Research on Cost-Driven Services Composition in an Uncertain Environment

Honghao Gao[1,2], Wanqiu Huang[2], Yucong Duan[3], Xiaoxian Yang[4], Qiming Zou[1]

[1] Computing Center, Shanghai University, China
[2] School of Computer Engineering and Science, Shanghai University, China
[3] College of Information Science and Technology, Hainan University, China
[4] School of Computer and Information Engineering, Shanghai Polytechnic University, China
gaohonghao@shu.edu.cn, vicky_2017@shu.edu.cn, duanyucong@hotmail.com, xxyang@sspu.edu.cn, kim@shu.edu.cn

Abstract

In recent years, increasing numbers of researchers have concentrated on service workflow to support cross-domain software development. However, the uncertain characteristics of the Internet impose high risks on service workflow reliability. The risk of failure caused by unavailable services may increase costs when using service workflow-based applications. Thus, it is necessary to consider the non-functional factors, such as service cost and reliability. In this paper, we propose a cost-driven services composition approach for enterprise workflows that employs formal verification to recommend appropriate services for abstract workflows. The services composition is measured quantitatively to ensure that the configuration to service the workflow solution has the best performance, high reliability and low cost. First, this solution introduces a service search approach based on an inverted index, and the service recommendation method is based on an improved Pearson formula. Next, the solution returns a minimum set of candidate services for constructing a workflow instance. Second, the service and workflow models are defined to formalize the behaviour of service composition; this is considered to be a verification model. Third, transformation rules are provided to change BPEL4WS into a verification model, and PCTL (Probabilistic Computation Tree Logic) formulae are used to specify the reliability and cost-related properties. The quantitative verification method checks each possible plan for service composition using probabilistic model checking. Finally, the results of a series of experiments show that our approach is effective in generating an optimal service workflow.

Keywords: Service workflow, Probabilistic model checking, Uncertain environment, Formal verifications, Service search and recommendation

1 Introduction

Service-Oriented Computing (SOC), which has been widely adopted in modern industry and academia, provides the ability to develop an information system quickly using Web services to integrate distributed applications [1-2]. However, a single service has difficulty in satisfying the complex requirements of business logic; multiple services are required instead, which calls for service composition. As one of a number of possible technologies, service workflows provide a flexible way to address this problem. Each activity of a business process in an abstract workflow can be mapped to a service to provide the value-added functions. The diversity and complexity of service composition to abstract workflow involves service search, service selection, and services verification [3-4].

However, the open, dynamic and ever-changing features of the Internet will inevitably cause workflow failures. Traditional workflow approaches are static and certain. They will break down because of the risk of failure, which can be randomly encountered in this uncertain environment [5]. If an enterprise workflow has no error-response mechanisms that configure workflows dynamically with new services, the service workflow will eventually cause serious performance issues.

Workflow's service selection aims to choose services and compose them into a new composite service to support abstract business processes. However, most existing studies [6-10] do not consider risk-aware workflow service composition. Instead, these techniques require services composition only to be consistent with the functional behaviour claims. In a practical execution environment, service failure risks include the possibility of service disruptions. Thus, it is possible that even a new recommended plan of services composition will again encounter failures. The new recommended plan has several limitations and may

*Corresponding Author: Xiaoxian Yang; E-mail: xxyang@sspu.edu.cn
DOI: 10.3966/160792642019052003009

cause inconvenience to users. To this point, service workflows implemented by different service composition plans pose different risks because their component services have different reliabilities. Therefore, the performance of a service workflow is dependent not only on service function but also correlated with service quality. To minimize the risks of the new service configuration also failing, we need to select the optimal service workflow with the lowest cost and the highest reliability. Consequently, the balance between cost and reliability should be considered, and this consideration may have a significant impact on the efficiency of the service workflow.

In this paper, we propose a cost-driven service composition approach that consists of two stages: a service search stage and a quantitative verification stage. The former returns possible plans for service composition, while the latter checks each plan to evaluate the performance of such a service workflow. First, the cost/reliability-oriented selection problem is discussed using an example of a device purchase auditing workflow. This example primarily demonstrates that the risks of service composition affect workflows in uncertain environments. Second, the traditional service search, which uses functional matching, is highly inefficient when the scale of services becomes large. Therefore, the inverted index search-based service selection and the functional consistency-oriented service search method are proposed. Considering the search efficiency, the improved Pearson formula is used to recommend related services. Third, probabilistic model checking is employed to verify whether each service configuration plan satisfies both the functional requirements and the non-functional objectives. Formal models of service workflow with cost and reliability specification are defined, and transformation rules are proposed to transform business processes into a formal model. Finally, quantitative verification results are used to guarantee that the service workflow is trustworthy.

As a core issue, we transform the problem of service composition into the process of applying a probabilistic model check. Each plan is checked quantitatively. Through this check, the verification results with cost and reliability help us to select better service matches.

The rest of this paper is organized as follows. Section 2 reveals our motivation through a workflow scenario. Section 3 shows the candidate service set generation process. Section 4 introduces the formal models and verification processes for checking service processes. Section 5 discusses the experimental analysis. Section 6 reviews related work, and Section 7 presents conclusions and provides future research directions.

2 Motivation Scenario

Figure 1 describes the business processes involved in a device purchase auditing workflow: this workflow requires a login, then performs a search activity, an accessories activity, a valuableDevice activity, an auditing activity, and a print activity. In this abstract process, each activity is mapped to a corresponding service that implements the required business logic. According to combinatorial theory, 2304 (4*4**4*3*3 *4) possible plans could be derived from Figure 1. Thus, the problem of service composition is changed into a service selection process that involves selecting one suitable solution among the possible service configurations. Each service has a different value of non-functional features that will impact the service workflow performance. Consequently, different solutions display different results. For example, <Login2, Search3, (Accessories3, valuableDevice3), Audit1, Print4> would behave differently from <login3, search4, (Accessories1, valuableDevice2), Audit2, Print1>.

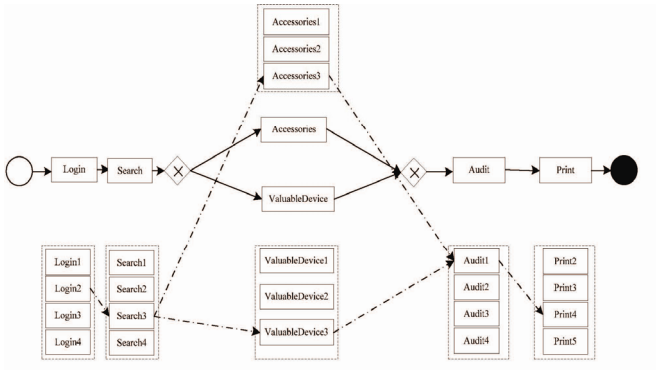


Figure 1. Example of device purchase auditing workflow

The details of all the available services, including cost and reliability, are presented in Table 1, in which column R shows the service reliability as a probability, and column C shows the cost of service execution. Suppose that an employee wants to apply to purchase devices via the business process depicted in Figure 1. To provide the best service, the workflow should provide an optimal performance. Specifically, the service workflow selects one service from {Login1, Login2, Login3, Login4} to fulfil the Login task, one service from {Search1, Search2, Search3, Search4} to fulfil the Search task, one service from {Accessories1, Accessories2, Accessories3} to fulfil the Accessories task, one service from {valuableDevice1, valuable Device2, valuableDevice3, valuableDevice4} to fulfil the valuableDevice task, one service from {Audit1, Audit2, Audit3, Audit4} to fulfil the Audit task, and one service from { Print1, Print2, Print3, Print4} to fulfil the Print task.

Table 1. Information details of all available services

Service	R	C	Service	R	C
Login1	0.91	10	Print4	0.56	9
Login2	0.9	20	Accessories1	0.99	42
Login3	0.94	15	Accessories2	0.91	33
Login4	0.78	14	Accessories3	0.83	35
Search1	0.93	11	ValuableDevice1	0.56	13
Search2	0.89	23	ValuableDevice2	0.92	32
Search3	0.8	30	ValuableDevice3	0.87	45
Search4	0.91	21	Audit1	0.91	45
Print1	0.45	30	Audit2	0.93	24
Print2	0.93	20	Audit3	0.9	33
Print3	0.42	32	Audit4	0.85	52

Suppose the cost of Login2 is 20, and the cost of Search3 is 30. If these tasks are invoked in sequence, the total cost will be 50. Further, assume that the reliability of Login2 is 0.9, and the reliability of Search3 is 0.8; the total reliability is 0.72. However, if the reliability of non-functional objectives does not satisfy the users' expectations, the current service plan will be unacceptable. Thus, it is reasonable to consider both functional behaviours and non-functional features.

