Information Technology Journal 10 (9): 1662-1672, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1662.1672
© 2011 Asian Network for Scientific Information

Reliable Web Services Selection Based on Finite State Machine Model

Hongjie Shen, Zhijun Ding and Hongzhong Chen
The Key Laboratory of Embedded System and Service Computing,
Tongji University, Ministry of Education, Shanghai 200092, China

Abstract: Dynamically selecting suitable Web Services (WSs) is crucial to users in Web Services Composition (WSC). Generally, most works regard a Web Service (WS) as the basic unit and compose the composite WS (CWS) end to end. However, a WS may comprise multiple operations that are invoked in sequence and the selection of WSs according to the model still is a NP problem. In this study, the WSs consist of some operations and then WSs selection problem is formalized as a Finite State Machine (FSM) process. This study proposed an algorithm to create the WSC Tree (WSCT), in which each path from root to leaf node is a feasible WSC execution path. Then, a heuristic algorithm is proposed to realize the selection of WSs based on the WSCT. The advantage of heuristic algorithm is that it can address the selection problem without traversing the whole WSCT but it still spends much time in the construction of the WSCT. So, in order to further decrease the time of selection, a backtracking algorithm is presented to select a feasible execution path without generating WSCT. At last, experiments show that heuristic algorithm is more effective than exhaustive method and the backtracking algorithm spend the less time than heuristic algorithm, the reliability of WSC execution path selected by backtracking algorithm can satisfy the requirement of users. So, this study proposed an algorithm to select a feasible WSC execution path within a reasonable time.

Key words: Web services, service composition, algorithm, service selection, reliability

INTRODUCTION

A WS is a software system identified by a Uniform Resource Locator (URL), whose public interfaces and bindings are defined and described using XML-based specifications (or standards). Its definition can be discovered by other software systems. These systems may then interact with the WSs in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols (Booth *et al*., 2004). In order to create the actual application, developers may compose some existing simple services provided by the system to resolve complex user problems. In the process of WSC, suitable candidates must be discovered and from them the most suitable ones must be selected (Shen *et al*., 2010).

Generally, the Web service composition can be viewed as a three-step process: (1) composite Web service specification, (2) selection of the component Web services and (3) execution of the CWS.

At the first step, the user submits the goal he/she wants the composite service achieves, along with some constraints and preferences that need to be satisfied (Ding *et al*., 2005, 2007). Most works regard a WS as the basic unit for composition. However, a WS may comprise multiple operations such that their invocation sequence is constrained. During the second step, component WSs fulfilling the user's goal are selected among a set of available services (Haddad *et al*., 2010). With more and more WSs with similar or identical functionality become available, the Quality of Service (QoS) attributes (e.g., availability, response time and throughput) are subsumed as non-functional attributes. But when more and more WSs become available, WSs selection problem is a complex NP-hard optimization problem. At the third step, execute all selected component Web services.

In order to define the possible orders of calling WSs at design time, many composition methods also have been proposed. Current service composition approaches include practical languages, such as BPEL, WSCL and OWL-S and commercial service platforms or products, such as the Sun ONE framework based on J2EE, Microsoft.NET, the Oracle BPEL Process Manager, the HP WSs Management Platform and the IBM WebSphere Application Server. At the same time, several formal process models have also been proposed, including FSM, Petri net, UML, activity diagram and process algebra. They are compared among each other with respect to some key requirements, including composition correctness, automatic composition and scalability (Ding *et al*., 2008). In fact, most works regard a Web

Corrresponding Author: Zhijun Ding, The Key Laboratory of Embedded System and Service Computing, Tongji University, Ministry of Education, Shanghai 200092, China

service as the basic unit and selection with end-to-end for composition, however, a WS may comprise multiple operations that are invoked in sequence (Hwang *et al*., 2008). By the comparison, FSM is a promising model and suitable to address most of the aforementioned requirements issues. Moreover, a FSM is a formalism that is suitable to describe reactive behaviors and has the notion of states which is useful for monitoring service executions (Yu *et al*., 2007). So, some works model Web services as finite state machine.

Hu and Wang (2004) first model WSs as FSMs and then in order to avoid some problems as deadlock, they deliver the process of composition into three parts according to the merging process: constraints before composition, constraints in composition and constraints after composition. Hwang *et al*. (2008) model WSs and CWS as FSMs, then propose a metric, called aggregated reliability, to measure the probability that a given state in a composite WS will lead to successful execution in an error-prone environment. Although they propose a powerful method to compute the aggregated reliabilities, when the number of atomic WSs becomes huge, calculating the all aggregated reliability is impractical, because it may take too much time. Berardi *et al*. (2003) model WSs as Finite State Automations (FSA). In order to take time constraints into account, then they add a time into the tuple of the FSA. At last, a new XML-based language, namely WSTL (Web Service Transaction Language) that integrates well with standard languages in order to completely specify Wss is presented. Considering automatic Web services composition is the ultimate goal of most composition efforts, so, based on their previous work, Berardi *et al*. (2004) present a framework that describes a Web service's behavior as an execution tree and then translates it into a FSM. They propose an algorithm that checks a composition's existence and returns one if it exists. In the process, the composition is proved correct and the algorithm's computational complexity characterization is given, ensuring that the automatic composition will finish in the finite number of steps. But, the authors cannot further research the WSC selection problem based on the model.

Although these articles have modeled Web services as FSMs, only Hwang *et al*. (2008) introduced a selection method to WSC based on a criterion of QoS, reliability. However, they ignore that the number of atomic WSs is huge or the WSs may comprise multiple operations will induce WSC selection become inefficient. In other words, when we model a WS as multiple operations that are invoked in sequence, we still need to consider how to select a feasible CWS to execute within reasonable time. At present, some articles focus on the selection optimization problem of WSC.

Zeng *et al*. (2004) proposed a method to compute an optimal set of WSs for each possible execution path in the process based on a weighted combination of QoS measures, including: price, duration, reputation, availability and successful rate. Although integer programming is utilized to accelerate the computation, with the number of possible execution paths become huge, the selection process may become impractical.

