Reliable Service Composition via Automatic QoS Prediction

Hongbing Wang, Haixia Sun  
School of Computer Science and Engineering  
Southeast University  
Nanjing, China  
Email: hbw@seu.edu.cn

Qi Yu  
College of Computing and Information Sciences  
Rochester Institute of Technology  
Rochester, NY, USA  
Email: qi.yu@rit.edu

Abstract—Service composition has received considerable attention nowadays as a key technology to deliver desired business logic by directly aggregating existing Web services. Considering the dynamic and autonomous nature of Web services, building high-quality software systems by composing third-party services faces novel challenges. As a solution, new techniques have been recently developed to automatically predict the QoS of services in a future time and the prediction result will facilitate in selecting individual services. Nonetheless, limited effort has been devoted to QoS prediction for service composition. To fill out this technical gap, we propose a novel model in this paper that integrates QoS prediction with service composition. The integrated model will lead to a composition result that is not only able to fulfill user requirement during the composition time but also expected to maintain the desired QoS in future. As user requirement is expected to be satisfied by the composition result for a long period of time, significant effort can be reduced for re-composing newly selected services, which usually incurs high cost. We conduct experiments on both real and synthetic QoS datasets to demonstrate the effectiveness of the proposed approach.

Keywords—Service composition; QoS prediction; Service selection

I. INTRODUCTION

New software techniques are continuously under development for building modern software systems, which have become increasingly complicated and large-scale. Service-oriented computing (SOC) based technologies, which advocate the development of new software and services on the basis of the existing ones, have received significant attention and wide adoption in business, government, and a lot of other domains [1]. Service composition, as one of the key techniques in SOC, aims to aggregate existing services to deliver new functionalities required by users. Services are typically offered by third-party providers from different enterprises or organizations that conduct service implementations, supply service descriptions, and provide related technical and business support [2]. As different services may be implemented using distinct technologies, deployed on different platforms, and delivered over different communication links, services that provide similar functionalities may vary significantly in the quality of service or QoS they offer. As a result, QoS-aware service composition becomes an important research area that has received considerable attention in service computing community [3], [4].

Figure 1 shows a service composition workflow generated according to the user's request and the alternative service candidates to realize each task in the workflow, which implement the same function but differ in QoS. The goal of QoS-aware service composition is to select a service for each task set such that the aggregated QoS values satisfy the user's end-to-end QoS requirements [5]. The most straightforward way is to generate and evaluate all possible composition results, which is certain to find the optimal composition result. However, performing an exhaustive search is usually infeasible, especially when the number of existing services is large. On the one hand, as some services are not free to invoke, trying all possible compositions may incur high financial cost. On the other hand, the time complexity of an exhaustive search is exponential, which introduces high computational cost.

Existing works employ different optimization techniques, such as linear programming [4], dynamic programming [3], genetic algorithms [6], and so on to avoid an exhaustive search so that efficient QoS-aware composition can be achieved. Nonetheless, the composition cost may still incur high cost especially when scaling to a massive number of candidate services. The dynamic nature of the QoS proper-
ties further complicates the problem. It is not uncommon for the QoS of a selected service to change soon after the service composition is formed. The change may be introduced by the hardware, platform, software change of the service provider as well as the fluctuation of the communication links. As a result, the newly formed composition may no longer meet the user’s QoS requirement and hence need to be re-composed. This will obviously introduce significant overhead for building complex software systems.

To attack the above central challenges, we develop a novel service composition model that integrates automatic QoS prediction into the composition process. The goal is to choose a composition solution that satisfies the user request not only for the composition time but also in a relatively long period into the future. Among the commonly used QoS attributes, such as reliability, throughout, response time, availability, and so on, we choose the reliability as the prediction object for three major reasons:

1) Reliability has a close relationship with the hardware/software configuration, network connection, and locations of the service/user, any change of which may lead to a change to the observed reliability value.
2) Once reliability fails to satisfy the basic requirement, the software system will not operate normally, which may lead to immeasurable loss, especially in domains that put high demand on reliability, such as military, aerospace, bank, and stock market.
3) The service composition and reliability prediction model will serve as a proof of concept, which can be extended to other QoS attributes in a straightforward fashion.

