UCLAO* and BHUC: two novel planning algorithms for uncertain Web service composition

Sen Niu[a,†], Guobing Zou[a,†,*], Yanglan Gan[b,*], Zhimin Zhou[a], Bofeng Zhang[a]
[a]School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
[b]School of Computer Science and Technology, Donghua University, Shanghai 201620, China
{sniu,gbzou}@shu.edu.cn, ylgan@dhu.edu.cn, {zmzhou,bfzhang}@shu.edu.cn

***Abstract*—The inherent uncertainty of Web service is the most important characteristic due to its deployment and invocation within a real and highly dynamic Internet environment. Web service composition with uncertainty (U-WSC) has become an important research issue in service computing. Although some research has been done on U-WSC via non-deterministic planning in Artificial Intelligence, they cannot handle the situation that uncertain Web services with the same functionality exist in a service repository and could not get all of possible solution plans that constitute an uncertain composition solution for a given request. To solve above research challenges, this paper models a U-WSC problem into a U-WSC planning problem. Accordingly, two novel uncertain planning algorithms using heuristic search called UCLAO* and BHUC, are presented to solve the U-WSC planning problem with state space reduction, which leads to high efficiency of finding a service composition solution. We have conducted empirical experiments based on a running example in e-commerce application as well as our large-scale simulated datasets. The experimental results demonstrate that our proposed algorithms outperform the state-of-the-art non-deterministic planning algorithms in terms of effectiveness, efficiency and scalability.**

***Keywords*—Web service; Web service composition; Uncertainty; Non-deterministic planning**

I. INTRODUCTION

Web service composition (WSC) is the task of combining a set of single Web services together to create a complex, value-added and cross-organizational business process [1]. It is applicable to those scenarios where individual Web service cannot satisfy the functionality requirement of a composition request. Many works have been done on WSC [2], [3], [4], [5], [6], where a WSC problem is modeled as a workflow business model or a classic AI planning problem that can be solved by an off-the-shelf automated planner to find a plan solution. However, most of these approaches suppose that Web services are stateless with certain execution effects. Thus, they seldom considered the feature of inherent uncertainty of services.

Since Web services are published, deployed and invoked in dynamic Web environment, there are multiple internal features with uncertainty, mainly including non-deterministic effects on functionality properties and inconsistent QoS values on non-functionality criteria. Therefore, uncertain Web service composition (U-WSC), composing existing Web services with the consideration of their uncertain features, has received many attentions and become a challenging research issue to be solved in service-oriented business applications.

Some efforts on uncertain Web service composition have been made in recent years. Based on integrity constrains, a WSC problem taking into account uncertainty with possible initial states is converted into a conformant planning problem [7]. And then, the conformant planning problem is solved by Conformant-FF planner using forward heuristic algorithm. This approach only considered uncertainty about initial states, instead of non-deterministic stateful execution effects in Web services. Moreover, WSC problem is also formulated as a partially observable non-deterministic planning problem solved by Model Based Planner (MBP) planner [8], where Web services are transformed into state transition systems (STSs). Although it realizes the interactions between Web services by belief states, the strong assumption is that MBP thoroughly reply on predefined Web services. This incurs the failure of finding a solution within a service repository where multiple uncertain services share the same functionality. Also, when there is more than one uncertain service with the same functionality, existing non-deterministic planning algorithms [9], [10], [11], [12] cannot find all possible paths for a given U-WSC problem. Thus, how to design novel algorithms to effectively and efficiently solve an uncertain Web service composition problem has become a research issue.

To address the above challenge, we proposed a framework modeling a U-WSC problem as a fully observable non-deterministic planning (FOND) problem that is expressively generated by a U-WSC planning problem. To solve the U-WSC planning problem, we proposed two novel kinds of U-WSC planning algorithms using heuristic graph search, called Uncertain Composition LAO* (UCLAO*) algorithm and Breadth Heuristic Uncertain Composition (BHUC) algorithm. Based on LAO* algorithm [9], the UCLAO* algorithm improved the node structure and the process of expanding the successor nodes, which effectively reduces the state search space and finds all possible solution paths with uncertainty. In terms of scalability, the BHUC is an efficient U-WSC algorithm using forward breadth search strategy. According to the value of heuristic function, the algorithm selects the applicable action with the maximum value. Compared to UCLAO* algorithm, it significantly improves the efficiency of finding a solution to an uncertain composition request, since it directly search all possible paths without redundancy.