Yu *et al*. (2007) model the WSC selection in two ways: combinatorial model and graph model. The combinatorial model defines the problem as a Multidimension Multichoice 0-1 Knapsack Problem (MMKP). The graph model defines the problem as a Multiconstraint Optimal Path (MCOP) problem. Efficient heuristic algorithms for service processes of different composition structures are presented in this article. At last, this study proposed a broker-based architecture to select the QoS-based services.

Different articles have different methods for addressing the selection problem. Jaeger *et al*. (2005) draw a conclusion to some selection algorithms, including: Greedy Selection, Discarding Subsets, Bottom-Up Approximation and Pattern-wise Selection. But because of the different WSC model, these algorithms are not suitable to the problem above-mentioned.

This study extends our previous work presented by Shen *et al*. (2010), where firstly an algorithm is proposed to create the WSCT. Then, a heuristic algorithm is presented to complete the selection of WSs based on the given WSCT. According to WSCT features, an effective heuristic function is designed. Moreover, an approximate solution is generated to get an execution path of WS composition within a reasonable period of time and the reliability of the execution path is feasible to user.

In this study, with the number of available WSs becomes huge, constructing the WSCT will spend much time. So, in order to further decrease the time of selecting the execution path, a backtracking algorithm is proposed to select a feasible execution path. According to the CWS and the WS community, a feasible execution path can be selected without constructing the whole WSCT to satisfy the requirement of users in reasonable time. And some improvements are made in the heuristic algorithm in the previous work.

PROBLEM DESCRIPTION

The selection problem about WSC has become a focus in the filed. Generally, most works regard a WS as the basic unit and compose CWS end to end, then, according to various QoS measures, such as response time, cost and reliability, users can select the suitable WSs (Zeng *et al*., 2004; Yu *et al*., 2007). However, a WS

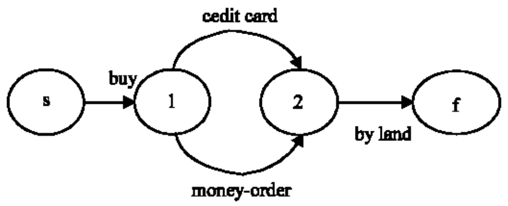

Fig. 1: A composition problem: buy a PC service

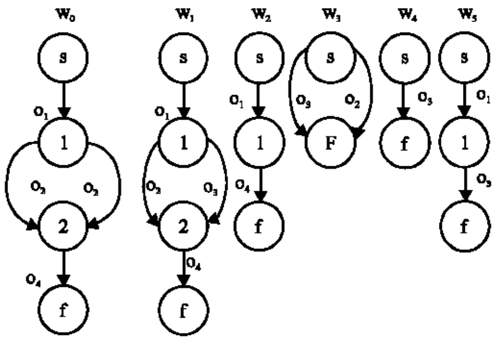

Fig. 2: Composite WS and WS community

may comprise multiple operations in practice that are invoked in sequence. So, the WS can be modeled as a FSM with tuples including state, operations and others (Hwang *et al*., 2008).

The state of a service (atomic service or CWS) describes the changes of its behavior and the state can transfer by the invocation of the operation. Depending on specific research topics, the state can be (a) the actual internal execution state, (b) only a part of the state of relevance to the parties connected with the WSs or (c) the state of the "external world". Furthermore, different models rely on different kinds of "operation" to change state, these might be (i) messages, (ii) activities and (iii) events (Gerede *et al*., 2004).

Here, two definitions are articulated for technical discussions which include a FSM approach to model a WS and the definitions of WSC. The definitions come from Hwang *et al.* (2008) which are not repeated here.

This study consider a CWS W_0, a WS community C consists of the atomic Wss and the WS community $C = \{W_1, W_2, ..., W_n\}$.

Suppose we intend to buy a Personal Computer (PC) in a commercial web site, we would like to develop a shopping service such that users apply to buy a PC and then pay for it, at last, the PC will be delivered to the users and we would like to be flexible on the payment. So, users can pay either by money order or by credit card, then users can deliver the PC by land, the example is shown in Fig. 1.

Here, a WS Community $C = \{W_1, W_2, W_3, W_4, W_5\}$ is given as shown in Fig. 2. W_0 is the buying a PC service in the Fig. 1. Consider the composition W_0 shown in Fig. 2, to select the delegation to invoke for the operation o_1, there are three choices: $W_1.o_1$, $W_2.o_1$ and $W_5.o_1$. Here, WSs consist of a sequence operations that is say, in the W_1, after the o_1, o_2 and o_3 are invoked successfully, the o_4 can be used and after all operations are invoked, the WS can be invoked.

In the Fig. 2, according to the WS composite W_0 and the atomic WS community C, we can get the composite WSs set to meet the user requirements, for example, the set $\{W_1\}$ can meet the composite WS W_0, at the same time, the set $\{W_2, W_3\}$ also meet the composite WS W_0. So, one suitable set need to be selected for user in the acceptable time according to the reliability criteria.

WEB SERVICES COMPOSITION AND SELECTION

WSCT: In fact, the Web service composition can be viewed as a three steps process, the first step is composite Web service specification. In this study, the composite Web service specification is to create the service composition tree and the general process of constructing the service composition tree by iterating following operations is: 1) search WSs whose inputs contain the given input, 2) attain outputs of the selected WSs, 3) search input matched to the selected output as the given input in the operation. At last, search the composition result from service composition tree (Chen *et al*., 2006).

According to the definition mentioned above, the general process of combination of services actually is to produce a tree and then select a suitable path to execute in order to meet the requirement. So, this study introduce the process of building the WSCT according to service requester's composite WS W_0 and candidate atomic WSs in the WS community. Different from the enumeration method by Chen *et al.* (2006), in this study, an algorithm is proposed to build a WSCT from back of WSs operation sequence. Due to operation set is a sequence in an atomic WS, only previous operations are selected to invoke, can the subsequent operations be selected which can avoid the exhaustion to all operations of WSs. So this study builds WSCT from back which can reduce search space slightly. For example, in Fig. 2, before operation o_4 in the W_1 is invoked by the application, operation o_2 or o_3 in the W_1 must be selected.

The main idea of algorithm 1 is as follows, search all possible nodes contain the given requirement from back of the Wss sequence and define a previous operations set for each candidate node. By the previous set of each candidate node algorithm 1 know whether the candidate node is a feasible node. When creating a WSCT, the node of WSCT includes one or more operations.

