording function into composite service execution engine (such as ServiceMix, BPMS, and ActiveBPEL), we can easily log a wealth of execution information of composite services to support monitoring Web service usage, evaluating quality of Web services, analyzing commercial activities, and others. The composite service execution information is used to do SESE pattern mining in this paper. For this purpose, the information needed to be recorded for each SOAP message is given in the following definition.

**Definition 7 (SLI: SOAP log item)**
This is a tuple recorded by the log recorder for each SOAP message, represented as \(<\text{ProcessID}, \text{InstanceID}, \text{ServiceID}, \text{Type}, \text{Time}, \text{Status}>\). The meaning of each item is described as follows:

- **ProcessID**: It identifies the composite service or the business process.
- **InstanceID**: It identifies an instance of the composite service that produces the log item.
- **ServiceID**: It specifies the concrete service that sends the request/response message. Each Web service can be identified by the tuple \(<\text{Url}, \text{portType}, \text{operation}>\) and can be obtained from its WSDL file.
- **Type**: It specifies the type of the SOAP message, that is, request message or response message.
- **Time**: It specifies the sending time of SOAP request/response messages.
- **Status**: It specifies whether the request or response is successful.

Using the SLI contents in Definition 7, we can extract the composite service executed instance set and prepare for pattern mining. Here, it needs to be emphasized that the contents recorded can be customized and extended to fit new requirements.

### 5.2. Process structure tree generating

One composite service can be decomposed into many SESE regions. To facilitate the management of them, canonical SESE regions are defined and organized into a program structure tree [2]. We call it a process structure tree of a composite service in this paper. Considering the decomposition example of the Travel Planning composite service shown in Figure 3, it has more regions than those that are shown explicitly. For example, the union of regions \(H \cup I\), denoted as \(H \cup I\), as well as \(G \cup H \cup I \cup L\) regions. They are, however, not canonical SESE regions.

The regions shown in Figure 3 are exactly the canonical SESE regions of the Travel Planning composite service. Canonical regions do not overlap, and two canonical regions are either nested or disjoint [2]. Therefore, it is possible to organize the canonical regions in a unique tree, that is, the process structure tree of the composite service. It is worth noting that a process structure tree can be computed in linear time [2].

Figure 4 is the structure tree of the process in Figure 3. The region composed of sequential child regions is called sequential SESE region. Because sequential SESE regions may include noncanonical SESE regions, which may contain SESE patterns, extra work is needed to mine SESE patterns from them. In the next stage, SESE patterns are mined on the basis of the recorded logs and the constructed process structure tree.

### 5.3. SESE pattern mining

A SESE pattern is a set of concrete services that are frequently executed together in the set of efficiently executed instances of a composite service. There are two steps to do SESE pattern mining. First, the set of executed instances of the composite service is extracted from the
recorded log repository and selected on the basis of their performance. Second, SESE patterns are mined on the basis of the extracted instance set.

In the first step, on the basis of the SLIs as defined in Definition, the process for extracting executed instance set can be generally described as follows: (1) for each executed instance whose QoS is higher than the threshold, select all SLIs of this instance by its InstanceID; (2) on the basis of the selected SLIs, generate a requested service sequence according the requesting time; (3) for the generated sequence, check the Status of all services in it, delete the services requested or responded unsuccessfully, and replace all service loops by the most frequent services; and (4) put the processed sequences into the sequence table, and the generated sequence table in the end returns the set of executed instances of the composite service. Because the extracting algorithm is straightforward, we will not unfold its detail here. Next, we will focus on the second mining step, which represents the core idea of our approach.

We use the idea of association mining in transaction database to discover SESE patterns. In the following, we first explain some key concepts in association mining. Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items and $D$ the set of transactions called the transaction database. Each transaction $T$ in $D$ is a set of items, such that $T \subseteq I$. We say that $T$ contains the item-set $A$ if $A \subseteq T$. The item-set that contains $k$ items is called $k$ item-set. The support count of the item-set $A$ with respect to $D$ is the number of transactions that contain $A$, denoted as $\sigma(A)$. If $\sigma(A) > min\_sup \times |D|$ ($min\_sup$ is the minimum support threshold and $|D|$ is the number of transactions in $D$), then it is called a frequent item-set. The set of frequent $k$ item-sets is denoted as $L_k$. 