Users may select component services with high reliability or low cost, because not all of the services in Table 1 have both low cost and high reliability. If only the reliability of each feasible solution is considered, the service composition result is {Login3, Search1, Print2, Accessories1, ValuableDevice2, Audit2}, while if only the cost of each feasible solution is considered, the service composition result is {Login1, Search1, Print4, Accessories2, ValuableDevice1, Audit2}. These two cases imply that attending to only one factor of non-functional requirements does not provide the optimal solution.

In contrast, when both cost and reliability features are considered during service selection, the ultimate result will be significantly improved and will bring more benefits to service workflow. However, it is not easy to solve the problem of cost-driven services composition in an uncertain environment.

The traditional service selection methods focus on maximizing one-dimensional benefits. Figure 2 shows four possible service composition plans for the workflow in Figure 1. However, we cannot judge which of those is the best service configuration for the current workflow. Based on the cost and reliability values of the available services in Table 1, Figure 2(c) shows the maximum reliability version in the form of a probability, Figure 2(d) shows the maximum cost version, and Figure 2(a) and Figure 2(b) show the general composition plans. Unfortunately, it is difficult to handle two- or more-dimensional evaluations, such as considering both cost and reliability. A number of alternative solutions used in service workflow may provide higher reliability, but cost more money than less reliable solutions. For example, although the cost of the workflow in Figure 2(d) is greater than that shown in Figure 2(c), the reliability of the workflow in Figure 2(c) is lower than that shown in Figure 2(d).

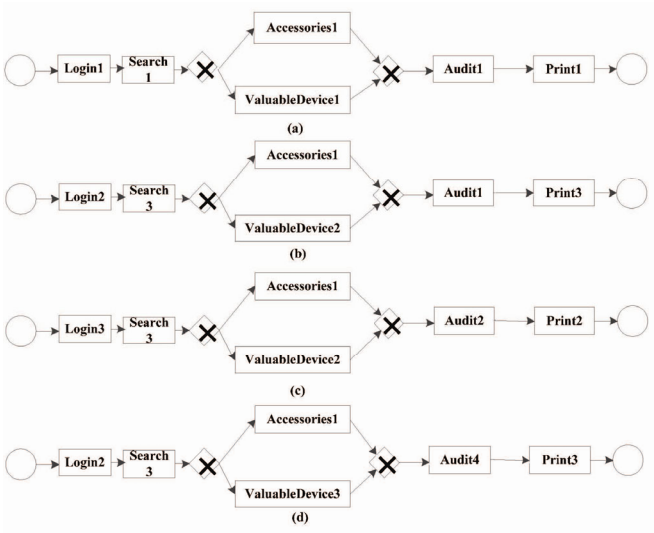


Figure 2. Service instances for the workflow in Figure 1

Thus, if there are service composition plans that meet threshold values for both cost and reliability, we would consider those to be better plans. However, solving this problem requires a method for quantitatively verifying a service workflow to consider each service composition plan in order to distinguish which single solution is best and return one that satisfies both the functional and non-functional requirements.

3 Service Selection for Composition

In this section, formal models are proposed for Web service and abstract workflow. The problems involved in service search and service recommendation to generate a service candidate set are discussed.

3.1 Formal Models and Problem Definitions

Definition 1 (Web Service). A Web service is defined as a tuple $ws::=(id,F,I,O)$, where

(1) id is the identifier for each Web service;

(2) F is the functional description for each Web service;

(3) $I=\{i_1,i_2,...,i_n\}$ is a set of inputs received from the invoker;

(4) $O=\{o_1,o_2,...,o_m\}$ is a set of outputs returned to the invoker.

A Web service receives the inputs I from an invoker and returns the outputs O to the invoker via interface operations. In general, the inputs and outputs are basic value or object types described in WSDL[1] files. The functional description F represents the service functionality, which can be considered to be the source used for function matching.

[1] WSDL. http://www.w3.org/TR/wsdl.

Definition 2 (Web Service Performance). To formalize non-functional behaviour, the cost and reliability of a Web service *ws* are defined as follows.

(1) The reliability of a Web service *ws* is denoted as $R(ws)\in[0,1]$, which represents the probability of reliable execution when a request is sent to invoke the service.

(2) The cost of a Web service *ws* is denoted as $C(ws)\in\mathbb{R}$, mapped to a real number, which shows how much the user should pay when agreeing to use service *ws*.

In an uncertain environment, a Web service may be unavailable, which means that software using it will encounter unpredictable risks. Definition 2 indicates that each service displays both functional and non-functional behaviours. During a service invocation, the user should be prepared for service failures. Thus, a service level agreement (SLA [2]) is often used to predefine different service costs under different reliability conditions.

Definition 3 (Business Workflow Model). A business workflow describes business logic in an abstract manner, and is defined as the tuple $BWM::=(N,C,T,s,e)$, where

(1) N is a set of logic tasks that can be mapped to different services;

(2) C is a set of control conditions for a workflow, e.g., the execution probabilities of branch and loop structures;

(3) $T \subseteq C\times N \cup N\times N \cup N\times C$ is a set of transitions, which are divided into three types: $C\times N$, $N\times N$, and $N\times C$;

(4) $s\in N$ is the starting node of the logic task, and $e\in N$ is the ending node of the logic task.

The enterprise information integration task involves mapping corresponding services to each logic task node to transform abstract business logic into an actual application. Thus, a workflow is considered to be a remarkable and promising solution to agile software engineering. However, how to conduct services composition based on abstract business processes is a key problem, because selecting different component services will cause the workflow to display different performances.

Definition 4 (Workflow Requirements). A workflow includes two types of requirements. One type is functional requirements, which are used to perform service matching. The other type is non-functional requirements, which are used to evaluate the service performance.

(1) The functional requirement $\alpha(n)=I_w$ denotes the inputs for each logic task $n\in N$. The functional requirement $\beta(n)=O_w$ denotes the outputs for each logic task $n\in N$.

During service comparison, a candidate service should have functions that match the given inputs and outputs. If Web service *ws* satisfies the functional requirement of logic node *n* of a workflow, then it should satisfy $ws.I\supseteq\alpha(n) \wedge ws.O\supseteq\beta(n)$.

(2) The non-functional requirements consist of local and global requirements. The local requirement focuses on the logic task, where $\delta^R(n)$ and $\delta^C(n)$. The global requirement focuses on the workflow, where χ^R and χ^C.

According to non-functional requirements, there are two major calculations. (1) For service instance, if Web service *ws* satisfies the non-functional requirement of logic node *n* in the workflow, then it should satisfy $R(ws)\geq\delta^R(n) \wedge C(ws)\leq\delta^C(n)$. (2) For workflow instance, to compute the global values in order to compare with χ^R and χ^C, it should pay attention to the flow structure of Definition 3. The general Cost/Reliability computing formulae [11-13] for service workflow are introduced in Table 2, where *WS* is a set of services that will be mapped to the abstract workflow. Note that p_{ws} is a specified probability for the selected branch *ws* in choice structure. The Cost/Reliability computing formulae include *Sequential* structure, *Parallel* structure and *Choice* structure. However, when the workflow contains loops, the formula above has difficulty in handling these conditions. Note that the key issue is to compute steady state probability when there has a loop-carried branch in service workflow. Therefore, we are motivated to compute these values using formal method.

Table 2. Cost/reliability computing for service workflow

	Cost	***Reliability***
Sequential	$\sum_{ws\in WS} C(ws)$	$\prod_{ws\in WS} R(ws)$
Parallel	$\sum_{ws\in WS} C(ws)$	$1-\prod_{ws\in WS}(1-R(ws))$
Choice	$\sum_{ws\in WS} p_{ws}\times C(ws)$	$1-\prod_{ws\in WS}(1-p_{ws}\times R(ws))$

3.2 Candidate Service Selection

Given a set of interface operations with inputs and outputs, a service search involves comparing each service from a service repository to find matches. The complexity of service planning is $O(M^N)$ in the worst case, where *N* is the number of logic task nodes of workflow, and *M* is the service number in the service repository. Therefore, finding matches becomes a large task as the number of workflow nodes and service numbers increase. Considering that semantic searching may return services with unmatched interface operations, the keyword search can be improved by using the inverted index method to build a service index to enhance the searching efficiency.

Definition 5 (Service Index). The inverted index for service is defined as tuple $SI::=(k,S,f)$, where

[2] SLA.https://en.wikipedia.org/wiki/Service-level_agreement

(1) k is the set of keywords against which inputs and outputs of interface operations are compared to identify matching services.

(2) S is a set of services related to the *id* of the Web service.

(3) $f(k)\rightarrow 2^S$ defines that services with same keyword, k, can be grouped into a set.

Because the service index is a one-to-many pattern, each index is an identification that will be mapped with different services, showing that they can support at least one interface. From an organizational perspective, the service index includes a set of interfaces grouped by inputs or outputs.