Here we give some signs introduction in the algorithm 1, S_{CBI} is a set to store nodes which can be invoked that is to say, their previous node have been invoked and you only can invoke the needful node from the S_{CBI} as the S_C. S_C is the candidate node set which include the nodes can be selected as part of the CWS. S_C is a subset of S_{CBI}. A node has previous nodes set S_P. When operations in the node will be invoked, the S_P of the node must be invoked firstly and the S_P can be used to judge whether the execution path is feasible at current state. If the S_P of the node can not be selected at next step, the S_P of the node must be added into the S_P of children of the node. S_{OS} [i] is the operation node set of the ith state in a WS. W_i. S_{OS}[i] is a node set of ith state in the W_i, for example, in W_1 operations set of the second state is $W_1.S_{OS}[2] = \{W_1.o_2, W_1.o_3\}$. Pop is the functions that gets a node from set or queue or stack and delete the node. GetPreviousOp is the function that gets the previous operation node. And root is the root node of the WSCT.

The description of algorithm 1 is as follows.

Initialization: Firstly, create a null node *root* as the root node of WSCT and push the root node of WSCT into the queue Queue; put all the last state operations of atomic WSs in the WS community into the S_{CBI}, set the S_P of root node of WSCT as null, i is state of CWS.

Step 1: Get a node from Queue as the parent node (line 2) and get all candidates as S_C (line 4 to 13), the process is: if the S_P of parent node is null, we do not need to consider the nodes in S_P, so, the S_C is the intersection of S_{CBI} and W_0. S_{OS} [i] (line 4 to 5), else we get the candidate nodes from S_P, so, S_C is intersection of S_P and W_0. S_{OS} [i] (line 6) and if the S_P have nodes but have no suitable candidate nodes (line 7), in other word, S_P of parent node will be used later, we still need to find the candidate nodes from S_{CBI}, so S_C is the intersection of the set of S_{CBI} and the set of W_0. S_{OS} [i] (line 8), meanwhile, the nodes in S_P of parent node must be added into the previous node set of candidate node (in fact, they are also children nodes of parent node) in order to be used later (line 9) and remove the unfeasible candidate nodes according to the previous set (line 10 to 11).

Step 2: Set all nodes in the S_C as the children node of parent node and push the children nodes into the Queue (line 18) and go the next state (line 23 to 25). Iterate the step 1 and 2 till the whole WSCT is returned.

Algorithm 1: CWSCT: Create the WSCT

```
Input: (CWS, WS Community)
Output: WSCT
Initialization: set root as the root of WSCT and push root
into the Queue, $S_{CBI}$, root's $S_P \leftarrow \emptyset$, i←state of CWS,
BEGIN
1: while Queue≠∅ do
2:   parent node←Pop(Queue)
3:   $S_P \leftarrow$GetPreviousOp(parent node)
4:   if $S_P = \emptyset$ then
5:     $S_C \leftarrow S_{CBI} \cap W_0.\ S_{OS}$ [i]
6:   else $S_C \leftarrow S_P \cap W_0.\ S_{OS}$ [i]
7:     if $S_C = \emptyset$ then
8:       $S_C \leftarrow S_{CBI} \cap W_0.\ S_{OS}$ [i]
9:       add the $S_P$ into previous operations set of each
             node in $S_C$
10:      if there are the same operations come from
           different WSs in the previous operations set
             of node in $S_C$ then
11:        remove the node from $S_C$
12:      end if
13:    end if
14:  end if
15:  if $S_C = \emptyset$ then
16:    remove the parent node
17:    break
18:  end if
19:  while $S_C \neq \emptyset$ do
20:    node←Pop($S_C$)
21:    set node as the children node of parent node
             and push the node into Queue
22:  end while
23:  if all nodes in the same state are considered then
24:    i←i-1
25:  end if
26: end while
27: return WSCT
END
```

In Fig. 2, we create the WSCT from back, firstly there are nodes $\{W_1.o_4\}$ and $\{W_2.o_4\}$ as the candidate nodes and their previous set is $\{W_1.o_2$ or $W_1.o_3\}$ and $\{W_2.o_1\}$ each other, at the same time, we set $\{W_1.o_4\}$ and $\{W_2.o_4\}$ as the children nodes, then go the next step, we get $W_1.o_4$ or $W_2.o_4$ as the parent node each other, if firstly we get $\{W_2.o_4\}$ as the parent node, the candidate nodes is $\{W_3.o_2$ or $W_3.o_3$, $W_4.o_3$ and $W_5.o_3\}$, their previous set is ∅, ∅ and $\{W_5.o_1\}$, since the previous set $\{W_1.o_4\}$ of $\{W_2.o_4\}$ can not be selected as candidate node, so the previous set $\{W_1.o_4\}$ of $\{W_2.o_4\}$ must be added into previous set of its children nodes, so the previous set of $\{W_3.o_2$ or $W_3.o_3\}$, $\{W_4.o_3\}$ and $\{W_5.o_3\}$ is $\{W_2.o_1\}$, $\{W_2.o_1\}$ and $\{W_5.o_1, W_2.o_1\}$ each other, here, because there are two o_1 coming from different WSs in the previous set of $\{W_5.o_3\}$, so the $\{W_5.o_3\}$ will be delete from the candidate node set. At last, iterate the process till returning the whole WSCT.