![Figure 3. Decomposition of the Travel Planning composite service into canonical single-entry single-exit regions.](image)

![Figure 4. The process structure tree of the Travel Planning composite service.](image)
Given the composite service \( G = (N, E) \), the concrete service sequence \( EI = \langle s_{1.1}, s_{1.2}, \ldots, s_{1.n} \rangle \) is an executed instance of \( G \). \( S = \{ EI_1, EI_2, \ldots, EI_m \} \) is an executed instance set extracted for \( G \). Regarding \( EI_i \) as an item-set and \( S \) as a transaction database, each SESE pattern of \( G \) must be a frequent item-set of \( S \).

The idea of the SESE pattern mining algorithm can be generally described as follows: In the executed instance set \( S \), an iterative method called layer searching is used to discover \( k+1 \) item-sets based on \( k \) item-sets. At first, all the frequent 1 item-sets are discovered, denoted as \( L_1 \), then the set of frequent 2 item-sets \( L_2 \) is discovered based on \( L_1 \), and \( L_3 \) is discovered based on \( L_2 \); this process stops until \( L_k \) is empty.

The discovering process of \( k \) item-sets \( L_k \) from \( k-1 \) item-sets \( L_{k-1} \) has two steps: ‘connection’ and ‘pruning’. The connection step generates candidate item-sets, and the pruning step determines the frequent item-sets. In the connection step, to discover \( L_k \), \( L_{k-1} \) can be connected with itself to generate all candidate \( k \) item-sets, denoted as \( C_k \). Two item-sets \( l_i \) and \( l_j \) in \( L_{k-1} \) can be connected if they only contain one different item, that is, one different concrete service. \( C_k \) is the superset of \( L_k \), namely not every item-set in \( C_k \) is frequent. The pruning step at first prunes \( C_k \) and then scans the transaction database to determine frequent item-sets \( L_k \) among the candidates. Pruning can reduce the required computation cost to a certain extent and improve this algorithm’s performance. There are two pruning rules. The first is called Apriori nature [27], in which any frequent item-set is impossible to have an infrequent sub-item-set. The second is called SESE nature, in which the \( k \) Web services contained in any discovered \( k \) item-set must constitute an SESE region, which can be decided using the composite service process structure tree as shown in Figure 4. All the candidates that do not satisfy the two presented rules in \( C_k \) can be pruned. The SESE pattern mining algorithm SESEPM is described in the following text.

**Algorithm: SESEPM**

**Input:** executed instance set \( S \) of the composite service \( G \), the process structure tree \( T \) of \( G \), the minimal support threshold \( \text{min}_\text{sup} \)

**Output:** all the SESE patterns of \( G \)

\( L_1 = \text{frequency}(S); \) // finding the frequent 1 item-sets in \( S \)

for ( \( k = 2 \); \( L_{k-1} \neq \emptyset \); \( k++ \) ) {
  \( C_k = \text{AprioriGen}(L_{k-1}); \) // generating new candidate item-sets \( C_k \)
  AprioriDelete(\( C_k, L_{k-1} \)); // pruning the item-sets violating the Apriori nature in \( C_k \)
  SESEDelete(\( C_k, T \)); // pruning the item-sets violating the SESE nature in \( C_k \)
  for ( \( EI \in S \) ) { // for each executed instance \( EI \) in \( S \)
    \( C_{EI} = \text{subset}(C_k, EI); \) // finding the candidate sets \( C_{EI} \) among \( C_k \) contained in \( EI \)
    // updating the support for each candidate item-set in \( C_{EI} \)
    for ( \( c \in C_{EI} \) ) c.UpdateSupport();
  }
  \( L_k = \{ c \in C_k | c.\text{support} > \text{min}_\text{sup} \} \);
  \}

return \( L = \cup_k L_k \); // function AprioriGen is to generate new candidate sets

procedure AprioriGen(\( L_{k-1} \)) {
  \( C_k = \emptyset; \) // initializing \( C_k \)
  for ( \( \forall l_i \in L_{k-1} \) ) {
    for ( \( \forall l_j \in L_{k-1} \) ) {
      if ( (l_i[1] = l_j[1]) && (l_i[2] = l_j[2]) && \ldots && (l_i[k-2] = l_j[k-2]) && (l_i[k-1] \neq l_j[k-1]) ) {
        c = l_i \cup l_j; // connecting the item-set \( l_i \) and \( l_j \) that satisfy the connection condition
        \( C_k = C_k \cup c; \) // putting \( c \) into the new candidate sets \( C_k \)
      }
    }
  }
  return \( C_k \);
}