A service search aims to find a set of candidate services that match the target interface operations. This service index method improves on traditional service search methods because it returns target services more quickly. For example, there are six indexes {I1, I2, I3, O1, O2, O3} in Table 3. A search for services that contain Output O2 returns {S1, S2, S3, S5, S7} as candidate services without having to compare other services.

Table 3. Example of a Service Index

k	S				
I1	S1	S3	S4	S6	
I2	S3	S4	S7	S10	S11
I3	S1	S5			
O1	S1	S4	S7	S1	
O2	S1	S2	S3	S5	S7
O3	S3	S4	S5	S11	

Definition 6 (Functional Consistency). Let $CS=\{cs_1, cs_2, \ldots, cs_n\}$ be a candidate service set, and let $cs_i \in CS$ be mapped to $n_j \in N$. Functional consistency requires that

$$\forall n \in N \bullet (\forall i \in \alpha(n), \exists cs \in CS \bullet \exists i' \in cs.I \wedge i=i') \wedge (\forall o \in \beta(n), \exists cs \in CS \bullet \exists o' \in cs.O \wedge o=o') \quad (1)$$

Note that $\alpha(n)$ and $\beta(n)$ are functional requirements which can be referenced to Definition 4. Functional consistency requires that each input or output of the task node of workflow be represented by a corresponding service in the candidate service set. However, there will be numerous candidate services; thus, the current set must be refined to return the most suitable services.

Definition 7 (Candidate Service Set Refinement). Suppose the target interface set is $\{op_1, op_2, \ldots, op_n\}$. The candidate service set refinement process is divided into two steps.

(1) The intersection function is used to build an initial set.

$$CS \leftarrow \bigcap_{i=1}^{i=n} f(op_i) \quad (2)$$

This function removes redundant services using the intersection function. For example, let {I1, I2, O3} be target interfaces. From Table 2, there are three sets that f(I1)={(I1,S1), (I1,S3), (I1,S4), (I1,S6)}, f(I2)={(I2,S3), (I2,S4), (I2,S7), (I2,S10), (I2,S11)}, and f(O3)= {(O3,S3), (O3,S4), (O3,S5), (O3,S11)}. We can obtain the candidate service set that CS={S3,S4}.

(2) The multi-objective selection function is used to remove candidate services that cannot satisfy the non-functional requirements.

$$CS \leftarrow CS \setminus \{ws \in CS \mid R(ws) < \delta^R(n) \vee C(ws) > \delta^C(n)\} \quad (3)$$

With the help of multi-objective selection function, the candidate service set can be refined by eliminating services that have low reliability or high cost. Thus, it can guarantee the high feasibility of services composition.

Another way to improve the search efficiency is through service recommendation, which is different from general service searching. In certain cases, dynamic services composition should be completed in seconds. Thus, recommending related services [14-15] for a workflow can reduce the service search duration. In the following, we will discuss the improved service recommendation method based on user usage records to select services to support business processes represented by abstract workflows.

Definition 8 (Service Recommendation). Suppose the usage frequency of services in a service repository is recorded and updated in real time. The service recommendation method based on the Pearson formula focuses on these usage records and includes the following steps.

(1) The usage frequency of a service is in the form of a vector. For user us_i, the historical record is $r_{ws1,usi}$, $r_{ws2,usi}$, …, $r_{ws(n-1),usi}$, $r_{wsn,\ usi}$ where $r_{wsj,\ usi}$ shows the usage frequency of service ws_i used by user us_j.

(2) After calculating the historical records of all users, a two-dimensional matrix of users and services is generated as follows:

$$\begin{array}{c} \\ us_1 \\ us_2 \\ \cdots \\ us_{m-1} \\ us_m \end{array} \begin{array}{c} \begin{array}{cccccc} ws_1 & ws_2 & & ws_{n-1} & ws_n \end{array} \\ \begin{Bmatrix} r_{ws1,us1} & r_{ws2,us1} & \cdot & r_{ws_{n-1},us1} & r_{ws_n,us1} \\ r_{1ws,us2} & r_{ws2,us2} & \cdot & r_{ws_{n-1},us2} & r_{ws_n,us2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{ws1,us_{m-1}} & r_{ws2,us_{m-1}} & & r_{ws_{n-1},us_{m-1}} & r_{ws_n,us_{m-1}} \\ r_{ws1,us_m} & r_{ws2,us_m} & & r_{ws_{n-1},us_m} & r_{ws_nus_m} \end{Bmatrix} \end{array}.$$

(3) The user-service matrix is changed into a service-service matrix according to the Pearson formula [16-17]:

$$v_{ws,ws'} = \frac{\sum_{i \in n} (r_{ws,us_i} - \bar{R}_{ws})(r_{ws',us_i} - \bar{R}_{ws'})}{\sqrt{\sum_{i \in n} (r_{ws,us_i} - \bar{R}_{ws})^2 \sum_{i \in n} (r_{ws',us_i} - \bar{R}_{ws'})^2}} \quad (4)$$

where $v_{ws,\ wsi}$ is a correlation value for services ws and

ws', $r_{ws,\,usi}$ is a count value for service ws used by user us_i, and $\overline{R}_{ws}$ is the average count value of user ws. Here, $r_{ws',usi}$ is a count value for the service ws' used by user us_i, and $\overline{R}_{ws'}$ is the average count value of user ws.

However, this approach should also consider interface operations, since any incompatibility will make the service unavailable. Obviously, the correlation value $v_{ws,\,ws}$ is not sufficient to determine whether two users are similar or not, especially if they are incompatible with each other. To this end, the $v_{ws,\,ws'}$ is improved to enhance the accuracy of user similarity based on the interface operation, cost, and reliability, allowing the Pearson formula to be suitable for recommending related services.

$$v'_{ws,ws'} = (\lambda \times \frac{|ws'.I| + |ws'.O|}{\underset{n\in N}{Max}\{|\sum \alpha(n)| + |\sum \beta(n)|\}} + \phi \times \frac{C(ws')}{\underset{n\in N}{Min}\{C^w(n)\}} + \varphi \times \frac{R(ws')}{\underset{n\in N}{Max}\{R^w(n)\}}) \times v_{ws,ws'} \quad (5)$$

In Formula (5), we consider the interface similarity $\frac{|ws'.I| + |ws'.O|}{\underset{n\in N}{Max}\{|\sum \alpha(n)| + |\sum \beta(n)|\}}$, the cost similarity $\frac{C(ws')}{\underset{n\in N}{Min}\{C^w(n)\}}$, and the reliability similarity $\frac{R(ws')}{\underset{n\in N}{Max}\{R^w(n)\}}$. Then, the parameters λ, ϕ and φ are used to control the weight values while computing the correlation value $v_{ws,\,ws'}$, where λ+ϕ +φ=1. If cost is most important, then ϕ can be greater than φ and λ. If reliability is most important, then φ can be greater than ϕ and λ. Finally, if interface is most important, then λ can be greater than ϕ and φ.

(4) Finally, the new correlation matrix is generated as follows:

$$\begin{array}{c} \\ ws_1 \\ ws_2 \\ .... \\ ws_{n-1} \\ ws_n \end{array} \begin{array}{c} \begin{array}{cccccc} ws_1 & ws_2 & & ws_{n-1} & ws_n \end{array} \\ \begin{pmatrix} v_{1,1} & v_{1,2} & . & v_{1,n-1} & v_{1,n} \\ v_{2,1} & v_{2,2} & . & v_{2,n-1} & v_{2,n} \\ . & . & . & . & . \\ v_{n-1,1} & v_{n-1,2} & & v_{n-1,n-1} & v_{n-1,n} \\ v_{n,1} & v_{n,2} & & v_{n,n-1} & v_{n,n} \end{pmatrix} \end{array} .$$

For each service, the correlation value can be ranked from low to high for service recommendation. However, according to the improved Pearson formula, there will still be a huge number of candidate services. The following function, *IPS(ws)*, will output the top-k related services as the recommendation set RS.

$$RS \leftarrow \bigcup_{i=1}^{k} IPS(ws) \setminus \{ws \in CS \mid R(ws) < \delta^R (n) \vee C(ws) > \delta^C(n)\} \quad (6)$$

Thus, Formula (6) is used to eliminate services that cannot satisfy the non-functional requirements, especially those of reliability and cost.

3.3 Candidate Service Set Optimization

Under the worst conditions, candidate services from the recommendation service set will lead to the state space explosion problem. To obtain the minimal set, the purpose of candidate service set optimization is to compare each service with a logic node of the workflow after applying the improved Pearson formula, focusing on the service input and output.