[†]These authors contribute equally to this study and share first authorship.
[*]Corresponding author.

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.75

To validate the feasibility of our proposed algorithms,an empirical experiment has been conducted on a case study in e-commerce real application. The experimental results demonstrated that our approach not only can effectively handle the WSC problem with uncertainty where multiple uncertain Web services with the same functionality exist in a service repository, but also can efficiently find all the possible solutions for a given U-WSC problem.

The rest of this paper is organized as follows. In Section 2, we describe a running example on a real-world e-commerce application. Section 3 presents the problem formulation. In Section 4, we propose two novel algorithms for uncertain Web service composition using non-deterministic planning. Experimental results on the running example are shown in Section 5. Section 6 reviews related work on Web service composition. Finally, Section 7 concludes the paper.

II. Motivating Example

A running example from e-commerce application will be used throughout the paper. It consists of six Web services, including Retailer, Manufacturers M_1, M_2, and M_3, Assemble and Ship. Each service is responsible for a specific task with a collection of functionalities by its operations. Specifically, the **Retailer** sends a product request. The **Manufacturers** M_1,M_2 and M_3 have the same functionality, receiving a purchase request and checking its availability for the given request of the product purchase. The **Assemble** makes the assembling service based on the available status of the product and the **Ship** provides the shipping service of the product order.

The goal is to construct an integrated business process (i.e., a composed service) for product purchasing, assembling and delivering by combining a set of uncertain Web services. These services collaborate with each other to achieve a situation where the **Ship** can successfully provide the service with a requested product delivery for the **Retailer**. The abstract process of product ordering among six services is shown in Fig.1.

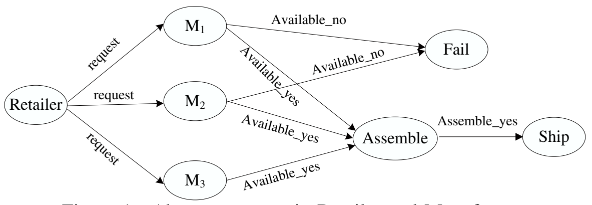


Figure 1. Abstract process in Retailer and Manufacturer.

In Fig.1, the **Retailer** sends a given product request (*product name, numbers*) to these manufacturers. The **Manufacturers** M_1, M_2 and M_3 can provide the given product requested by the **Retailer**. Assume that M_1 and M_2 may not be satisfied for the request, while M_3 must be available. After receiving the given product request, the manufacturers check their availability (*CheckAvail*) for the request. If the checking status is available (*Available_yes*), the **Assemble** provides the assembling order service, and finally the **Ship** service delivers the given product to the **Retailer** (*product name, numbers, price, date*).

III. Problem Formulation

We first formulate uncertain Web service composition (U-WSC) problem by a set of formal definitions, and then present what a composition solution is to a U-WSC problem. Here, we mainly focus on the inherent functional uncertainty of Web service, which is defined as below.

Definition 1 (Uncertain Web Service)**.** An uncertain Web service ws consists of a finite set of operations, denoted as $ws = \{op_1, op_2, \cdots\}$ and $\exists op_i \in ws$ which is an uncertain operation. When an uncertain operation executes, it returns an output set with multiple possible execution effects.

In the running example, there are six Web services, including the Retailer $R = \{sendrequest\}$, the manufacturer $M_1 = \{receiverequest, checkavail\_M_1\}$, the manufacturer $M_2 = \{receiverequest, checkavail\_M_2\}$, the manufacturer $M_3 = \{receiverequest, checkavail\_M_3\}$, Assemble $A = \{packing\}$, Ship $S = \{deliver\}$, where the operation in the manufacturer $checkavail\_M_i$ $(i = 1, 2)$ is an uncertain operation with multiple possible execution effects, i.e., $\{available\_yes, available\_no\}$. Thus, the M_1 and M_2 are uncertain Web services, while other services Retailer R, manufacturer M_3, Assemble A and Ship S are certain Web services.

Definition 2 (Uncertain Web Service Repository)**.** An uncertain Web service repository, denoted as $W = \{ws_1, ws_2, \cdots\}$, is a set of available services, where $\exists ws_i \in W(i = 1, 2, \cdots)$, it is an uncertain Web service.

An uncertain Web service repository consists of all the available services published by service providers on the Internet, including uncertain Web services as well as those certain ones. In the running example, it has six Web services, $W = \{Retailer, M_1, M_2, M_3, Assemble, Ship\}$.