Using the idea of the classical algorithm AprioriGen, algorithm SESEPM can discover all SESE patterns of the composite service \( G \) in the executed instance set \( S \). Each mined pattern is a set of concrete services that are frequently executed together in \( S \).
5.4. Composite service selection based on SESE patterns

Given a composite service $G$, suppose $SP = \{sp_1, sp_2, \ldots, sp_n\}$ is the SESE pattern set of $G$ mined from the executed instance set $S$. Now, we apply $SP$ to do the correlation context-aware selection of $G$ in two steps: (1) dividing $G$ based on $SP$ and (2) regarding the divided SESE regions as selection units to optimize service selection.

(1) Dividing composite service based on SESE patterns

The division of the composite service process is carried out on the basis of the concept of ‘max frequent SESE region’ defined in Section 3.2. Any divided SESE region must have a certain number of candidate SESE patterns and, at the same time, should be the max region that matches the first condition. Conceptually speaking, an SESE pattern ensures the non-occasionality of the integrated service has executed instance set $T$.

For each SESE pattern $sp_i$ in $SP$, the candidate pattern count and candidate pattern set of each node in $G$’s process structure tree $T$ are updated, and the last generated max frequent SESE regions and their candidate patterns are the division results for $G$. It should be emphasized that sequential canonical regions in $T$ needs special treatment in this division process. One sequential canonical region having more than three child nodes can be divided into more than one SESE regions that are not contained in $T$, whereas these regions may be max frequent regions and become the final division results of the composite service.

For each SESE pattern $sp_i=<s_{i,1}, s_{i,2}, \ldots, s_{i,k}>$ in $SP$, the updating rules and method of the candidate pattern count and candidate pattern set of each node in $T$ will be given. To facilitate explanation, we suppose that each abstract service $WS_i$ ($1 \leq i \leq 8$) in the Travel Planning composite service has five candidate services $ws_{ij}$ ($1 \leq j \leq 5$). The SESE patterns listed in Table I and the $T_1$ process structure tree shown in Figure 4 are used as examples.

1. Each sequential canonical region having more than three child regions in the process structure tree $T$ is extended, and all SESE regions of the composite service $G$ are represented in the extended tree $T'$. The extended tree of the process structure tree in Figure 4 is shown in Figure 5.

2. For all nodes contained in the pattern $sp_i$, the candidate pattern count and candidate pattern set are updated. For example, the nodes requiring update are A, B, C, D, and E for the pattern 1 in Table I, and the nodes requiring update are I, J, K, M, <I, J>, <I, J, M>, and <I, J, K> for the pattern 4.

3. For each node contained in the pattern $sp_i$, if $sp_i$ provides new candidate pattern for it, then its candidate pattern set should be updated. For the nodes not containing any child alternative region, their candidate pattern count equals $n + 1$. For the other nodes, their candidate pattern count equals the minimum count of their contained alternative paths. Taking the example of the SESE pattern set listed in Table I, for the node $E$ that does not contain any child alternative region, its candidate pattern set is $\{<w_{s11}, w_{s22}, w_{s33}>$, $<w_{s14}, w_{s22}, w_{s35}>$, $<w_{s13}, w_{s21}, w_{s35}>, <w_{s22}, w_{s33}>\}$, and its candidate pattern count equals 3. For the node <I, J, M> containing some alternative regions, its candidate pattern set is $\{<w_{s51}, w_{s63}, w_{s72}>, <w_{s53}, w_{s62}, w_{s71}>, <w_{s52}, w_{s63}, w_{s85}>\}$. There are two alternative paths in the node <I, J, M>, such as <I, J, K> and <I, J, L>. Their candidate pattern counts respectively equal 2 and 1, so the count of the node <I, J, M> equals 1.