The rest of this paper is organized as follows: section II further motivates the proposed composition and QoS prediction model. Section III gives an overview of some related work. Section IV details the proposed service composition model. Section V presents the experimental results, which justify the feasibility and effectiveness of the proposed approach. Section VI concludes the paper.

II. MOTIVATION

The area by composing services together to deliver integrated business applications for users has been widely researched in recent years. Apart from functional requirements, Quality of Service (QoS) properties such as reliability and response time play an increasingly important role in users’ requirement. To achieve the QoS requirements, QoS-aware service composition has emerged with a focus on achieving an optimal composition solution that best suits users’ QoS requirement. Meanwhile, since QoS properties usually change in real-time, QoS prediction has also become a popular research field in service computing. It’s no doubt that the change of atomic services QoS will lead to the QoS change of composite service, which may make the composition solution no longer optimal or even unqualified.

Prediction techniques in traditional software systems are mostly conducted during the architectural design phase to reduce the time and money loss after deployment. A service-oriented system has its unique features because component services are selected and bounded in runtime to realize functionality of the system. Anticipating QoS properties of dynamically composed services is challenging as it is typically not feasible to create and test each possible composition solution that a customer might request, which incur high cost in time and money [7].

To illustrate the problem that may arise in a complex service composition scenario and the significance of solving it, let’s consider a simple example. Assume a user wants to access a storage Web service (e.g., Amazon Simple Storage Service) shown in Figure II and compose it with other components to construct a larger software system. The time that the user needs to transmit the data to the service can be influenced by three major factors: the processing speed of user terminal, network transmission speed, and the processing speed of server. Suppose the user expects the overall response time to be less than 2 seconds. One optional strategy that can be used has terminal response time, network transmission time, and server processing time as 0.1s, 0.2s, and 0.3s, respectively. A strategy as such clearly meets the user requirement. However, if the communication link that connects to the service is unreliable and may break down for some time. This may cause a big delay to response time, resulting in a long waiting time, say 10s. In this case, the strategy no longer works and user has to select again. A more robust solution is to predict the future response time of three parts as that the resultant composition can continue to serve the user not only during the composition time but for a relatively long time into the future.

Existing QoS-aware service composition techniques typ-

![Figure 2. Example of user accessing a storage Web service](image-url)
ically use one-time measurements of QoS attributes for service selection and assume that the measurements are the same for different users and remain static during the whole lifecycle of the composed service. This assumption apparently does not hold for a dynamic service environment, within which QoS properties may change dynamically all the time. The worst case is that QoS values of one or more chosen services become unsatisfactory right after the service composition is completed. In this case, the composition process should be re-executed, which will incur a high overhead. Existing QoS prediction techniques, on the other hand, primarily focus on individual service selection and have not been applied in the context of service composition. The proposed approach aims to seamlessly integrate QoS prediction into the service composition process to deliver a composed service that meets the user’s QoS requirement not only at the composition time but also for a relatively long period of time. Besides reducing the overhead due to recomposition, the proposed composition model will also provide better user experience as users can continuously use the same composition for a long time without major abrupt interruptions.

III. RELATED WORK

In this section we briefly go over the state-of-the-art in the areas of QoS-aware service composition and service reliability prediction.

A. QoS-aware service composition

The service-oriented computing paradigm and its realization through standardized Web service technologies provide a promising solution for the seamless integration of business applications to create new value-added services [5]. In service-oriented architecture, web service composition, by integrating existing services, can fulfill customers’ complicated requirements and implement complex systems. Intuitively, a service composition should not only realize the users’ functional request but also meet their quality constraints, such as price, response time, and so on. Therefore, to distinguish multiple Web services with identical functionality, Quality of Service (QoS) properties are introduced.

QoS is defined as a broad concept that encompasses a number of nonfunctional properties such as price, availability, reliability, and reputation [4]. QoS-aware service composition is to select an optimal set of concrete services to instantiate the abstract workflow with optimal QoS [8]. Currently there are three general solutions to QoS-aware service composition: local selection, global optimization, and the integration of both. Local selection is to select one service from each class of service candidates not relying on the others, which translates the composition into several parallel local selections. The global optimization method focuses on the composite service level and exhausts all composition solutions to find the optimal one. This method can be modeled as a Multi-Choice Multidimensional Knapsack problem, which has exponential time complexity [5], [9]. Some approximation techniques have been developed to tackle the high computational cost. For example, [4] uses linear programming to find the optimal selection of component services. The integration approach has been proposed in [5] recently, which uses mixed integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints and used distributed local selection to find the best Web services that satisfy these local constraints in order to compromise between optimality of composition solution and computational time. Our integration approach is variation of the third one which integrates the QoS prediction into the service composition process.