Figure 3 shows the WSCT based on algorithm 1 according to W_0 and the atomic WSs in Fig. 2. Dotted line frame is the wrong path that is removed by the algorithm 1. According to composite WS W_0, we can use some atomic WSs from the WS community to get some execution paths that meet the composite WS W_0, when

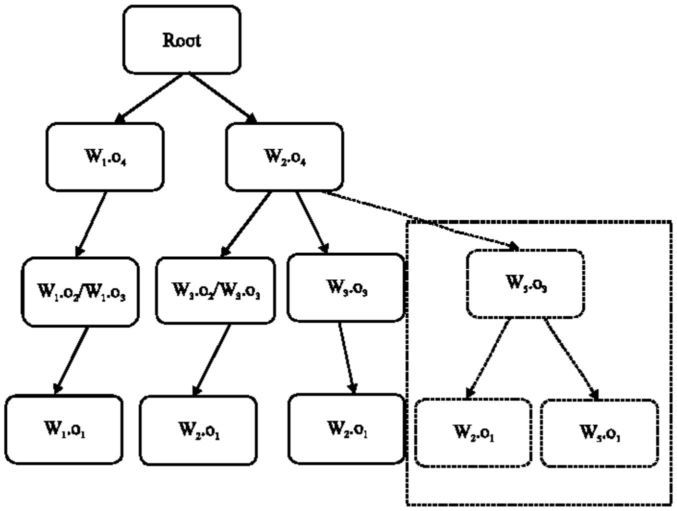

Fig. 3: The WSCT of Fig. 2 created by the algorithm 1

the number of atomic WSs becomes large, calculating all the reliability of all WSC execution paths may take too much time and thereby we give a heuristic algorithm to select the WSs for composite WS.

In Table 1 we can see that each path from root to leaf node can meet the requirement of CWS. So we must select a path to execute the CWS according to the QoS. Here, we consider a QoS property, reliability. The reliability of a service is the probability that a request is correctly responded within the maximum expected time frame (Zeng *et al.*, 2004). Moreover, Hwang *et al.* (2008) proposed a service metric called aggregated reliability, to measure the probability that a given state in a composite WS will lead to successful execution in an error-prone environment. Although proposed powerful method to compute the aggregated reliabilities, if the number of atomic WSs becomes huge, calculating the all aggregated reliability is impractical, because it may take too much time. So we propose a heuristic algorithm to select the WSC based on WSCT above mentioned and difference from Hwang *et al.* (2008), in this study, the reliability of a whole execution path of composite WS is a product of reliability of all operations.

Definition 3 (Reliability): Given a composite WS W_0 and a WS community $C = \{W_1, W_2, ..., W_n\}$, the Reliability of W_0 using C is $R(W_0) = \Pi R(W_i.o_j)$, $1 \le i, j \le n$, where:

- $R(W_0)$ is the Reliability of CWS W_0
- $R(W_i.o_j)$ is the Reliability of operation o_j in the service W_i

Using Heuristic Algorithm (HA) to Select the WSC Based on WSCT: Generally, the heuristic evaluation function will be used to evaluate and find the suitable path from the current node to the goal node. Heuristic evaluation function consist of two parts, one part is viewed as G (x), standing for known information, another one is H (x), standing for the unknown information. Our proposed solution to select the suitable composition is utilizing a Heuristic Evaluation Function (HEF) to evaluate the reliability of the path from current node to the goal node current node and then select the suitable path according to heuristic evaluation function.

Designing appropriate heuristic evaluation function is very important to the heuristic algorithm. In order to design better heuristic evaluation function, we must try our best to gain known information. So, we analyze the execution tree of the WSC selection. In this study, we define heuristic evaluation function as the function that evaluates the reliability of the path from the current node to the goal node. The evaluation of path from current node to the goal node is divided into four parts. The definition of HEF as follows.

Definition 4 (HEF): Let HEF be a function $F(x) = G_1(x) \times G_2(x) \times H_1(x) \times H_2(x)$, where:

- $G_1(x)$ is the current node information
- $G_2(x)$ stand for nodes information that must be selected into WSC according to current node
- $H_1(x)$ is the next step node information and it is also the children node information of current node
- $H_2(x)$ is these nodes information that can not be considered in $G_1(x)$, $G_2(x)$ and $H_1(x)$ but these nodes belong to the CWS

The value of $G_1(x)$, $G_2(x)$, $H_1(x)$ and $H_2(x)$ is the reliability of the part of path in the corresponding function. According to the WSCT we know that in atomic WSs, execution of some operations will induce the inevitable occurrence of a number of previous operations in the same atomic WS. For example, in Fig. 2, assumption that the operation $W_2.o_4$ be selected which will induce the inevitable occurrence of previous operation $W_2.o_1$ that is to say, $W_2.o_1$ must be selected into CWS W_0. Therefore, we consider that the information of selecting $W_2.o_1$ will be part of the heuristic evaluation function. Here the heuristic evaluation function includes two parts, one part is viewed as G (x), standing for known information and G (x) is divided into two parts, G_1 (x) and G_2 (x) (definition 3). Another part of the heuristic function is H (x) which also is divided into two parts, H_1 (x) and H_2 (x) (see definition 3). For example, in Fig. 3, we consider selecting the $W_1.o_4$ or $W_2.o_4$, if the $W_1.o_4$ is selected, the information of $W_1.o_4$ is the G_1 (x) and according to compare with W_0, $W_1.o_2$, $W_1.o_3$ and $W_1.o_1$ must be selected, so the information of $W_1.o_2$, $W_1.o_3$ and $W_1.o_1$ is the G_2 (x), according to WSCT, $W_1.o_2$ and $W_1.o_3$ also belong to the H_1 (x). Because there are the same operations o_2 and o_3 in G_2 (x) and H_1 (x), the operations are considered once. H_2 (x) does not have any operations here, so the reliability of H_2 (x) is 1. If the $W_2.o_4$ is selected, the information of $W_2.o_4$ is the G_1 (x) and according to comparison with W_0, $W_2.o_1$ must be selected, so the information of $W_2.o_1$ is the G_2(x), according to WSCT, $W_3.o_2$ and $W_3.o_3$ is the H_2 (x) and H_1 (x) is empty, so, the reliability of H_1 (x) is 1.

After designing the heuristic evaluation function, the process of algorithm can be designed naturally.

Here we give some signs introduction in the algorithm 2 and 3. WSCT is the root node of WSCT. W_{EP} is a stack that is used to store the selected nodes. P_R is the reliability of the execution path in W_{EP}. S_{CL} is a set that store the children node of a parent node and node is a node. F_{HEF} is the value of function HEF.

The idea of algorithm 2 is as follows: firstly, calculating the reliability of each branch of according to current nodes, then by comparing them selecting the maximal branch and remove the other branch, at last, finish all selection of the nodes and get the WSC execution path and the reliability of the path.

The description of algorithm 2 is as follows.