<table>
<thead>
<tr>
<th>ID</th>
<th>SESE pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${&lt;w_{s11}, w_{s22}, w_{s33}&gt;}$</td>
</tr>
<tr>
<td>2</td>
<td>${&lt;w_{s11}, w_{s22}, w_{s35}, w_{s43}, w_{s51}, w_{s63}, w_{s72}&gt;}$</td>
</tr>
<tr>
<td>3</td>
<td>${&lt;w_{s13}, w_{s21}, w_{s35}, w_{s44}&gt;}$</td>
</tr>
<tr>
<td>4</td>
<td>${&lt;w_{s53}, w_{s62}, w_{s71}&gt;}$</td>
</tr>
<tr>
<td>5</td>
<td>${&lt;w_{s52}, w_{s63}, w_{s85}&gt;}$</td>
</tr>
</tbody>
</table>
4. In the extended tree \( T_0 \), all the max frequent SESE regions (i.e., their candidate pattern count \( n \geq \min_num \) and not contained in other frequent regions) need to be found, and at the same time, metaregions that only contain one Web service should be deleted. Consider the example shown in Table I, let the minimum pattern count threshold \( \min_num \) equal 3, then there are two max frequent SESE regions: \( E \) and \( \langle I, J \rangle \), as shown in Figure 5.

5. Because of the presence of sequential canonical regions in the extended tree \( T_0 \), there are possibly some overlapped nodes among the discovered max frequent SESE regions. For example, the nodes \( \langle H, I, J \rangle \) and \( \langle I, J, M \rangle \) in Figure 4 may be all max frequent regions. Consequently, the repeated nodes need to be removed before reaching the final division results of the composite service \( G \). The removing rule is that the max frequent regions satisfying the following three conditions are preserved: (1) the candidate pattern count is larger; (2) the region contains more items (Web services); and (3) the position code is smaller in the tree’s preorder traversal. At the same time, the repeated nodes are removed from the regions overlapping with the preferential region.

6. All the frequent SESE regions produced by the aforementioned steps are the final division results for the composite service \( G \). The division results of the Travel Planning composite service based on the SESE pattern set listed in Table I are shown in Figure 6.

The composite service division algorithm CSDA is given in the following text.

**Algorithm: CSDA**

**Input:** the SESE pattern Set \( SP \) of the composite service \( G \), the process structure tree \( T \) of \( G \), the minimum pattern count threshold \( \min_num \).

**Output:** the set of frequent SESE regions \( R \) forming the final division to \( G \)

\[
T' = \text{Extend}(T); \quad \text{// generating the extended tree } T' \\
\text{for } (\forall sp_i \in SP)\{
    N = \text{Included}(sp_i, T'); \quad \text{// finding all nodes in } T' \text{ contained by } sp_i \\
    \text{for } (\forall n_i \in N) \text{ Update}(n_i); \quad \text{// updating the candidate pattern set and pattern count for each node}
\}
\]

\[
R = \text{FMFR}(T') \quad \text{// finding all max frequent SESE regions in } T'
\]

\[
\text{if } (\text{Overlapped}(R))
\]

\[
R = \text{RemoveRepeated}(R); \quad \text{// removing the repeated nodes}
\]

Return \( R \);

The division algorithm is described at a high level, and the related details in each function are not given. Suppose that the number of patterns contained in \( SP \) is \( n \), for the number of nodes in the extended tree \( T' \) is fixed, the time complexity of algorithm CSDA is \( O(n) \). Using algorithm CSDA,
the frequent SESE regions that form a final division to the composite service $G$ and their candidate pattern sets can be mined to do correlation context aware selection.

(2) Selecting composite service based on SESE patterns

In the process of correlation context-aware service selection, all candidate SESE patterns of each SESE region need to calculate their QoS first. Then, regarding the divided frequent regions as component services, and their candidate pattern sets as candidate service set, composite service selection can be carried out using the existing QoS-based global optimizing selection algorithms. In the selection process, for the Web services not included in any frequent region, their candidate service sets remain unchanged.

The objective of composite service selection is to maximize the composite service QoS on the premise of meeting certain QoS constraints. Consequently, the calculation method of composite service QoS is the basis of service selection. For our division-based composite service selection algorithm, the QoS values of each SESE patterns need to be calculated on the basis of the QoS of related component services, further to calculate the QoS of the composite service.