(1) The input number of candidate services is defined as follows:

$$IW(RS) ::= |\sum_{rs\in RS} rs.I| - |\sum_{n_j\in N} \alpha(n_j)| \quad (7)$$

Then, the input target is changed to compute the minimal set *RS'*:

$$\begin{array}{ll} Min & IW(RS') \\ Subject\,to & RS' \in 2^{RS} \wedge IW(RS') > 0 \end{array} . \quad (8)$$

(2) The output number of candidate services is defined as follows:

$$OW(RS) ::= |\sum_{rs\in RS} rs.O| - |\sum_{n_j\in N} \beta(n_j)| \quad (9)$$

.

Then, the output target is changed to compute the minimal set *RS'*:

$$\begin{array}{ll} Min & OW(RS') \\ Subject\,to & RS' \in 2^{RS} \wedge OW(RS') > 0 \end{array} \quad (10)$$

Definition 9 (The Minimal Service Set). The requirement to find the minimal set is a multi-objective optimization process that will return a subset, RS^{min}, in which the minimal set is as small as possible while still supporting the logic nodes. This process is defined as follows:

$$RS^{Min} = \begin{cases} Max & W(RS') = (-IW(RS'), -OW(RS')) \\ S.T & RS' \in 2^{RS} \end{cases} . \quad (11)$$

Considering the inputs and outputs of the interface requirements and RS^{Min}, the particle swarm optimization algorithm is employed to find a global optimal solution during multiple iterations. Thus, the improved particle swarm optimization algorithm (PAMS Algorithm) is designed to find an optimal set when candidate services haves redundant input and output. The algorithm details are as follows

Algorithm. *PSO-like Algorithm for Minimal Services (PAMS)*
Input: *Service SR[] and User Request UR[]*
Output: *The minimal service set* RS^{min}

Function PAMS (RS[], UR[])
Inti the total best $G_{best}=(G_1,G_2,\ldots,G_{M-1},G_M)$
Init totalInput=0, totalOutput=0
For each item in UR
totalInput +=UR[item].input
totalOutput +=UR[item].output
End for
For Each particle in SR.length
Random the Position of particle, such as
$x_i=\{x_{i1},x_{i2},\ldots,x_{i(k-1)},x_{ik}\}, x_{ik}\in\{0,1\}$
Random the Victor of particle
like $v_i=\{v_{i1},v_{i2},\ldots,v_{i(k-1)},v_{ik}\}, v_{ik}\in\{-1,1\}$
Inti the history best $p_i=(p_{i_1},p_{i_2},\ldots,p_{i_{M-1}},p_{i_M})$
End For
Init $T_{max}=(SR.length+UR.length)^2$
While t in T_{max}
Init flag=0, InputCount=0, OutputCount=0
For each particle in SR.length
For each item in x_i
If $x_i[item]=1$ *then*
InputCount +=SR[i].input
OutputCount +=SR[i].output
End if
End for
End For
If inputCount >=totalInput and
OutputCount>=totalOutput then
$F(x_i(t))=\sum_{j=1}^{R}(v_i[j])$
If $F(x_i(t))<p_i(t)$ *then*
$p_i(t)=F(x_i(t))$
End If
If $p_i(t)<G_{best}$ *then*
Update G_{best} *that* $G_{best}\leftarrow p_i(t)$
End If
End if
$c_1(t)=c_{1init}+\cos(t*\pi/T_{max})$
$c_2(t)=c_{2init}-\cos(t*\pi/T_{max})$
$\omega(t)=(\omega_{max}-\omega_{min})/2+\cos(t*\pi/T_{max})$
For each particle in SR.length
$v_i^{(k+1)}=\omega*v_i^k+c_1*(p_i^k-x_i^k)+c_2*(G_{best}-x_i^k)$
$x_i^{(k+1)}=x_i^k+0.5*v_i^{(k+1)}$
End for
End While
RETURN G_{best}
End Function

The algorithm checks the service set step by step according to the particle swarm iteration. The position of each particle is defined as $x_i=\{x_{i1}, x_{i2},\ldots,x_{i(k-1)},x_{ik}\}$, $k\in[1,R]$, $i\in[1,N]$, where R is the number of particles, and $x_{ik}=\{0,1\}$ corresponds to user requirement k mapped to the i^{th} service. The velocity of each particle is defined as $v_i=\{v_{i1}, v_{i2},\ldots,v_{i(k-1)},v_{ik}\}$where $v_{ik}=\{-1,1\}$.

The initial particle positions and velocities are randomly selected form {0,1} before iteration begins. The historical optimal value of each particle is $p_i=\{p_{i1}, p_{i2},\ldots,p_{i(M-1)},p_{iM}\}$, and the global optimal value is $G_{best}=\{G_1, G_2,\ldots,G_{(M-1)},G_M\}$. Then, the updated formula for the place of the particle and velocity is

$$v_i^{(k+1)}=\omega*v_i^k+c_1*(p_i^k-x_i^k)+c_2*(G_{best}-x_i^k) \quad (12)$$

$$x_i^{(k+1)}=x_i^k+0.5*v_i^{(k+1)}. \quad (13)$$

The ω is called an inertial factor and is a convex and concave integrated function:

$$\omega(t)=(\omega_{max}-\omega_{min})/2+\cos(t*\pi/T_{max}), \quad (14)$$

where ω_{max} and ω_{min} represent the maximum and minimum values, respectively, t is the number of the current iteration, and T_{max} is the maximal number of iterations. This formula guarantees that the iteration is convergent.

The learning factor c is defined to control the cognition degree:

$$\begin{cases}c_1(t)=c_{1init}+\cos(t*\pi/T_{max})\\ c_2(t)=c_{2init}-\cos(t*\pi/T_{max})\end{cases}, \quad (15)$$

where c_{1init} and c_{2init} represent the initial value of individual cognition and the initial value of collective cognition, respectively. During the particle iteration process, the criteria for historical optimal value should be satisfy the formulae *IW*(*RS*) and *OW*(*RS*). Then, the fitness function is defined to calculate the minimum number of services required to build the service set.

$$F(x_i(t))=\sum_{j=1}^{R}(v_i[j]). \quad (16)$$

Then, this equation compares $p_i(t)$ with the fitness function. When $F(x_i(t))<p_i(t)$, then the historical optimal value is updated $p_i(t)=F(x_i(t))$. Finally, G_{best} will return the global optimal value to the user.

4 Applying Probabilistic Model Checking to Service Composition

The challenge of service composition for abstract workflow lies in evaluating the solution. To solve this problem, we apply probabilistic model checking to service composition to quantitatively verify each plan of service configurations.

4.1 Verification Process Overview

According to the service plan, candidate services will be mapped to each task to achieve the target service workflow. The probabilistic model checking employed to verify service workflow includes three steps: formal modelling, verification property generation, and verification execution. The verification result contains not only the satisfiability assertion between the model and the property but also quantitative information concerning cost and reliability.

Step 1. The behavioural model is generated in the form of a formal model that considers the temporal relations of service invocation.

Step 2. The workflow requirements are transformed into a verification property in the form of temporal logic formulae.

Step 3. The model checking tool PRISM[3] is used to perform automatic verification after the formal model and verification property are coded into PRISM input language.

Step 4. The quantitative verification results about probability and cost are analysed to confirm the optimal plan to services composition.

4.2 Formal Verification Model

Definition 10 (Formal Model of Service Composition). The verification model is a labelled transition system defined as a tuple $FWS::=(S, s0, \delta, P, C, AP, L)$, where

- S is a finite set of states in which each state corresponds to a Web service;
- $s0$ is an initial state;
- δ (S) $\rightarrow S$ is a finite set of edge relations that represent the service invocation relations;
- $P(\delta) \rightarrow [0,1]$ is a transition probability matrix, where $\sum_{s' \in S} P(s,s')=1$ for all $s \in S$;
- $C(S) \rightarrow \mathbb{R}$ is a cost function that shows the service cost as a real number;
- AP is a finite set of atomic propositions;
- L: $S \rightarrow 2^{AP}$ is a labelling function that assigns a set of atomic propositions to each state $s \in S$.

For state s, $s' \in S$, the transition probability matrix gives the probability $P(s,s')$ of making a transition from s to s' in one discrete step. The service cost is specified by the function $C(S)$, which yields the cost of each state in S. Each transition is annotated with a probability value indicating the likelihood of its occurrence during possible service invocations. The transition probability matrix denotes the set of all probabilistic distributions over the state space.

4.3 Service Workflow Transformation Rules

As one of the service workflow implementations, BPEL4WS is a way to define an execution process based on Web services. In practice, the user would prefer to use BPEL4WS[4], rather than the formal model. The following XML tags are used to provide the rules to transform BPEL4WS into a formal model.