Initialization: Set W_{EP} as null and set P_R is 1:

- **Step 1:** If WSCT is not null, we get all children as the children set S_{CL} (line 1 to 2)
- **Step 2:** Calculate the HEF of all nodes in S_{CL} (Algorithm 3), then push the suitable node into the W_{EP} (line 3 to 5)
- **Step 3:** WSCT point to the root of sub tree of WSCT and the root node of sub tree is node (line 6). And repeatedly the Step1 and 2 till the WSCT has no child and return the selected path W_{EP} and the reliability P_R of the path

Algorithm 2: HS: Heuristic Select

```
Input: (WSCT, Composite WS, WS Community)
Output: ($W_{EP}$ and $P_R$)
Initialization: $W_{EP} = \emptyset$, $P_R = 1$
 BEGIN
1: while WSCT≠Ø do
2:   $S_{CL}$←get all children node of WSCT
3:   get the children node with maximal HEF by
        calling CF ($S_{CL}$)
4:   $P_R$←the reliability of node×$P_R$
5:   push the node into $W_{EP}$
6:   WSCT point to the root of sub tree of WSCT and the
        root node of sub tree is node
7: end while
8: return $W_{EP}$ and reliability $P_R$
END
```

The idea of algorithm 3 is as follows. Firstly, get the F_{HEF} of all nodes in S_{CL} according to definition of G_1(x), G_2(x), H_1(x) and H_2(x) and get the children node with maximal F_{HEF}, then return the node and the reliability P_R.

Algorithm 3: CF: Calculate F_{HEF}

```
Input: $S_{CL}$
Output: node and reliability $P_R$
 BEGIN
1: while $S_{CL}$≠Ø do
2:   node←Pop($S_{CL}$)
3:   $G_1$(x)←the reliability of node
4:   $G_2$(x)←the reliability of part of path in $G_2$(x)
5:   $H_1$(x)←the reliability of children node of node
6:   $H_2$(x)←the reliability of part of path in $H_2$(x)
7:   $F_{HEF}$←$G_1$ (x)×$G_2$ (x)×$H_1$(x)×$H_2$(x)
8: end while
9: get the children node with maximal $F_{HEF}$
10: $P_R$←the reliability of node
11: return the node and $P_R$
END
```

The steps are as follows:

- **Step 1:** Get a node as node from S_{CL} (line 2) and get the value of G_1(x), G_2(x), H_1(x) and H_2(x)
- **Step 2:** Get F_{HEF} of all nodes in S_{CL} according to G_1 (x), G_2 (x), H_1(x) and get the children node with maximal F_{HEF} in S_{CL}.
- **Step 3:** return node and P_R of the node

Consider the composite WS W_0 and the WS community C = {W_1, W_2, W_3, W_4, W_5} shown in Fig. 2. For simplicity, like article Hwang *et al.* (2008), assume that each operation in node has the same chance to be selected, the children nodes have the same chance to be selected and has the same reliability 0.8, except for operations $W_3.o_2$, $W_3.o_3$ and $W_4.o_3$, whose reliabilities are all 0.75.

In Fig. 3, if $W_1.o_4$ is selected into the composite WS W_0, by comparison between the W_0 and W_1, we know that $W_1.o_1$, $W_1.o_2$ and $W_1.o_3$ must be selected into the execution path of WSC, so, we can calculate the $G_2(x) = R(W_1.o_1)\times 1/2\times(R(W_1.o_2)+R(W_1.o_3)) = 0.64$ $W_1.o_4$ is the current operation, so, $G_1(x) = R(W_1.o_4) = 0.8$, here, because the next step selection operations also belong to the $G_2(x)$, so, $H_1(x) = 1$ and there are no operations between W_0 and W_1. We define $H_2(x) = 1$. So, $F_{HEF} = 0.8\times0.64\times1\times1 = 0.512$.

As discussion above, if $W_2.o_4$ is selected, we have $G_2(x) = R(W_2.o_1) = 0.8$, $G_1(x) = R(W_2.o_4) = 0.8$, we know that the next step selection operations are $W_3.o_2$, $W_3.o_3$ and $W_4.o_3$ according to WSCT as shown in Fig. 3. $W_3.o_2$ and $W_3.o_3$ have an equal chance to be selected, $H_2(x) = 1/2\times(1/2\times(R(W_3.o_2)+R(W_3.o_3)) + 1/2\times(R(W_4.o_3)) = 0.75$. and the different operations set is null, so $H_1(x) = 1$, then $F_{HEF} = 0.8\times0.8\times1\times0.75 = 0.48$.

So, we select $W_1.o_4$ as the next step, we can continue till all steps are finished. At last, we can gain a feasible execution path $\{W_1.o_4, W_1.o_3 \text{ or } W_1.o_2, W_1.o_1\}$.

Using backtracking algorithm (BA) to select the WSC: Although the heuristic algorithm can resolve the problem that select the suitable the execution path, the time that spend in the process of creating the WSCT still is too much. So we consider selecting a feasible execution path without creating the WSCT that is say, according to the merit of model, we can select a path to accomplish the requirement of users instead of firstly spending larger of time in creating WSCT and then selecting an optimal or approximately optimal path.

According to the analysis above, we still execute the WSC process from back. The process of select a feasible path by backtracking algorithm is: firstly, we get all possible candidate nodes and then according to the merits of WSs model we can attain each Selection Function (SF) of all candidate nodes, SF is used to evaluate the path including the current node. At last, we sort the candidate nodes according to the SF and select the best candidate node into WSC according to the value of the SF of candidates nodes, if the selected candidate node is failed in the future process, we will backtrack and select the second-best candidate node and iterate the process till we get a feasible execution.

In Fig. 2, for example, we consider selecting o_4. Firstly we get all possible candidate nodes, the set is $\{W_1o_4, W_2.o_4\}$ and then we calculate the SF of each node in the set. Suppose we select the $W_2.o_4$ as the part of WSC according to the value of SF and if $W_2.o_4$ is failed in the future process, we will backtrack that is say, we will select $W_1.o_4$ to replace the $W_2.o_4$ and remove the WSs including $W_2.o_4$ in WS Community then iterate the above process till we get a whole feasible path.

Now we introduce how to calculate the selection function *SF(x)*.