B. Reliability prediction

Reliability is defined as the probability of the failure-free operation of a software system for a specified period of time in a specified environment [10]. The aim of software reliability prediction is to analyze the system based on system’s characteristics and the historical data, predict the failure probability of the system, and then propose an solution to decrease or eliminate failures. However, the various software frameworks have different emphasis and methods on failure prediction. For example, the component-based system reliability depends both on the reliability of the components and the probabilistic distribution of utilization of the components to provide the service [11].

Considering the special nature of SOC, the reliability prediction of SOC is significantly different from others, which poses some novel challenges. First, the number of Web services keeps increasing in a fast pace. Predicting the reliability of all available candidate services will be very time consuming. Secondly, the reliability of a service oriented system depends on the autonomous third-party services that are outside the control of the system owner and (possibly unreliable) communication links that connect to these services. As one of the first works on reliability prediction for service oriented software systems, Zheng etc. in [12] propose a collaborative reliability prediction approach, which exploits the past failure data of other similar users to predict the Web service reliability for an active user.

Given the above overview of related work on the two research areas, [7] points out that the missing integration of predictive capabilities in the composition frameworks is a major obstacle to the use of quality prediction approaches and makes the QoS properties of dynamically composed services challenging. Therefore, they present a novel service composition process that includes QoS prediction for composed services as an integral part and describe how composition frameworks can be extended to support this process. In this paper, we integrate two concrete methods together to concretize the frameworks, namely the hybrid
of global optimization and local selection, and collaborative filtering reliability prediction.

IV. Model

We assume that the user’s requirements have been already modeled and presented as a set of abstract tasks \( T_{\text{request}} = \{T_1, T_2, \ldots, T_n\} \) as shown in Figure 1. Every abstract task has a set of candidate services \( S_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\} \) which achieve the same functionality but behave differently in QoS attributes. Currently there are many existing solutions to select an optimal composition result among the candidates to meet the user’s QoS requirements and the composition result is represented as \( CS_{\text{result}} = \{s_1, s_2, \ldots, s_n\} \).

To tackle the dynamical changes of QoS properties, we combine the QoS prediction into the service composition process, leading to an integrated service composition model. Based on three reasons given in Section Introduction, we choose reliability as the example QoS attribute to be predicted. In particular, we divide the composition process into four phases, including global QoS decomposition, services selection, reliability prediction, and final selection, which are shown in Figure 3. Table I illustrates the symbols that are used throughout the paper along with their corresponding definitions.

A. Phase 1: Global QoS Decomposition

The first step is to decompose global QoS constraints into local ones, which in turn become the constraints of service selection. Given the problem description, we want to decompose the global QoS attributes into the \( n \) abstract tasks, known as local constraints in the following, which in turn must meet the global ones when being aggregated together. In addition, we want the obtained local constraints to qualify services as more as possible which can avoid eliminating services with high reliability. Hence, we define the optimal decomposition as follows.

DEFINITION 1: (Optimal decomposition) The optimal decomposition of a given vector of global QoS constraints is a set of local constraints which not only satisfy the global constraints when being aggregated together but also maximize the number of qualified services.

For simplicity, we first divide every QoS attribute value in every service set into \( d \) equal divisions:

\[
q_{\min}^{ik} \leq q^1_{ik} \leq \cdots \leq q^z_{ik} \leq \cdots \leq q^{\max}_{ik}, \tag{1}
\]

where \( q^z_{ik} = q^{\min}_{ik} + \frac{z}{d}(q^{\max}_{ik} - q^{\min}_{ik}) \)

where \( q^z_{ik} \) is the \( z \)-th division point about the \( k \)-th QoS attribute in the \( i \)-th service set, among which \( q^{\min}_{ik} \) and \( q^{\max}_{ik} \) correspond to the minimum and maximal value of the \( k \)-th QoS attribute in the \( i \)-th service set.