Definition 3 (Uncertain Composition Request)**.** A user's functionality composition request, R, is a two-tuple $R = \{R_{in}, R_{out}\}$, where $R_{in} = \{r^1_{in}, r^2_{in}, \cdots\}$ is an interface parameter set provided as request inputs, and $R_{out} = \{r^1_{out}, r^2_{out}, \cdots\}$ is an uncertain goal specification provided as desired results.

Note that R_{out} includes all possible execution output states, e.g. success and failure execution states. In the running example, we assume that a user sets an uncertain request $R = \{\{request\}, \{\{ship\_yes\}, \{fail\}\}\}$, where $R_{in} = \{request\}$ is designated as initial state and $R_{out} = \{\{ship\_yes\}, \{fail\}\}$ points out all possible goal execution output results.

Definition 4 (U-WSC Problem)**.** An uncertain Web service composition problem, denoted as U-WSC, is defined as a three-tuple $< W, R_{in}, R_{out} >$, where

(1) $W = \{ws_1, ws_2, \cdots\}$ is an uncertain Web service repository;

(2) $R_{in} = \{r^1_{in}, r^2_{in}, \cdots\}$ is an input parameter set as the initial state and presents an input request;

(3) $R_{out} = \{r^1_{out}, r^2_{out}, \cdots\}$ is an uncertain output parameter set including all possible goal specification states and presents user's expected goal.

Following the running example, we can define a uncertain Web service composition problem as $U\text{-}WSC = \{W, R_{in}, R_{out}\}$, where uncertain web service repository $W = \{Retailer, M_1, M_2, M_3, Assemble, Ship\}$, the input parameter set $R_{in} = \{request, productname, numbers\}$ and the user's expected output parameter goal set $R_{out} = \{ship\_yes, productname, numbers, price, date\}$.

A solution to a U-WSC problem involves all possible execution paths which are invoked from the input initial state to one of the desired goal execution output states. It is defined as below.

Definition 5 (U-WSC Solution)**.** Given a U-WSC problem $< W, R_{in}, R_{out} >$, an uncertain composition solution consists of a set of execution paths $\{\prod_1, \prod_2, \cdots, \prod_n\}$. For $\forall \prod_i = \{op_1, op_2, \cdots, op_k\}$(1≤i≤n), it is a sequence of uncertain Web service operations that can be executed from R_{in} to one of the desired output state $r \subseteq R_{out}$.

Note that a composition solution to a U-WSC problem is an ordered sequence of services choosen from an uncertain Web service repository. With the combination of all the operation sequences $\prod_1, \prod_2, \cdots, \prod_n$ via these services, the solution considers all the possible paths that start from initial state R_{in} and leads to all the desired output goal states involved in R_{out}. Thus, multiple execution paths may be composed together as an uncertain composition solution to a U-WSC problem. For instance, when $R_{in} = \{request\}$ and $R_{out} = \{ship\_yes\}$, there are several possible execution paths in the running example.

(1) $\prod_1$=$\{sendrequest, receiverequest, checkavail\_M_1, packing, deliver\}$;

(2) $\prod_2$=$\{sendrequest, receiverequest, checkavail\_M_3, packing, deliver\}$;

(3) $\prod_3$=$\{sendrequest, receiverequest, checkavail\_M_1, checkavail\_M_3, packing, deliver\}$;

With the consideration of execution paths above, the manufacturer M_1 is available in $\prod_1$, while it is unavailable in $\prod_3$, because M_1 is an uncertain service and the execution output state of its uncertain operation $checkavail\_M_1$ may be $available\_yes$ or $available\_no$.

Given an uncertain service composition problem $< W, R_{in}, R_{out} >$, our objective is to find all the possible execution paths and compose them together as a solution to the U-WSC problem. When an uncertain Web service is invoked and the output of its operation cannot lead to a path to satisfy one of the desired goal state, our designed algorithms should select another uncertain or certain Web service with the same functionality to replace the failed uncertain Web service, rather than like those traditional Web service composition approaches that hold a strong assumption that they only reply on predefined Web services without functional repetition.

IV. Approach

In this section, we first give the formulations of U-WSC planning problem, and then propose two planning algorithms for solving the U-WSC planning problem.

A. U-WSC planning formulation

To apply non-deterministic planning algorithm to solve a U-WSC problem, we translate it into the U-WSC planning problem using the transition strategies. Since we mainly focus on uncertain planning algorithms, the detailed transition process is omitted here.