Given a candidate pattern $sp$ of the composite service region $r$, if the values of its related QoS attributes are $\{ Q_1 = q_1, Q_2 = q_2, \ldots, Q_n = q_n \}$, then the QoS of $sp$ denoted as $QoS_{sp}$ equals $\sum_{i=1}^{n} w_i \times f_i(q_i)$. Here, $w_i$ and $f_i(q_i)$ can be defined by customers. $w_i$ is the weight of the service QoS attribute $Q_i$, and $f_i(q_i)$ is a customer’s describing function to $Q_i$. The quality $QoS_{CS}$ of the composite service $CS$ can be calculated in the same way. The techniques of service QoS aggregation have been extensively studied, and for details, please refer to [28, 29].

6. EXPERIMENTS

In this section, we experimentally evaluate our selection approach, including the execution time, the quality of selected composite services, and the effectiveness of the mined knowledge.

6.1. The selection of composite service selection algorithms

(1) Integer programming selection approach based on SESE patterns

First, the integer programming selection approach based on SESE patterns will be given. For the composite service composed by $m$ regions $\{r_1, r_2, \ldots, r_m\}$, suppose that the region $r_i$ has $n_i$ candidate patterns $\{sp_{i,1}, sp_{i,2}, \ldots, sp_{i,n_i}\}$, then the service selection problem can be modeled as

$$
x_{ij} = \begin{cases} 
1 & \text{if } sp_{ij} \text{ is selected} \\
0 & \text{otherwise} 
\end{cases} \tag{2}
$$
Expression (3) ensures that there is only one candidate SESE pattern selected for each region, and expression (4) maximizes the QoS of the composite service, that is, to satisfy user requirements to the greatest extent.

For the composite service selection problem with multiple QoS constraints, each QoS constraint can be represented by the integer programming expression (5), and the problem can be solved using integer programming approach.

\[
\sum_{i=1}^{m} \sum_{j=1}^{n_i} q_{ij} x_{ij} \leq Q^l_c
\]

Here, \( l = 1, \ldots, L \), \( L \) is the number of service quality constraints, \( m \) is the number of regions contained in the composite service, \( n_i \) is the number of candidate patterns of the \( i \)th region, \( q_{ij} \) is the value of the candidate pattern \( sp_{ij} \) on the service quality attribute \( l \), and \( Q^l_c \) is the constraint condition on the quality attribute \( l \). Expression (5) shows that the QoS aggregation of the selected results on \( l \) must be smaller than \( Q^l_c \). For different service quality attributes, the relational operator in the integer programming expression may be ‘\( \leq \)’, ‘\( = \)’, ‘\( \geq \)’, and so on. For example, the operator for ‘response time’ is ‘\( \leq \)’, and for ‘availability’ is ‘\( \geq \)’. After these integer programming expressions are given, division-based composite service selection can be carried out by the integer programming approach [11].

(2) Heuristic selection approach based on SESE patterns

This approach can rapidly discover feasible SESE pattern for each composite service region and optimizes the composite service QoS iteratively in polynomial time. Its core idea is that on the basis of the objective of service selection, the initial selection solution is given for each composite service region, and the current solution is replaced with new solution iteratively on the heuristics of existing selection results, until feasible SESE pattern set meeting user requirements is obtained. There are mainly three steps:

1. finding initial feasible SESE pattern set;
2. optimizing the composite service QoS on the premise of violating no constraint; and
3. optimizing the composite service QoS on the premise of violating some constraints and adjusting the selection result to meet the constraints.

The second step may only find locally optimal solution, and the third step breaks the local optimal limit to find the globally optimal solution [14].

6.2. The generation of experimental data

The Travel Planning composite service shown in Figure 3 is used to experimentally analyze our service selection approach. We develop and deploy this service adopting Apache ServiceMix technology and record related logs for generating the simulation data.