Then we assign each \( q^z_{ik} \) a \( p^{z}_{ik} \in [0,1] \), which is the probability to be chosen as the local constraint:

\[
p^{z}_{ik} = \frac{c(q^z_{ik})}{m} \tag{2}
\]

where \( c(q^z_{ik}) \) represents the number of qualified services when \( q^z_{ik} \) is chosen as the local constraint and \( m \) is the number of candidate services.

As we expect to maximize the number of qualified services, the selection of appropriate local levels can be modeled as an Linear Programming (LP) problem, where both the objective function and the constraint are linear functions. There are three elements in a LP, decision variable, objective function and constraint. To construct the objective function, we firstly introduce an indicator variable \( x^r_{ik} \in \{0, 1\} \) for each \( q^r_{ik} \). Specifically, \( x^r_{ik} = 1 \) means that \( q^r_{ik} \) is chosen and 0 otherwise. Then, our objective function based on equations (1) and (2) can be defined as

\[
\max \left( \sum_{i=1}^{n} \sum_{k=1}^{r} \sum_{z=1}^{d} p^{z}_{ik} x^z_{ik} \right) \tag{3}
\]

where \( n \) is the number of service sets, and every set has \( r \) services whose QoS value is divided into \( d \) divisions.

Since the integrated local constraints should meet the global ones, the constraint of the above objective function is expressed as

\[
\sum_{i=1}^{n} \sum_{k=1}^{r} d^{z-1}_{ik} x^{z}_{ik} \leq q_k, \tag{4}
\]

\[
\sum_{z=1}^{d} x^{z}_{ik} = 1, 1 \leq k \leq r \tag{5}
\]

Through the above approach, we obtain a list of QoS \( \{q_1, \ldots, q_m\} \) for each service set, which can be used as local constraints for selecting the qualified services in each service candidate set.

B. Phase 2: Service Selection

Existing service selection approaches aim to find an optimal composition by combining the best service in each set. Our selection method is different from existing ones. Since the decision result we finally make is not only based on the QoS constraints but also on the prediction result, our aim is not to select a single service for every task, but an acceptable set of qualified services, whose number is controlled in a user acceptable range. In this phase, we take the user’s preference on the QoS attributes into account. For example, suppose we only consider throughput, response time and price, and among the three QoS, the user concerns response time the most, throughput the second, and price the least. Then we can assign a weight vector \( \{w_1, w_2, w_3\} \) to reflect the degree of preference as \( \{0.6, 0.3, 0.1\} \).

Since we have distributed a global constraint into the \( n \) service sets, service selection will be conducted in the
After the decomposition and local selection, $n$ sets of qualified local services have been identified. Furthermore, all possible compositions among these selected services should meet the global QoS constraints. Conventional software engineering literature suggests that assessing software quality at system implementation time is often times too late [13]. If significant problems are identified when implementation has already been conducted or the system is already in operation, it may cause the re-engineering of the entire system, which will incur high cost. Therefore, quality attributes must be “built into” the software system throughout design and development, and particularly during architectural design phase. Consequently, in our work we integrate the prediction into the composition process before the composed service is in operation. Among all the QoS attributes, we choose reliability as an example to illustrate the key idea of QoS prediction. The approach can be extended to other QoS attributes in a straightforward way.

As the number of qualified services is much smaller due to the selection phase, all composition solutions are considered for reliability prediction. Since the composition has not been formed, each possible composition solution does not have the historical QoS records. Hence, the overall reliability prediction can’t be based on the previous QoS experience of the composition solution under prediction. Nonetheless, the services selected in the second phase have been invoked by other users in the past. Their reliability in a future time can be predicted based on their past performance. Therefore, we employ a two-step process to achieve reliability prediction of a service composition, which consists of atomic service reliability prediction and prediction aggregation.

**Step 1: Atomic service reliability prediction:** There have been plenty of approaches developed to predict the software component reliability. However, if the service is invoked by the current user for the first time and its reliability is known only for the current, reliability prediction...
for the future will become difficult due to the insufficient past failure data. The idea of collaborative filtering, which uses the past failure data of other similar users to predict the Web service QoS for a given active user in a future time, has been adopted in some recent works [12], [14]. The underlying hypothesis of collaborative filtering is that if an active user perceived similar reliability on a set of services with another user, it is expected that the reliability which he/she will perceive from a previously unknown service is, is similar to that user has perceived from the service. Hence, two users are regarded as similar if they perceive similar reliability from a common set of services. Along the same lines, two services are regarded as similar if they deliver similar reliability to a common set of users.