Definition 6 (U-WSC Action)**.** It is formalized as $a = (name(a), pre(a), eff(a))$, where $name(a)$ is action name, $pre(a)$ is a set of propositions as action preconditions, and *eff(a)* is a set of uncertain propositions as action effects.

A U-WSC action corresponds to an uncertain operation of Web service in our U-WSC problem and its preconditions and effects are translated from the input and output parameters of the corresponding operation. As a result, a U-WSC action has three kinds of execution effects, including certain effects, uncertain effects, and conditional effects.

Definition 7 (Effects of Uncertainty)**.** Given a U-WSC action a, its uncertainty of effects is expressed as *effs* $= \{C\} or \{U\} or \{L\}$, where C is a set of certain effects in a, U is a set of uncertain effects in a, and L is a set of conditional effects in a.

Following the running example, the operation $checkavail\_M_1$ in uncertain Web service manufacturer M_1 is a U-WSC action. It has multiple possible execution output effects which is a set of uncertain effects. Thus, its effects of uncertainty can be denoted as *effs*$(checkavail\_M_1) = \{U\} = \{available\_yes, available\_no\}$. The operation $packing$ in Web service Assemble is a certain U-WSC action and has an output effect. Its effect is a set of certain effects and can be expressed as *effs*$(packing) = \{C\} = \{assemble\_yes\}$.

Definition 8 (U-WSC Planning Problem)**.** A U-WSC planning problem is defined as a five-tuple (S, A, γ, I, g), where S is a finite set of states, A is a finite set of U-WSC actions, $\gamma : S \times A \rightarrow 2^S$ is the non-deterministic transition function between S and A, I and g are initial state and desired goal specifications, respectively.

A U-WSC planning problem can be seamlessly expressed as a non-deterministic planning problem, which consists of a planning domain D and a planning problem P. For a U-WSC planning problem, the planning domain D includes all the states, U-WSC actions and the preconditions and effects of each U-WSC action. The planning problem P consists of an initial state I and the desired goal specifications g.

B. UCLAO* algorithm

LAO* algorithm is a heuristic state-space search algorithm for and-or graph, which searches for a cyclic solution to a non-deterministic planning problem. However, a solution to an uncertain Web service composition problem is inappropriate with cyclic executions, since this kind of execution cannot guarantee the termination of Web services with uncertainty. With the extension of existing LAO* algorithm, we proposed the UCLAO* algorithm for uncertain Web service composition, which avoids recursive action executions and consider service replacement when an uncertain service fails to perform a desired functionality.

Algorithm 1: UCLAO* planning algorithm

```
Input: a U-WSC planning domain $D$ and a planning
       problem $P =< I, g >$;
Output: an uncertain composition solution $\pi$;
1  $G' \leftarrow < I, \varnothing >$;
2  if $I \subseteq g$ then
3      Tag the initial node $I$ as solved;
4  while $I$ is not tagged as solved do
5      $N \leftarrow$Unexpandednongoal ($G'$);
6      if $N \neq \varnothing$ then
7          foreach $n \in N$ do
8              $S \leftarrow Expand(n)$; //invoke algorithm 2
9              foreach $s \in S$ do
10                 if $s \subseteq g$ then
11                     Tag the node $s$ as solved;
12     $G' \leftarrow S$;
13     $Z = Backwardreach(N)$;
14     Tag the nodes in the set $G'$;
15     Update node costs by value iteration in the set $Z$;
16 if $I$ is tagged as solved then
17     $\pi \leftarrow TraverseSolution(G')$;
18     return $\pi$;
19 else
20     return failure;
```

Definition 9 (Implicit graph)**.** An implicit graph G' consists of a set of vertices N and a set of edges E, $G' =< N, E >$, where $N = \{n_1, n_2, \cdots\}$, n_i is a vertice and represents a set of states and $E = \{e_1, e_2, \cdots\}$, e_i is a edge and represents the invoked relationships between vertices.

The implicit graph is an and-or graph, which has k-connectors that connect a node to a set of k successor nodes. The edges represent actions in U-WSC planning domain, including certain actions and uncertain actions. The and-or graph represents the relationships among nodes by actions. And-nodes correspond to the uncertain actions which can generate multiple possible output effects, while or-nodes correspond to alternative actions that can be applied to the current state.