The activity <*invoke*> consists of two sequential transitions, including the message send activity <*reply*> and receive activity <*receive*>, which can be mapped to input and output actions, respectively. The <*link*> activity is a sequence transition execution. The <*switch*> activity is a multi-way conditional branch. The <*pick*> activity combines a <*switch*> activity applied to various sequences of other activities with a condition such as <*if*> or <*while*>. Thus, the <*pick*> translating activity can obtain automata branches. The <*flow*> activity encompasses all the transitions, where sub-activities are executed concurrently. Extracting the <*flow*> activity returns the parallel composition of Web services. The time restriction of <*flow*> activity can be extracted as the guard to transition and invariant condition. The <*assign*> activity can represent an update function for a transition [18].

Rule 1. Each basic <*invoke*> activity is mapped to a state of the *FWS* model. Each < *link* > transition among the BPEL4WS activities is mapped to a transition of the *FWS* model. The special activities <*reply*> and <*exit*> are mapped to a state and a self-transition, respectively, and represent terminal states because they are the ending condition.

Rule 2. The <*sequence*> activity composes sub-activities into a sequence transition. For example, Figure 3 shows are three activities in a sequence structure. According to the sequence transition rule, it returns $S=\{A, B, C\}$ and $\delta=\{(A,B), (B,C)\}$.

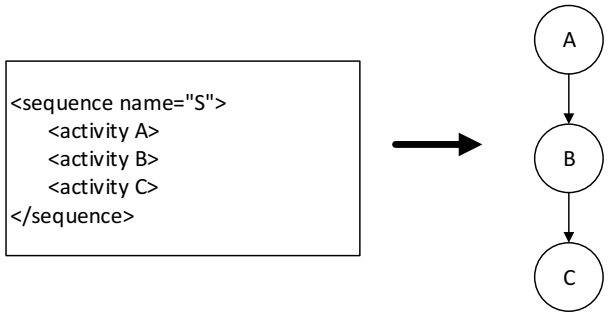
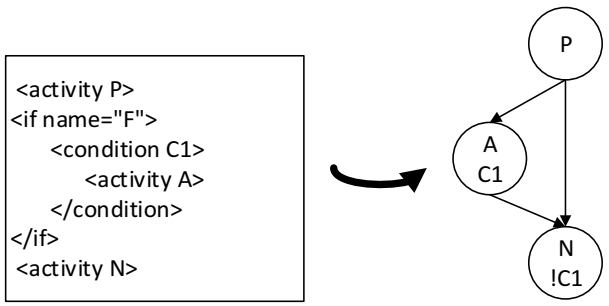


Figure 3. Activity rxample of <sequence>

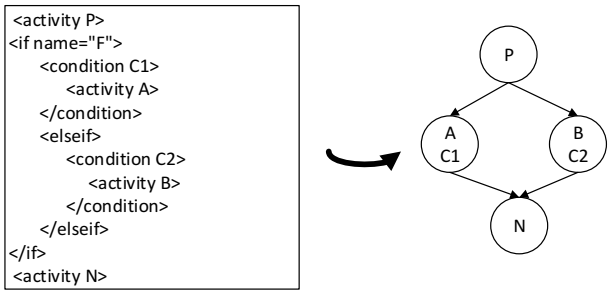
Rule 3. The activities <*if*> and <*switch*> are mapped to conditional branches. For example, in Figure 4, conditions C1 and C2 are extracted as atomic propositions to states *A* and *B,* respectively. Condition !C1 is extracted as an atomic proposition to state *N.* In Figure 3(a), $S=\{P, A, N\}$, $AP=\{C1, !C1\}$, $L(A)=\{C1\}$, $L(N)=\{!C1\}$ and $\delta=\{(P,A), (A,P), (P,N)\}$. In Figure 3(b), $S=\{P, A, B, N\}$, $AP=\{C1, C2\}$, $L(A)=\{C1\}$, $L(B)=\{C2\}$ and $\delta=\{(P,A), (P,B), (A,N), (B,N)\}$.

[3] PRISM. http://www.prismmodelchecker. org

[4] BPEL4WS. http://ode.apache.org/ws-bpel-20.html



(a)



(b)

Figure 4. Activity rxample of <if> and <switch>

Rule 4. The <*while*> activity represents a loop structure. The condition is mapped to atomic propositions of the first state in the loop structure. The opposite condition is mapped to atomic propositions of the break state at the loop exit. For example, in Figure 5, the conditions C and !C are changed into an atomic proposition. C is used to continue the loop, and !C is used to exit the loop. Thus, S={*P, A, B, N*},δ={(*P,A*), (*A,B*), (*B,A*), (*B,N*), (*P,N*)}, AP={*C*, !*C*}, and $L(A)$={*C*}, $L(N)$={!*C*}.

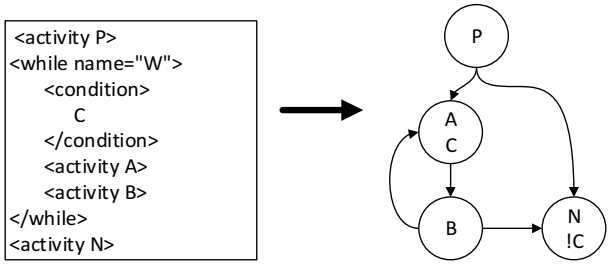
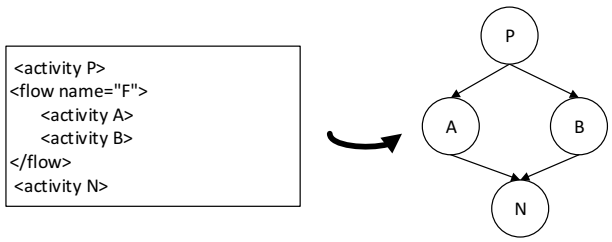
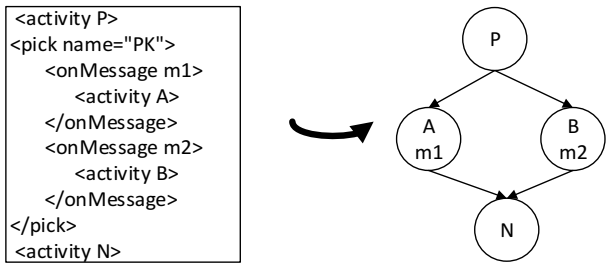


Figure 5. Activity rxample of <while>

Rule 5. The activities <*flow*> and <*pick*> are used for concurrent processing and changed into branches according to different labels. For example, in Figure 6(a), the <flow> activity shows two transitions from state P to states A and B in which S={*P,A,B,N*} and δ={(*P,A*), (*P,B*), (*A,N*), (*B,N*)}. However, Figure 6(b) is more complex; the activity <pick> carries messages that select its successor activities. Thus, the activity <pick> considers a message as an atomic proposition to each state, where S={*P,A,B,N*}, AP={*m1*,*m2*}, $L(A)$={*m1*},$L(B)$={*m2*} and δ={(*P,A*), (*P,B*), (*A,N*), (*B,N*)}.



(a) Activity rxample of <flow>



(b) Activity rxample of <pick>

Figure 6.

4.4 Verification Property Generation

Definition 11 (Workflow Verification Property). The verification property is notated as Probabilistic Computation Tree Logic (PCTL) [19], whose syntax is defined as follows:

$$\varphi ::= true \mid false \mid a \mid \varphi \wedge \varphi \mid \varphi \quad \varphi \mid \varphi \rightarrow \varphi \mid \neg \varphi \mid$$
$$P_{\sim p}[X\varphi] \mid P_{\sim p}[\varphi \mathbf{U} \varphi] \mid P_{\sim p}[\varphi \mathbf{U}^{\leq k} \varphi] \mid P_{\sim p}[F\varphi] \mid P_{\sim p}[G\varphi]$$

where $\sim \in \{<, \leq, >, \geq\}$, $0<p<1$ is a probability bound or threshold, a is an atomic proposition, φ and φ are formulae, and $k \in \mathbb{N}$ is denoted as time steps.

The symbol **P** is the probability operator, and the symbols **X**, **F**, **G** and **U** are temporal operators, meaning "ne**X**t state", "**F**uture state", "**G**lobal state" and "**U**ntil", respectively. We also use $P_{\sim p}(\phi)$ and $P_{=?}(\phi)$. The operator $P_{\sim p}(\phi)$ yields a *true* value when the probability of a path formula ϕ being true in state satisfies the bound $\sim p$; otherwise, it outputs *false*. The operator $P_{=?}(\phi)$ returns a probability value for the given path formula. For example,

- $P_{<0.15}$(**F** state=hunger): with a probability of 0.15 or less, the hunger state will be visited eventually.
- $P_{\geq 0.9}$ (X selection): user selection will operate at a probability of 0.9 or higher.
- $P_{=?}$(**F** state=bad): represents the probability of encountering a bad state.