The commonly used similarity measures include Pearson Correlation Coefficient (PCC), Euclidean distance, and Cosine Similarity. Cosine Similarity is mostly used to calculate the similarity between service and the active user perceived similar reliability on a set of services. Along the same lines, two services are regarded as similar if they deliver similar reliability to a common set of users.

Similarly, we calculate the similarity between service and the active user perceived similar reliability on a set of services. Along the same lines, two services are regarded as similar if they deliver similar reliability to a common set of users.

The following equation calculates the similarity of user and the active user using Euclidean distance,

\[
\text{Sim}(a, u) = \sum_{i \in S_c \cap S_a} \sqrt{(p_{a,i} - p_{u,i})^2 / s_{num}}
\]  

where \(s_{num}\) is the number of services invoked by both users. Similarly, we calculate the similarity between service and service as

\[
\text{Sim}(i, j) = \sum_{u \in U_c \cap U_j} \sqrt{(p_{a,i} - p_{u,j})^2 / u_{num}}
\]

where \(u_{num}\) is the number of users that invoked both services.

After the similarity calculation, we set a threshold \(k\) to select top-\(k\) users as similar users of an active user. The set of services that are similar to service can be selected by using a similar approach. Next we use the similar users set and similar service set to predict the missing value \(p_{u,i}\), which denotes the future reliability of service we predict when invoked by user. The value derived through the similar user set is \(p^1_{u,i}\), and whereas the value derived from the similar service set is \(p^2_{u,i}\):

\[
p^1_{u,i} = \bar{p}_u + \sum_{a \in \text{sim}(u)} (p_{a,i} - \bar{p}_a)
\]

\[
p^2_{u,i} = \bar{p}_i + \sum_{j \in \text{sim}(i)} (p_{u,j} - \bar{p}_j)
\]

where \(\text{sim}(u)\) and \(\text{sim}(i)\) are the selected similar user set and service set, respectively; \(\bar{p}_u\) and \(\bar{p}_i\) are the average failure probabilities of different Web services observed by users \(u\) and \(i\); \(\bar{p}_a\) and \(\bar{p}_j\) are the average failure probabilities of services \(i\) and \(j\) by all users. The final prediction is calculated as

\[
p_{u,i} = \lambda \cdot p^1_{u,i} + (1 - \lambda) \cdot p^2_{u,i}
\]  

where \(\lambda\) is a user specified weight to balance between \(p^1_{u,i}\) and \(p^2_{u,i}\).

Step 2: Prediction aggregation: Upon the first step, the missing reliability values of services in the future are predicted for the current user. The next step is to predict the reliability of the composition solutions by integrating the predicted reliability of atomic services based on the service composition workflow. We consider four types of workflow structures: sequence, branch, loop, and parallel [16]. Given the atomic service reliability, the composite reliability can be calculated based on the specific structure features.

- **Sequence or Parallel**: Sequence is the basic structure to compose other structures, in which user executes tasks one after another, whereas all tasks in the Parallel structure are executed at the same time. But the common point of two is that once any task in Sequence or any branch in Parallel fails, the entire composition system will fail. Suppose there are \(n\) tasks in the Sequence (or branches in the Parallel) and the failure probability of each task (or branch) is represented as \(p_i\), the entire composite failure probability will be

\[
P_{\text{fail}} = 1 - \prod_{i=1}^{n} (1 - p_i)
\]

- **Branch**: Different from the Parallel structure, only one branch in the Branch will be executed for each execution. Thus the failure probability can be described as

\[
P_{\text{fail}} = 1 - \sum_{i=0}^{n} b_i (1 - p_i)
\]

where \(b_i\) is the probability that the \(i\)-th branch will be executed and \(p_i\) is as same as above to indicate the probability that the \(i\)-th branch will fail.

- **Loop**: The Loop structure can be understood as a Branch structure is executed for multiple times. Its failure probability is calculated by

\[
P_{\text{fail}} = 1 - \sum_{i=0}^{n} l_i (1 - p_i)^i
\]

where \(p_i\) is the failure probability of the branch and \(l_i\) presents the probability of the branch running for \(i\) times, which satisfies that \(\sum_{i=0}^{n} l_i = 1\).