Definition 5 (SF): Let SF be a function $SF(x) = G_1(x)\times G_2(x)\times H(x)$, where:

- $G_1(x)$ is the current consideration node information
- $G_2(x)$ stand for nodes that must be selected into WSC according to current node information
- $H(x)$ is these operations information that can not be considered in $G_1(x)$, $G_2(x)$

Different from Definition 4, since we do not create the WSCT, we can not know the next step node information, then we can not divide the H (x) into two parts. When we consider selecting o_4, we can get all possible candidate operations set which is $\{W_1.o_4, W_2o_4\}$. Assumption that the operation $W_2.o_4$ be selected into the WSC which will lead to the inevitable occurrence of operation $W_2.o_1$ that is to say, $W_2.o_1$ must be selected into WSC. So, the information of $W_1.o_4$ belong to $G_1(x)$ and the information of $W_2.o_1$ belong to $G_2(x)$. Since we do not know the next step node information, we do not know where the o_2 or o_3 come from, so the next step nodes of $W_2.o_4$ is all possible nodes in the set $\{W_3.o_2 \text{ or } W_3.o_3, W_4.o_3, W_5.o_3\}$, we denote the average value of them as the information of H (x). At the same time, if the $W_1.o_4$ is selected, information of $W_1.o_4$ belong to $G_1(x)$ and according to the W_0 and W_1, the information of set $\{W_1.o_1, W_1.o_2, W_1.o_3\}$ belong to $G_2(x)$ and H (x) is empty, so the reliability is 1.

Here we give some signs introduction in algorithm 4 to 6: $Q_{OQ}[i]$ is an ordered queue that stores the backtracking candidate nodes in the ith state, SF is the value of function SF (x).

The general idea of algorithm 4 is as follows: we still select the suitable nodes of each state from back, by calculating SF (x) of each candidate node in the same state, we can sort order to them, then, we select the candidate node that has the maximal SF (x), if we need backtrack, we select second maximal one, repeatedly till there are no any candidate node in the same state, then we backtrack to the previous state, repeatedly steps mentioned above. At last, we can get a feasible WSC execution path.

The description of algorithm 4 is as follows.

Initialization: push the root into the W_{EP}, we put all the last state operation nodes of atomic Wss in the WS

Community into the S_{CBI}, set S_P of root as null. For example, at first, in Fig 2, S_{CBI} = {$W_1.o_4$, $W_2.o_4$, $W_3.o_2$ or $W_3.o_3$, $W_4.o_3$ and $W_5.o_3$}:

- **Step 1:** Get a node as node from W_{EP} (line 1) and get candidate nodes set S_{CS} of next state (line 4 to 11)
- **Step 2:** If need backtrack, see algorithm 6, or get the candidate node have maximal SF (x), (algorithm 5), then repeat the process till we get a feasible execution path W_{EP} and P_R of the W_{EP}

Algorithm 4: BTS: Backtracking select

```
Input: WS Community
Output: $W_{EP}$ and $P_R$
Initialization: stack $W_{EP}$, push root into $W_{EP}$, i is the
current state, Queue $Q_{OQ}$ [i]
 BEGIN
1: while Queue$\neq\emptyset$ do
2: parent node$\leftarrow$get a node from $W_{EP}$
3:  $S_P\leftarrow$GetPreviousOp(parent node)
4:  if $S_P = \emptyset$ then
5:   $S_C\leftarrow S_{CBI}\cap W_0$. $S_{OS}$ [i]
6:  else $S_C\leftarrow S_P\cap W_0$. $S_{OS}$ [i]
7:   if $S_C = \emptyset$ then
8:    $S_C\leftarrow S_{CBI}\cap W_0$. $S_{OS}$ [i]
9:    add the $S_P$ into previous operations set of each
          node in $S_C$
10:   if there are the same operations come from
      different WSs in the previous operations set
         of node in $S_C$ then
11:     remove the node from $S_C$
12:   end if
13:  end if
14: if $S_C = \emptyset$ then
15:    BT ($W_{EP}$)
16:    break
17: else $Q_{OQ}$ [i]$\leftarrow$SF($S_C$ )
18: node$\leftarrow$Pop $Q_{OQ}$ [i] ;
19:  push( $W_{EP}$, node)
END
```

The description of algorithm 5 is as follows:

- **Step 1:** Get all SF of candidates in S_C, (line 2 to) and push all SF into Q_{OQ} [i]
- **Step 2:** Sort order to all SF by descending and return the Q_{OQ} [i]

Algorithm 5: SF: Selection function

```
Input: $S_C$
Output: $Q_{OQ}$ [i]
 BEGIN
1: while $S_C\neq\emptyset$ do
2: node$\leftarrow$Pop($S_C$)
3: $G_1$(x)$\leftarrow$the reliability of node
4: $G_2$(x)$\leftarrow$the reliability of part of path in the $G_2$(x)
5: H(x)$\leftarrow$the reliability of part of path in the H(x)
6: SF = $G_1$ (x)$\times G_2$ (x)$\times$H(x)
7: push SF into $Q_{OQ}$ [i]
8: end while
9: sort the set $Q_{OQ}$ [i] by Descending
10: return $Q_{OQ}$ [i]
 END
```

The description of algorithm 6 is as follows.

The algorithm 6 is the backtracking process, when the first maximal SF (x) need to be backtracked, because the Q_{OQ} [i] is a sort order sequence, we can invocate the second-best SF(x) in the same state, if there is not any other node in the Q_{OQ} [i] of ith state, we must backtrack to previous state (line 5 to 6) and repeatedly till we can get a substitute node and push node into W_{EP} as the replacement of node:

- **Step 1:** If the Q_{OQ} [i] is not null, get a node as node from Q_{OQ} [i], (line 1 to 2)
- **Step 2:** If the Q_{OQ} [i] is null, we go back the last state (line 6), W_{EP} need to delete a node (we need to select the node again) and repeat Step 1

Algorithm 6: BT: Backtracking

```
Input: $W_{EP}$
Output: $W_{EP}$
 BEGIN
1: while $W_{EP}\neq\emptyset$ do
2:   if $Q_{OQ}$ [i]$\neq\emptyset$ then
3:      node$\leftarrow$Pop ($Q_{OQ}$ [i])
4       break
5:    else
6:      i$\leftarrow$i-1
7:     Pop($W_{EP}$)
8:    end if
9:  end while
10: push node into $W_{EP}$
11: return $W_{EP}$
    END
```

Similarly, we consider the composite WS W_0 and the WS community C = {W_1, W_2, W_3, W_4, W_5} shown in Fig. 2. As mentioned above, assume that each operation has an equal chance to be selected and has the same reliability 0.8, except for operations $W_3.o_2$, $W_3.o_3$ and $W_4.o_3$, whose reliabilities are all 0.75 and $W_5.o_1$ and W $_5$o $_3$ whose reliabilities are all 0.95. We denote the R ($W_1.o_2$) as the reliability of the operation $W_1.o_2$.