The verification property also provides a reward command to denote cost computing, that syntax is defined as follows:

$$\varphi ::= true \mid false \mid a \mid \varphi \wedge \varphi \mid \varphi \quad \varphi \mid \varphi \rightarrow \varphi \mid \neg \varphi \mid$$
$$R_{\sim p}[X\varphi] \mid R_{\sim p}[\varphi \mathbf{U} \varphi] \mid R_{\sim p}[\varphi \mathbf{U}^{\leq k} \varphi] \mid R_{\sim p}[F\varphi] \mid R_{\sim p}[G\varphi]$$

where the reward operator R includes bound

computing and query computing. For example,

- $R_{\leq 5}$ (*true* $\mathbf{U}^{\leq 3}$ *login*): the reward of a successful login in 3 or fewer tries is less than or equal to 5.
- $R_{=?}$ (*true* **U** *fail*): represents the reward for reaching a failed state.

Thus, using the probability operator **P** and the reward operator **R,** it is possible to specify cost and probability-related properties for verification requirements. In our study, coverage criteria such as node coverage and transition coverage are employed to generate these properties from an abstract workflow based on the workflow requirements in Definition 4. For instance, global requirements about χ^R and χ^C can be used, which are generated as reachability properties.

4.5 Model Checker PRISM and Its Language

PRISM is a probabilistic model checker that supports various model types, including Discrete Time Markov Chain (DTMC), Markov Decision Process (MDP), Continuous Time Markov Chain (CTMC) and Probabilistic Timed Automat (PTA). In this paper, we mainly use DTMC to code the services composition model. PRISM's DTMC syntax is in the form of action-guard commands that force two or more modules to make transitions simultaneously with different probabilities.

[]<location>→<prob>:<location'>+
…+<prob>:<location'>

where *location* is a label corresponding to a state of the *FWS* model, and *prob*∈[0,1] is the probability function after changing the current *location* to a new *location'*.

To extend the probability for each transition, the reliability of the mapped service is used to generate the transition probability. The transition probability is calculated as follows.

$$p(s,s') = \frac{r(s')}{|\{(s,s'') \mid \exists (s,s'') \in \delta\}|}. \quad \textbf{(17)}$$

For example, in Figure 7(a), suppose state s0 has two ongoing transitions (s0, s1) and (s0, s2). According to formula (17), the new transition probability for (s0, s1) is 0.91/2=0.405, and (s0, s2) is 0.95/2=0.425. In Figure 7(b), the special state *fail* denotes the failure probability when states s1 and s2 are unavailable.

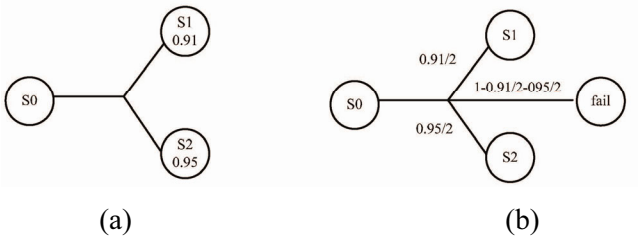


Figure 7. Transition probability example

PRISM supports the reward (or, equivalently, cost) structure that assigns real values to states. The reward expression is as follows.

rewards "name"
exp_1
....
exp_n
endrewards

where each *exp_i* item describes an expression in the form of <guard>:<reward>, where *guard* is a condition and *reward* is a value for cost.

For example, in Figure 8, states s=0, s=1, and s=2 have been denoted with costs. However, they belong to a different cost definition. Thus, we define two reward expressions, that is, cost r1 and cost r2. Under cost r1, if state s=0 is visited, then the cost r1 is incremented by 1.2. If state s=1 is visited, then cost r1 is incremented by 2.8. Under cost r2, if state s=2 is visited, then cost r2 is incremented by 0.8.

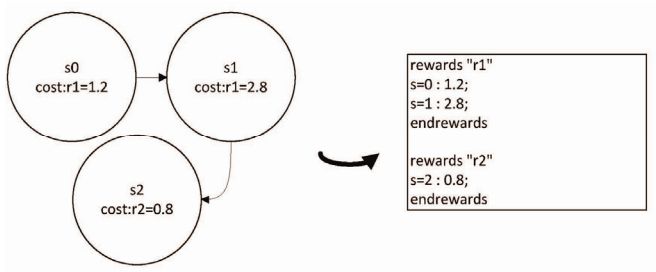


Figure 8. Reward example of service cost

5 Experiments

In this section, we focus on the efficiency and effectiveness of our method. We have implemented the proposed algorithms and transformation rules described previously in Java to select workflow services, and integrated PRISM to verify the service workflow quantitatively. All the experiments were conducted on a x230i ThinkPad PC with a 2.50 GHz Intel Core i3-3120 CPU and 3.23 GB of main memory running Windows 7.

5.1 Data Preparation

Because no standard experimental platform and test dataset exists, we designed workflows for device management derived from a real system in the Equipment Office of Shanghai University. The procedural steps to prepare the test data and the test scenario include the following:

(1) WSDL4J is used to generate 200 Web services; it can automatically code the WSDL for services after the input and output parameters have been configured. Each service has, at most, two inputs and two outputs.

(2) To increase the number of services, the names of services generated in Step 1 were randomly changed. The total number of Web services was close to 500.

(3) Each service is assigned cost and reliability values, where cost ranges from 1 to 100, and reliability (as a probability) ranges from 0 to 1.

(4) The workflow language BPEL4WS is extended with cost and reliability descriptions to express user requirements.

Service composition starts with the BPEL4WS created in step 4), and aims to map each logic task to suitable services. Then, service selection is performed to search for matching services in the test dataset created in Step 1. For example, Figure 1 shows a BPEL4WS description and Table 1 shows a service repository. As shown in Figure 2, service composition will return different types of composition plans.

We focus on node number, which is the number of nodes in a service workflow, and on candidate service number, which is the number of candidate services for each node in a workflow. To examine the impact of these two parameters on the cost and reliability performance of the generated service workflow, we list four sets of parameters in Table 4. During the experiments, one of these two parameters is varied while the other parameter is fixed.

Table 4. Experiments setting

ID	Node Number	Candidate Service Number
1	5	7
2	10	30
3	15	50
4	20	100

A series of experiments was conducted to evaluate the performance among different methods. The first contrast experiment compared our service selection method with the traditional keyword matching method. The second experiment involved using PRISM to check the service workflow to reveal the error discovery rate.

5.2 Performance Analysis

In the first set of experiments, we implemented the service index method and compared it with the traditional keyword matching method. We recorded response times and memory consumption.

Figure 9 shows the time consumption when varying the number of nodes from N=5 to 20 to test the response time. The unit of measurement is ms. The service index method (SIM) has a faster response time than the traditional keyword matching (TKM) method. As the number of nodes increases, the TKM response time increases. However, SIM response time is almost a straight line, and requires less service search time because the service index bounds the search to a more accurate scope. In contrast, TKM must search the full service name space, causing increased computational time.

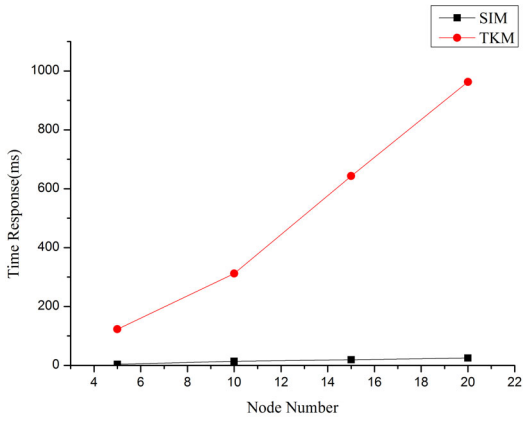


Figure 9. Response time of service selection

Figure 10 shows the memory consumption when varying the number of candidate services from N=7 to 100. The unit of measurement is bytes. As the number of candidate services increases, the memory consumption of TKM increases sharply. The service index method (SIM) maintains a lower memory consumption than does the traditional keyword matching (TKM) method because TKM spends more time searching for possible services among all the services in the repository. Thus, we can conclude that TKM requires more memory to complete the search tasks.

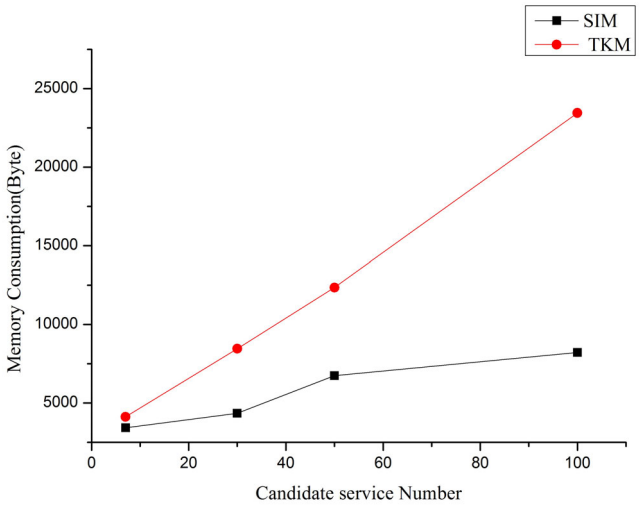


Figure 10. Memory consumption of service selection

In the second set of experiments, we used PRISM to check the service workflows. PRISM checks each service workflow plan to find bugs.