Based on the workflow structure of a given composition solution, we use the above equations or the combination of them to calculate the failure probability of each composition.
result represented as $CS_{fail} \in [0, 1]$ and the smaller its value is, the more reliable the composition result is.

D. Final Selection

Given the reliability prediction result, we can combine it with QoS attributes satisfaction to achieve the final selection decision. A simple way to aggregate the prediction with QoS attributes is to use a weighting mechanism, which assigns $w_{rel} \in [0, 1]$ to weight the importance of user putting on the future reliability and $(1 - w_{rel})$ to the current QoS attributes satisfaction which is defined as $f(s_{ij})$ before in Equation(5). Hence, user preference on a final composition solution is given by

$$W_{result} = f(s_{ij}) \times (1 - w_{rel}) - CS_{fail} \times w_{rel} \quad (15)$$

According to the definition of $CS_{fail}$ and $f(s_{ij})$, the composition result is more qualified when with bigger $W_{result}$.

V. EXPERIMENTS

In this section, we present an experimental evaluation for the integrated service composition model. The aim of this evaluation is to validate that our approach achieves more reliable composition solution that is expected to consistently serve for the user for a relative long time.

Therefore, we choose the reliable degree of the selected composition strategy to evaluate the effectiveness of the proposed approach. More specifically, reliable degree of a composition solution is defined as

$$Rel_{degree} = \frac{\min(CS_{fail}^z)}{CS_{fail}^z} \quad (16)$$

where $CS_{fail}^z$ denotes the reliability of $z$-th composition solution and $\min(CS_{fail}^z)$ denotes the minimum failure probability among all possible composition solutions, namely the most reliable composition solution. A bigger $CS_{fail}$ implies higher failure probability and as a result the $Rel_{degree}$ has a smaller value.

A. Experimental Dataset

The experiments are performed on two QoS datasets in our evaluation. For real dataset, the publicly available dataset QWS [5] is the classical dataset for service composition with measurements of nine QoS attributes for 2,507 real-world web services. Real-world Web service performance dataset (published at www.wsdream.net) [12] is comprised of failure probabilities of 100 Web services observed by all the 150 service users. This data is presented as a 150 × 100 failure probability matrix, which we refer to as RP (i.e., Reliability Performance). We combine the above two datasets to serve the purpose of our experiments by assigning the values of service performance in RP to the services in the QWS dataset randomly, which will be used as the reliability prediction basis. The second dataset is totally synthetic, which assigns arbitrary QoS values and the corresponding reliability performance to generates 10,000 services, which also makes it a much larger dataset. We consider three QoS properties in both datasets: reliability, price and response time and vary the number of service classes and candidates respectively to validate our integration approach achieves more reliable solution. In addition, each set of experiment was performed repeatedly for 20 times, and the average performance was calculated.

B. Experiments on Real Dataset

Since there are 2,507 services in QWS, first, we fix the service classes as 20 and vary the number of service candidates in each class from 50 to 90 to perform the experiment. In the second experiment, we validate the effect of the size of service classes with an assumption that there are 80 service candidates in each class. Suppose users’ preference on three QoS is $\{0.2, 0.45, 0.35\}$, the experimental result is shown in Figure 4. The red line shows the reliable degree of our integration approach and on the contrary, blue line indicates the traditional hybrid of global optimization and local selection proposed in [5]. Through the experiment data, we prove that prediction based composition model clearly outperforms the traditional composition approach and consistently reports higher reliability degrees.

C. Experiments on Synthetic Dataset

Similar to the above dataset, the experiments over the synthetic dataset include two parts. One part is that the number of service classes is fixed as 30 and the number of candidate services for each class is changed from 100 to 300 with a step of 50 to grow. The second in turn is to fix the candidate size as 200 and the size of classes varies from 25 to 45, as is presented in Figure 5, in which the reliable degree of our approach is higher than or at least the same as the traditional.

VI. CONCLUSION

In this paper, we present a new service composition model that considers the dynamic changes of the QoS attributes during the service selection of a composition process. Reliability is chosen as a proof of concept among other dynamic QoS attributes. By integrating the reliability prediction
mechanism into the composition process, it aims to deliver more reliable composition solutions that will not only serve the user for the composition time but also continue to deliver reliable performance into the future. Experimental results on both real and synthetic datasets justify the effectiveness of the proposed composition model.

This work is partially supported by JSNSF of China (No.BK2010417).

REFERENCES