To do experiments effectively, 100 concrete services are generated for each of the eight abstract services contained in the Travel Planning service. Then, for each concrete service, the QoS attributes (cost, response time, and reputation) are selected. In practice, a better service is usually more expensive, but it is difficult to obtain a perfect method to generate such realistic QoS data. So, the QoS values of each candidate service are just generated within a certain
interval using normal distribution. For example, the cost of airplane service is between $500 and $2000, the response time between 2 and 10 s, and the reputation between 20% and 100%. The detailed QoS generating method that considers the Web service QoS correlation is described as follows:

1. selecting a certain proportion of concrete services randomly (initially 50%) as correlation-supported services and generating two values for each of their QoS attribute; for other concrete services, only generating one value for each attribute; and
2. for each correlation-supported concrete service $s$ randomly choosing one or two other abstract services. For example, for the payment service, choosing the hotel service or the airplane service or both of them;
3. randomly selecting 10 to 20 candidate services for each chosen abstract service; and
4. taking a better value as the correlative QoS value depending on the candidate services selected in step 2.b and the other one as default value.

The aforementioned generated service QoS values set a basis for service selection. After generating concrete Web services, the set of efficiently executed instances of the Travel Planning composite service need to be produced for SESE pattern mining. In real situation, the QoS correlation knowledge of a composite service may come from customers or from QoS-based filtering from the set of executed instances of the composite service. In our experiment, efficiently executed instances are generated as follows. First, classical selection algorithms are used to select services on the basis of default QoS values of concrete services, and then the selection results are optimized randomly on the basis of the generated correlative QoS data. During mining, the generated executed instances need to be filtered according to the real QoS of the composite service, and those with higher QoS values are selected for SESE pattern mining. The real QoS value of the composite service can be calculated on the basis of the real QoS of concrete services, which can be determined by checking whether they are correlated with other selected services.

### 6.3. Measuring computation cost

The first experiment aims at comparing the four composite service selection algorithms on computation cost: the general integer programming algorithm IP, the general heuristic algorithm HEU, the SESE pattern-based integer programming algorithm SESE_IP, and the SESE pattern-based heuristic algorithm SESE_HEU. For all of them, we study the effect of the number of candidate services of each abstract service on the selection cost (the time used to select composite service).

Before the experiment, 2158 items are selected first from 5000 executed instances generated using the approach discussed in the aforementioned subsection. After that, SESE pattern mining is conducted, generating two SESE regions as shown in Figure 6. The time spent for mining process is 173.2 s.

The result of service selection cost is shown in Figure 7. For each test case, we execute the composite service selection 10 times and compute the average time cost.

Figure 7 shows that on computation cost, the heuristic algorithm HEU is better than the integer programming algorithm IP, and in addition, SESE pattern-based service selection algorithms are better than their corresponding HEU and IP algorithms because the number of abstract services and the number of candidate services are both reduced by using SESE patterns. It is worth noting that SESE pattern-based algorithms are based on additional mining process, which also incurs cost. But a mining result can be reused for several service selections.

### 6.4. Measuring quality of composite services

Besides computation cost, the quality of the selected composite service is also an important indicator to evaluate service selection algorithms. Our second experiment is conducted to study the composite service QoS selected by the four algorithms: IP, HEU, DP_IP, and DP_HEU. Figure 8 presents the quality of selected composite services, where we vary the proportion of the correlative services. The QoS values shown in Figure 8 are calculated by normalizing the three QoS attributes, cost, response time, and reputation, which have the same weight. We use 5000 executed instances to do this
experiment. The SESE_IP and SESE_HEU curves shown in Figure 8 are the selection results based on mined SESE patterns.

In our experiments, the better one of the two randomly generated QoS values acts as the correlative QoS. So, the composite service QoS of all algorithms is improved along with the increase of the proportion of correlation-supported services. Figure 8 shows that from the quality of the selected composite services, algorithm IP is better than algorithm HEU. However, with the increase of the proportion of correlat supported services, there is no significant increase in QoS for both of the algorithms. In contrast, for SESE pattern-based service selection algorithms, although the QoS of the selected services is slightly lower than their HEU and IP algorithms at the beginning, its advantage is becoming more and more significant with the increase of the proportion of the correlative-supported services.

6.5. Measuring the effectiveness of the mined knowledge

(1) Measuring the relation between composite service QoS and log amount

The relation between the selection effectiveness of SESE patterns and the used log amount is analyzed in the next experiment. The effectiveness of SESE patterns can be measured by the QoS of selected composite services. To do this experiment, certain amount of log data are selected randomly from the initially generated 5000 executed instances, and then filtering and mining are carried out repeatedly. The composite service QoS selected based on the mined SESE patterns is listed in Table II. The proportion of correlation-supported services equals to 30% in the experiment.