Table 5. Capacity of "Bad" service discovery

	#7	#30	#50	#100
PMC	5	12	34	73
MC	3	5	13	27
WT	1	1	3	3

Before starting these experiments, a number of 'bad' services with lower cost and reliability were

purposefully inserted as traps. The header row of Table 5 indicates the different workflow paths, #7, #30, #50 and #100. The probabilistic model-checking method (PMC) is compared with the model-checking (MC) method and workflow test (WT). The model check is performed using NuSMV. The workflow test is conducted manually. Table 5 shows the ability to discover 'bad' services under different mode sizes.

Figure 11 shows the comparison of 'bad' service discovery. Obviously, the WT has difficulty finding any 'bad' services because it uses function-oriented testing, which focuses only on input and output tests without considering temporal logic behaviours. Under the same mode size, MC is weaker than PMC for discovering 'bad' plans. Although model checking can check temporal logic behaviours, it lacks quality computing. The probabilistic model checking evaluates the business process in a quantitative way to find 'bad' services that have low performance. Thus, our method can efficiently scale up to large-scale datasets from both theoretical and practical perspectives.

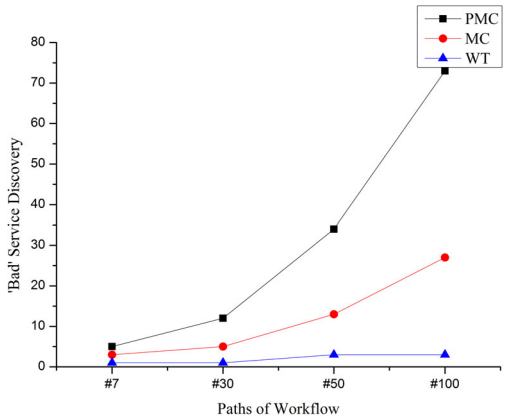


Figure 11. Comparison of "Bad" service discovery

6 Related Works

Web services are changing traditional business models because they help modern enterprises to seize business opportunities and accelerate inter-enterprise collaborations. Much of the research on service composition has been published from different perspectives. We present a review of the major techniques and methods most closely related to our work.

Considerable research has focused on service selection. Wang et al. [20] proposed algorithms for QoS-aware service selection based on the artificial bee colony (ABC) algorithm. Yue et al. [21] proposed a skyline-based Web service selection method to address the efficiency problem as well as to solve the frequent requests problem. To improve service selection efficiency, the KD-tree-based search algorithm was designed to determine the skyline and later reduce the search space. Wang et al. [22] proposed a type of incentive contract that motivated the service providers to offer higher QoS at lower prices. An incentive mechanism for effective service selection was proposed to fulfil global QoS requirements. Dionisis et al. [23] considered a number of service selection issues related to the WS-BPEL scenario adaptation, aiming to enhance the adaptation quality and improve the QoS offered to end users. However, none of the existing methods considered service search in conjunction with service recommendation to improve service selection efficiency.

Considerable research has also focused on service cost. Yu et al. [24] proposed a backwards composition context-based service selection approach for service composition. These researchers considered several contextual factors, including cost policy and composition time, during service selection. Marco et al. [25] focused on the definition of cloud service compositions driven by certified non-functional properties. They defined a cost evaluation methodology to provide a composition that minimized the total costs of a cloud provider. Xiao et al. [26] proposed a novel process algebra called PTPA that incorporated both price and cost. They presented the syntax and semantics of PTPA and proposed an algorithm to construct a cost state space to select services composition with optimal cost. Robson et al. [27] proposed a solution to analyse the costs of service compositions by considering service reliability and all classes of cost behaviours. David et al. [28] presented an approach for integrating user preferences concerning completion time and workflow accuracy in a workflow composition system. Philipp et al. [29] formalized the problem of searching the optimal set. These researchers presented algorithms to solve the complex optimization problem. These methods are useful for including cheaper services during service composition. However, service workflow performance involves non-functional requirements and should also consider QoS features.

Another large portion of the research has focused on service QoS. Ding et al. [30] addressed the issues of selecting and composing services using the genetic algorithm and proposed a transaction and QoS-aware selection approach. Guidara et al. [31] presented a heuristic-based time-aware service selection approach to efficiently select a close-to-optimal combination of services. Yin et al. [32] proposed a data filtering-extended SlopeOne model (filtering-based CF), which is based on the characteristics of a mobile service and considers location when predicting QoS values. Zeng et al. [33] presented a middleware platform that, in a way. addressed the service selection for the purpose of composition. It aimed to maximize user satisfaction expressed as utility functions over QoS attributes. Deng et al. [34] proposed a novel method of service selection called the correlation-aware service pruning

(CASP) method. It managed QoS correlations by accounting for all services that could be integrated into optimal composite services and pruned suboptimal candidate services. Tao et al. [35] designed a broker-based architecture to facilitate the selection of QoS-based services. The objective of service selection was to maximize an application-specific utility function under end-to-end QoS constraints. However, these methods are difficult to use for dynamic service composition because their service selections require human intervention to make decisions during QoS computing. Moreover, QoS is a collection of multidimensional indexes that show how to evaluate service composition, while QoS values should be translated into a unified evaluation index.

In contrast to the existing research described above, we propose a two-phase services composition approach for abstract workflows. In the first phase, service selection is performed through service search and service recommendation. In the second phase, probabilistic model checking is employed to determine which plan of service configurations for workflow is most optimal by considering low cost and high reliability.

7 Conclusions

Service workflow, as one method for integrating enterprise information, has been widely used in e-commerce and scientific computing. However, because of the uncertain Web service environment, workflows have failure risks that cause component services to be unavailable. To guarantee that a service workflow satisfies both functional and non-functional requirements, it is necessary to study cost-driven services composition. The goal of this paper is to select services that meet the demands of abstract workflows by using probabilistic model checking to quantitatively verify the service plan.

First, the inverted index method is used to generate a service index to improve service search efficiency. Next, the service search selects services based on the interface operations of a user's functional requirements. Furthermore, candidate service sets matching the abstract workflow will be returned for service configuration. Third, usage frequency is used to generate a service-service correlation matrix by applying an improved Pearson formula that considers interface, cost, and reliability factors to recommend correlated services. Fourth, transformation rules for changing BPEL4WS into a formal model are discussed, and the PCTL formula is introduced to specify quantitative properties. Finally, the PRISM model checker is employed to perform a formal verification whose result helps to identify the plan of service configuration most suitable for the current workflow.

This paper discusses only cost-driven services composition in an uncertain environment. However, in practice, service invocations have time limitations [36]. Thus, in future work, we plan to extend the services composition verification to consider time constraints.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China under Grant No. 61502294, 61662021, and the IIOT Innovation and Development Special Foundation of Shanghai.

References

[1] J. J. Hu, X. L. Chen, Y. Y. Cao, L. H. Zhu, A Comprehensive Web Service Selection Algorithm on Just-in-Time Scheduling, *Journal of Internet Technology*, Vol. 17, No. 3, pp. 495-502, May. 2016.

[2] J. Yin, X. Zhao, Y. Tang, C. Zhi, Z. Chen, Z. Wu, CloudScout: A Non-Intrusive Approach to Service Dependency Discovery, *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 5, pp. 1271-1284, May, 2017.

[3] G. S. Fan, H. Q. Yu, Q. Wu, L. Q. Chen, D. M. Liu, A Requirement-Driven Method for Secure and Reliable Web Service Composition, *Journal of Internet Technology*, Vol. 14, No. 3, pp. 485-496, May. 2013.

[4] W. Jiang, S. Hu, D. Lee, Continuous Query for QoS-aware Automatic Service Composition, *IEEE International Conference on Web Services (ICWS 2012)*, Honolulu, USA, 2012, pp. 50-57.

[5] H. H. Gao, D. Q. Chu, Y. C. Duan, The Probabilistic Model Checking Based Service Selection Method for Business Process Modeling, *Journal of Software Engineering and Knowledge Engineering*, Vol. 27, No. 6, pp. 897-923, August, 2017.

[6] H. H. Gao, Y. C. Duan, H. K. Miao, An Approach to Data Consistency Checking for the Dynamic Replacement of Service Process, *IEEE ACCESS*, Vol. 2017, No. 5,11700-11711, June, 2017.