In Fig. 2, at first, we have two choices about o_4, they are $W_1.o_4$ and W .o$_2$. $_4$Consider the W .o , $_1$G$_4$ (x)$_1$ = R ($W_1.o_4$) = 0.8, selected into the composite WS W_0 and $W_1.o_1$, $W_1.o_2$ and $W_1.o_3$ must be selected into WSC, so G_2 (x) = R ($W_1.o_2$)×1/2×(R ($W_1.o_2$)+R ($W_1.o_3$)) = 0.64. By comparison between the W_0 and W_1, we know that there is not operation in H(x), so, H(x) = 1 and SF = G_1 (x)×G_2 (x)×H (x) = 0.512. On the other hand, $W_2.o_4$ is the current operation, so, G_1(x) = R ($W_2.o_4$) = 0.8 and G_2(x) = R ($W_1.o_2$) = 0.8, here, because the next step possible selection operations are {$W_3.o_2$, $W_3.o_3$, $W_4.o_3$, $W_5.o_3$} according to WSCT as shown in Fig. 3, so, H (x) = 1/3×(1/2×(R ($W_3.o_2$)+R ($W_3.o_3$))+1/3×(R ($W_4.o_3$)+1/3×(R ($W_5.o_3$)) = 0.82, then SF = G_1 (x)×G_2 (x)×H (x) = 0.523.

So, we select $W_2.o_4$ as the next step operation and then we continue to select the next step operation. The next operation set is $\{W_3.o_2$ or $W_3.o_3, W_4.o_3, W_5.o_3\}$. Similarly, the SF is {0.656, 0.656, 0.901}, we select $W_5.o_3$ but, in next step, because we have two optional operations $W_5.o_1$ and $W_2.o_1$, we need backtrack and we select $W_4.o_3$ or $W_3.o_2$ or $W_3.o_3$ to substitute $W_5.o_3$, then continue till all steps are finished. At last, we can gain a feasible execution path $\{W_2.o_4, W_4.o_3, W_2.o_1\}$.

EXPERIMENTS AND EVALUATION

In order to evaluate our proposed Heuristic Algorithm (HA) and Backtracking Algorithm (BA), two algorithms are compared with each other and with Exhaustion Method (EM). Exhaustion method traverses all the operations in the WSCT and then gets the execution path with best reliability. In our previous study, the heuristic algorithm have been compared with exhaustion method And then, we compared the three methods. The experiments are run on a PC configure with Intel Petium (R) IV 3.00 GHz CPU, 1G RAM.

Here we mainly evaluated the total reliability of the selected path and the selection time among exhaustion method, our proposed heuristic algorithm and backtracking algorithm.

In the first scenario, we firstly generated a composite WS with six states and twelve atomic WSs into the WS community. Then, we select a suitable CWS according to twelve atomic WSs and we carried out the experiment for ten groups in HA and EM and each group the operations reliability is generated by random from 0.8 to 1.0. Here, we do not consider the time of constructing WSCT and only compare the time of the selection of EM and HA. The experimental results are shown in Table 1. The x-axis stands for that we carry out the experiment for ten groups and the y axis stands for the WSC reliability of the total execution path and the data is shown in Table 1. Then on basis of the Table 1 we generated a composite WS with six states and seven WS communities with different number of atomic WSs from 10 to 22. We carried out the experiment according to different WS community for ten groups and got the average time of selecting an execution path. The experimental results are shown in Table 2 the x-axis stands for the number of atomic WSs and the y axis stand for the average time of the execution path is selected for ten groups by corresponding method, the data is shown in Table 2 and the unit of time is millisecond. Table 1 shows that the average reliability of the path selected by HA is about 95% reliability of the best execution path selected by EM and from the Table 2, when the number of atomic WSs is twelve, the average time of selection by HA is about one fourth of selection time by EM. In other word, we spend one fourth selection time by EM but we get about 95% precision. So, the execution path selected by our HA can meet the requirement of users and Table 1 shows that with the increasing of the atomic WSs, the selection time by HA is relatively stable and the selection time by EM increase obviously .

In the second scenario, we evaluate the three selection methods, we firstly still generate a composite WS with six states and twelve atomic WSs into the WS community and then, we applied three methods to select a CWS and considered the time of creating WSCT. Meanwhile, we carried out experiment for ten groups and each group the operations reliability is generated by random from 0.8 to 1.0. The experimental results Table 3.

Then, a composite WS with six states and seven WS communities with different number of atomic WSs from 10 to 22 are generated. Experiments are carried out to each WS community for ten groups and got the average time of selecting an execution path. The experimental results shown in Table 4.

Table 3 shows that the reliability of the path selected by EM, HA and BA for ten groups with different operations reliability. Results of experiments show that the reliability of execution path selected by BA is about 84% reliability of execution path selected by HA and about 81% reliability of execution path selected by EM. Although reliability of the execution path selected by our BA may be lower reliability than the best execution path selected by EM or the path selected by HA, the feasible path can be got in shorter time shown in Table 4.