When the proportion of correlative QoS is 30%, the selected composite service QoS of the original selection algorithms IP and HEU is respectively 0.657 and 0.59. It can be seen from the table that when 500 executed instances are adopted to do mining, the composite services selected by the SESE pattern-based selection algorithms SESE_IP and SESE_HEU have similar quality with the ones selected by the corresponding IP and HEU algorithms. Consequently, our correlation context-aware selection
approach shows no advantage when the log amount is less than 0.5K. Furthermore, it can be known from Table II that along with the increase of the number of adopted executed instances, the QoS of selected composite services also becomes higher, which means the mined SESE patterns become more effective. When the adopted log amount is larger than 1K, the algorithms based on SESE patterns are obviously better than their IP and HEU algorithms.

(2) Measuring the composite service QoS on given QoS correlation change rates

In our approach, the knowledge patterns mined can be reused to do several service selections. However, Web services are dynamic, and the effectiveness of the mined knowledge needs to be verified as time goes on. The next experiment mainly analyzes the relation between the effectiveness of the mined knowledge and the change rate of QoS correlations. The method of this experiment is that for each correlation-supported services $s$ generated, a certain proportion of its associated concrete services is changed randomly, and the composite service QoS selected based on the present SESE patterns is calculated again. This experiment is performed 10 times. The selected log amount is 5K, and the proportion of correlation-supported services is 30%. The average composite service qualities on given QoS correlation change rates are listed in Table III.

As time goes on, the correlations among Web service QoS may change, so that the selection effectiveness of the mined knowledge may decrease. Simulation results are given in Table III. From this table, we can see that when the change rate of QoS correlations reaches 10%, the composite service QoS selected based on present SESE patterns is similar to the general IP and HEU algorithms. Consequently, when the correlation among Web service QoS changes to a certain extent (10% for our experimental dataset), the mining process needs to be carried out again for maintaining the performance of correlation context-aware selection.

From all the aforementioned experiments, we can reach the conclusion that when there are complex correlations among Web services, the SESE pattern-based service selection approach proposed in this paper can shorten selection time and improve the quality of selected composite services. Certainly, we can also find that our approach needs initial accumulation of logs and periodical pattern mining. These are the limitations of our approach.

7. CONCLUSION

In this paper, we have studied the problem of selecting efficient composite services by taking the complex correlations among component services into account. A correlation context-aware approach for composite service selection has been proposed. In our approach, correlation context information of a composite service is first collected from those past efficiently executed instances of the composite service in a mining process and organized as SESE patterns. Then, composite service selection is conducted on the basis of the mined SESE patterns. A liner programming selection approach and a heuristic selection approach have been devised with experimental evaluation. The

<table>
<thead>
<tr>
<th>Data_Amount (K)</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoSDIV_IP</td>
<td>0.649</td>
<td>0.772</td>
<td>0.841</td>
<td>0.862</td>
<td>0.866</td>
<td>0.868</td>
</tr>
<tr>
<td>QoSDIV_HEU</td>
<td>0.593</td>
<td>0.703</td>
<td>0.759</td>
<td>0.786</td>
<td>0.789</td>
<td>0.791</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change_Rate (%)</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoSDIV_IP</td>
<td>0.868</td>
<td>0.735</td>
<td>0.659</td>
<td>0.613</td>
<td>0.592</td>
<td>0.575</td>
<td>0.562</td>
</tr>
<tr>
<td>QoSDIV_HEU</td>
<td>0.791</td>
<td>0.667</td>
<td>0.593</td>
<td>0.565</td>
<td>0.541</td>
<td>0.534</td>
<td>0.527</td>
</tr>
</tbody>
</table>
experimental results have shown that our correlation context-aware approaches for composite service selection outperform the general approaches in terms of computation cost and quality of selected composite services.

ACKNOWLEDGEMENTS
This work is supported by the National Natural Science Foundation of China under grant nos. 60773218, 60903009, 61100267, 60903009, 60903008, 61100028, and 61100090, the Australian Research Council under Linkage grant LP0990393, the National High Technology Research and Development 863 Program of China under grant no. 2009AA01Z122, and the Fundamental Research Funds of China for the Central Universities under grant nos. N110417001 and N110417004.

REFERENCES