[7] L. Q. Yang, G. S. Kang, L. P. Guo, Z. Y. Tian, L. Zhang, X. N. Zhang, X. Gao, *Process Mining Approach for Diverse Application Environments, Ruan Jian Xue Bao/Journal of Software*, Vol. 26, No. 3, pp. 550-561, March, 2015.

[8] W. M. Zhang, C. C. Liu, Z. G. Luo, A Review on Scientific Workflows, *Journal of National University of Defense Technology*, Vol. 33, No. 3, pp. 56-65, June, 2011.

[9] X. X. Yang, T. Yu, H. H. Xu, A Novel Framework of Using Petri Net to Timed Service Business Process Modeling, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 26, No. 4, pp. 633-652, May, 2016.

[10] C. H. Lee, S. Y. Hwang, I. L. Yen, A Service Pattern Model for Service Composition with Flexible Functionality, *Information Systems and E-Business Managemen*t, Vol. 13, No. 2, pp. 235-265, May, 2014.

[11] S. Y. Hwang, H. J. Wang, J. Tang, J. Srivastava, A

Probabilistic Approach to Modeling and Estimating the QoS of Web-services-based Workflows, *Information Sciences*, Vol. 177, No. 23, pp. 5484-5503, December, 2007.

[12] P. Podili, K. K. Pattanaik, P SinghRana, BAT and Hybrid BAT Meta-heuristic for Quality of Service-based Web Service Selection, *Journal of Intelligent Systems*, Vol. 26, No. 1, pp. 123-137, January, 2017.

[13] C. Gerardo, D. P. Massimiliano, E. Raffaele, A Famework for QoS-aware Binding and Re-binding of Composite Web Services, *Journal of Systems and Software*, Vol. 81, No. 10, pp. 1754-1769, October, 2008.

[14] Y. Y. Yin, F. Z. Yu, Y. S. Xu, L. F. Yu, J. L. Mu, Network Location-Aware Service Recommendation with Random Walk in Cyber-Physical Systems, *Sensors*, Vol. 17, No. 9, pp. 2059-2079, September, 2017.

[15] Y. Y. Yin, A. H. Song, M. Gao, QoS Prediction for Web Service Recommendation with Network Location-Aware Neighbor Selection, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 26, No. 4, pp. 611-632, May, 2016.

[16] Y. C. Jiang, J. X. Liu, M. D. Tang, An Effective Web Service Recommendation Method Based on Personalized Collaborative Filtering, *2011 IEEE International Conference on Web Services (ICWS 2011)*, Washington, DC, 2011, pp. 211-218.

[17] Z. Zheng, H. Ma, M. Lyu, I. King, WSRec: A Collaborative Filtering Based Web Service Recommender System, *IEEE International Conference on Web Services (ICWS 2009*), Los Angeles, CA, 2009, pp. 437-444.

[18] H. H. Gao, Y. Li, Generating Quantitative Test Cases for Probabilistic Timed Web Service Composition, *6th IEEE Asia-Pacific Services Computing Conference (APSCC2011)*, Jeju island, South Korea, 2011, pp. 275-283.

[19] M. Kwiatkowska, G. Norman, D. Parker, Probabilistic Model checking in Practice: Case Studies with PRISM, *ACM SIGMETRICS Performance Evaluation Review*, Vol. 32, No. 4, pp. 16-21, March, 2005.

[20] X. Z. Wang, X. F. Xu; Q. Z. Sheng, Z. J. Wang, L. Yao, Novel Artificial Bee Colony Algorithms for QoS-Aware Service Selection, *IEEE Transactions on Services Computing*, Vol. PP, No. 99, 1-1. September, 2016.

[21] Y. Wang, Y. Song, M. Y. Liang, A Skyline-based Efficient Web Service Selection Method Supporting Frequent Requests, *IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2016)*, Nanchang, China, 2016, pp. 328 -333.

[22] P. W. Wang, X. Y. Du, QoS-aware Service Selection Using An Incentive Mechanism, *IEEE Transactions on Services Computing*, Vol. PP, No. 99, 1-1, August, 2016.

[23] D. Margaris, P. Georgiadis, C. Vassilakis, On Replacement Service Selection in WS-BPEL Scenario Adaptation, *IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA 2015)*, Rome, Italy, 2015, pp. 10-17.

[24] H. Yu, S. Reiff-Marganiec, A Backwards Composition Context based Service Selection Approach for Service Composition, *International Conference on Services Computing SCC 2009)*, Bangalore, India, 2009, pp. 419-426.

[25] M. Anisetti, C. A. Ardagna, E. Damiani, F. Gaudenzi, A Cost-Effective Certification-Based Service Composition for the Cloud, *IEEE International Conference on Services Computing (SCC 2016)*, San Francisco, CA, 2016, pp. 58-65.

[26] F. X. Xiao, H. Q. Min, B. Xu, C. Q. Jiang, G. E. Xia, Modeling Cost-aware Services Composition Using a Priced Formal Method, *The 6th IEEE International Conference on Software Engineering and Service Science (ICSESS 2015)*, Beijing,China, 2015, pp. 309-312.

[27] R. W. A. de Medeiros, N. S. Rosa, L. F. Pires, Predicting Service Composition Costs with Complex Cost Behavior, *2015 IEEE International Conference on Services Computing (SCC 2015)*, New York City, NY, 2015, pp. 419-426.

[28] D. Chiu, G. G. Agrawal, Cost and Accuracy Aware Scientific Workflow Composition for Service-Oriented Environments, *IEEE Transactions on Services Computing*, Vol. 6, No. 4, pp. 470-483, October, 2013.

[29] P. Leitner, W. Hummer, S. Dustdar, Cost-Based Optimization of Service Compositions, *IEEE Transactions on Services Computing*, Vol. 6, No. 2, 239-251, April, 2013.

[30] Z. J. Ding, J. J. Liu, Y. Q. Sun, C. J. Jiang, M. C. Zhou, A Transaction and QoS-Aware Service Selection Approach Based on Genetic Algorithm, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 45, No. 7, pp. 1035-1046, July, 2015.

[31] I. Guidara, N. Guermouche, T. Chaari, S. Tazi, M. Jmaiel, Heuristic Based Time-Aware Service Selection Approach. *2015 IEEE International Conference on Web Services ICWS 2015)*, New York, NY, 2015, pp. 65-72.

[32] Y. Y. Yin, W. T. Xu, Y. S. Xu, H. Li, L. F. Yu, Collaborative QoS Prediction for Mobile Service with Data Filtering and SlopeOne Model, *Mobile Information Systems*, Vol. 2017, No. 3, pp. 1-14, June, 2017.

[33] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-Aware Middleware for Web Services Composition, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp. 311-327, May, 2004.

[34] S. G. Deng, H. Y. Wu, D. N. Hu, J. L. Zhao, Service Selection for Composition with QoS Correlations, *IEEE Trans. Services Computing*, Vol. 9, No. 2, pp. 291-303, March, 2016.

[35] T. Yu, Y. Zhang, K. J. Lin, Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints, *ACM Transactions on the Web*, Vol. 1, No. 1, pp. 1-26, May, 2007.

[36] C. C. Xiang, P. L. Yang, X. G. Wu, H. He, S. C. Xiao, QoS-Based Service Selection with Lightweight Description for Large-scale Service-oriented Internet of Things, *Tsinghua Science & Technology*, Vol. 20, No. 4, pp. 336-347, August, 2015.

Biographies

Honghao Gao received the Ph.D. degree in Computer Science and started his academic career at Shanghai University in 2012. He is an IET Fellow, BCS Fellow, EAI Fellow, IEEE Senior Member, CCF Senior Member, and CAAI Senior Member. Prof. Gao is currently a Distinguished Professor with the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, China. His research interests include service computing, model checking-based software verification, wireless network and IoT, and sensors data application.

Wanqiu Huang is M.S. Degree candidate in Computer Science with the School of Computer Engineering and Science, Shanghai University, Shanghai, China. Her research interests include Web service and model checking.

Yucong Duan received the PhD in software engineering from Institute of Software, Chinese Academy of Sciences, China, in 2006. He is currently a Professor and vice director of Computer Science Department, Hainan University, China.His research interests include: theoretical and empirical software engineering, model driven software development, etc.

Xiaoxian Yang, received the Ph.D. degree in Management Science and Engineering from Shanghai University, Shanghai, China, in 2017. She is currently an assistant professor at Shanghai Polytechnic University, China. Her research interests include business process management, and formal method.

Qiming Zou, received the Ph.D. degree in Machine Manufacturing from Shanghai University, Shanghai, China, in 2015. He is currently an assistant professor at Shanghai University, China. His research interests include cloud computing, grid computing computer aided manufacturing.