Table 1:The WSC reliability of the execution path for ten groups experiment

Group	EM	HA
1	0.673	0.592
2	0.668	0.591
3	0.704	0.704
4	0.579	0.567
5	0.764	0.754
6	0.668	0.646
7	0.714	0.708
8	0.629	0.617
9	0.651	0.651
10	0.621	0.621
Average	0.667	0.645

Table 2: The time of selected by different number of atomic WSs

No. of atomic WS	EM (ms)	HA (ms)
10	48	16
12	60	16
14	87	17
16	113	16
18	173	18
20	220	24
22	286	28

EM: Exhaustion method, HA: Heuristic algorithm

Table 3: The WSC reliability of the execution path for ten groups experiment

Group	EM	HA	BA
1	0.673	0.592	0.417
2	0.668	0.591	0.642
3	0.704	0.704	0.584
4	0.579	0.567	0.401
5	0.764	0.754	0.681
6	0.668	0.646	0.519
7	0.714	0.708	0.428
8	0.629	0.617	0.629
9	0.651	0.651	0.573
10	0.621	0.621	0.532
average	0.667	0.645	0.541

BA: Backtracking algorithm

Table 4: The time of selected by different number of atomic WSs

No. of atomic WS	EM (ms)	HA (ms)	BA (ms)
10	98	56	21
12	138	78	32
14	192	98	45
16	224	124	67
18	326	185	48
20	443	270	36
22	565	312	143

In the community with twelve atomic WSs, Table 2 shows that the time of selecting path is 16 ms but Table 4 shows that the time of including WSCT is 78 ms, so the time of creating WSCT is about 80% of whole time, in fact, when we get about 95% reliability of EM, we spend half selection time by EM, however, we spend one fifth selection time of EM in BA, then we can get about 81% reliability of EM. From Table 4 we can also see that time of selection by BA is relatively stable and much shorter than EM and HA. So, BA can find a feasible path with less time.

CONCLUSIONS

WSC has become focus in the field of WSs. Different composition methods have different merits. This study uses FSM to model the permitted invocation sequences of Web service operations. We summarize our contributions to the above problem as follows:

- WSs comprise multiple operations and their invocation sequence is constrained, so we propose an algorithm to create the WSCT from back to front in order to decrease the space of WSCT slightly which can avoid exhaustion to all operations of WSs
- Generally, selecting an optimal path needs traverse the whole WSCT. For the sake of decreasing the time of selection, we propose a heuristic algorithm to select an approximately optimal execution path according to the WSCT. By the heuristic algorithm, we only need to traverse parts of the WSCT, then the time of selection become shorter
- In order to further decrease the time of selecting the execution path, we propose a backtracking algorithm to select a feasible execution path without constructing the WSCT. According to the merit of model, each time we select a feasible execution operation to meet the requirement of WSC, if need, backtracking the previous operation selection, then till we accomplishing the whole WSC. In other words, we select a feasible execution path instead of firstly spending a large amount of time in creating WSCT and then selecting an optimal or approximately optimal path

This study only consider an attribute of QoS, reliability, in the future works, we will use more attributes of QoS to evaluate the WSC. Besides, the model of describing the WSs in this study only consider the sequence and we will consider more complicated structure to be better suitable for application more widely.

ACKNOWLEDGMENT

This study is partially supported by National Natural Science Funds (No.60803032, 90718012), Program for New Century Excellent Talents in University (NCET-10-0598), Program for Shanghai Science and Technology Commission (No. 09JC1414200), “Shu Guang” project supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation and Shanghai Rising-Star Program.

REFERENCES

Berardi, D., G.D. Calvanese, G. De-Giacomo, M. Lenzerini and M. Mecella, 2003. Automatic composition of e-services that export their behavior. Proceedings of 1st International Conference on Service-Oriented Computing, Dec. 15-18, Springer Verlag, pp: 43-58.

Berardi, D., F. De-Rosa, L. De-Santis and M. Mecella, 2004. Finite state automata as conceptual model for e-services. J. Integrated Design Process Sci., 8: 105-121.

Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, 2004. Web services architecture, W3C working group. http://www.w3.org/TR/ws-arch.

Chen, Z., J. Ma, L. Song and L. Lian, 2006. An efficient approach to web services discovery and composition when large scale services are available. Proceeding of the IEEE Asia-Pacific Conference on Service Computing, Dec. 12-15, Guangzhou, Guangdong, pp: 34-41.

Ding, Z.J., J.L. Wang and C.J. Jiang, 2005. Semantic web service composition based on OWL-S. Proceedings of 1st International Conference on Semantic, Knowledge and Grid, Nov. 27-29, Beijing pp: 98-98.

Ding, Z.J., J.L. Wang and H. Song, 2007. AI planning for web service automatic composition using petri nets. Proceedings of 11th International Conference on Computer Supported Cooperative Work in Design, April 26-28, Melbourne, Australia, pp: 519-524.

Ding, Z.J., J.L. Wang and C.J. Jiang, 2008. An approach for synthesis petri nets for modeling and verifying composite web service. J. Inform. Sci. Eng., 24: 1309-1328.

Gerede, C.E., R. Hull, O.H. Ibarra and J. Su, 2004. Automated composition of e-services: Lookaheads. Proceedings of 2nd International Conference on Service Oriented Computing, Nov. 15-19, New York, pp: 252-262.

Haddad, J.E., M. Manouvrier and M. Rukoz, 2010. TqoS: Transactional and QoS-aware selection algorithm for automatic Web service composition. IEEE Trans. Services Comput., 3: 73-85.

Hu, Y. and H. Wang, 2004. Constraints in web services composition. Proceedings of 4th International Conference on Wireless Communications, Networking and Mobile Computing, Oct. 12-14, Dalian, pp: 1-4.

Hwang, S.Y., E.P. Lim, C.H. Lee and C.H. Chen, 2008. Dynamic web service selection for reliable Web service composition. IEEE Trans. Services Comput., 1: 104-116.

Jaeger, M.C., G. Muhl and S. Golze, 2005. QoS-aware composition of web services: A look at selection algorithms. Proceedings of the IEEE International Conference on Web Services, July 11-15, IEEE Computer Society, Washington, DC., pp: 807-808.

Shen, H.J., Z.J. Ding and H.Z. Chen, 2010. Reliable web service selection using a heuristic algorithm. Proceedings of Conference on Grid and Cloud Computing, Nov. 1-5, Nanjing, pp: 290-295.

Yu, T., Y. Zhang and K.J. Lin, 2007. Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans. Web, 1: 1-23.

Zeng, L., B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, 2004. QoS-aware middlware for web service composition. IEEE Trans. Software Eng., 30: 311-327.