The UCLAO* algorithm is described in Algorithm 1. It takes a U-WSC planning domain and a planning problem as inputs. The planning state of the search is represented within an and-or search implicit graph G', which represents a search space of states. The node in the implicit graph is a UCLAO* node, including two state sets. The domain represents all the states and their invocation relationships. In addition, each node generated in the search process is associated with a "label". If a node is labeled as solved, an execution plan that leads from the node to the goal can be found. The algorithm outputs all the possible execution plans as a composition solution which starts from the initial node and leads to the desired goal nodes.

The algorithm first initializes the implicit search graph with the initial node at line 1 and check if the empty plan is a solution to the problem at lines 2-3. Then, the main loop is entered at lines 4-15, where the graph is expanded until the initial node is labeled as the status solved. The body of the main loop proceeds by expanding the graph and updating state and labeling the actions. With the first step in the loop, a node which is non-terminal node is selected and expanded from the implicit graph and added to a node set at line 5. At lines 6-11, a node is expanded and labeled considering every possible action, as described in Algorithm 2. Any new successor nodes are added to the graph at line 12. With the second step in the loop, at lines 13-15, the cost of each node in the explicit graph is updated by value iteration with the selection strategy of LAO* and the nodes in graph are tagged. After the loop terminates, the solution plans is constructed by traversing the whole implicit graph from initial node to goal node at lines 16-20.

Algorithm 2: UCLAO* node expansion

```
Input: a non-goal node $n$ and a finite set of possible
       actions $A$;
Output: a set of expanded nodes $N$;
1  $A' \leftarrow \varnothing$;
2  $S \leftarrow \varnothing$;
3  foreach $a \in A$ do
4      if $n \subseteq pre(a)$ then
5          $A' \leftarrow A' \bigcup a$;
6  foreach $a \in A'$ do
7      $s \leftarrow execute(n, a)$;
8      if $s \notin n$ then
9          $S \leftarrow S \bigcup s$;
10 foreach $s \in S$ do
11     create a new node n;
12     $n \leftarrow s \bigcup n$;
13     $N \leftarrow N \bigcup n$;
14 return $N$;
```

In Algorithm 2, it takes a non-terminal node in implicit graph as input and outputs a set of its successor nodes. The algorithm first initializes an action set and a state set at lines 1-2. The all possible applicable actions are selected from the planning domain for the specified non-terminal node at lines 3-5. After the execution of every action in A', it generates all the successor states and adds each state to S, which did not appear in ancestor states, to the new states set. The new successor nodes set is generated based on the new states set at lines 10-14.

Following the motivating example, the input *request, not ship_yes* is as the initial node *I*. The algorithm expands the initial node and executes the action *select_manufacturer*. Then, add new states check_avail to the state space and check whether the state belong to the goal state set *ship_yes*. If the result is yes, the algorithm will tag the node as solved and backward to the initial node. Otherwise, the algorithm expands the node unexpanded in implicit graph by iteration until the initial node is tagged as solved.

A U-WSC planning problem can be effectively solved by UCLAO* algorithm, which is based on LAO* state space search via heuristic functions. It generates an uncertain composition solution to the problem with all the possible execution plans, such that our algorithm can handle the service composition problem with uncertainty by the techniques of offline service replacement.

C. BHUC Algorithm

Although UCLAO* can find a composition solution with all possible execution plans, its time computation complexity is high since there are redundant nodes expanded in the implicit graph. To improve the efficiency of finding a composition solution to an uncertain composition problem, a novel algorithm with the idea of breadth heuristic search strategy, called BHUC, is designed for solving a U-WSC planning problem.

Definition 10 (BHUC node)**.** A BHUC node n is composed of a set of states and a real number d, where d is the depth of node n in implicit graph.

The set of states of a node is inherited from its ancestor. The depth of a node is the depth of the node in implicit search graph.

Definition 11 (BHUC queue)**.** A queue Que is used to hold the BHUC nodes set $N = \{n_1, n_2, \cdots\}$, where $n_i \in N$ is a BHUC node and a non-terminal node in implicit graph.

The queue is a set of BHUC nodes. The nodes have not been expanded and they are not the goal nodes. Every node in the queue would be expanded in the process of state space search.

Algorithm 3: BHUC planning algorithm

```
Input: a U-WSC planning problem $< D, I, g >$;
Output: an uncertain composition solution $\pi$;
1  $G' \leftarrow < I, \varnothing >$;
2  $Que \leftarrow I$;
3  while $Que \neq \varnothing$ do
4      $n \leftarrow Removefirst(Que)$;
5      if $(n \not\subset g) \wedge (\neg label(n, expanded))$ then
6          $N \leftarrow$Expand($n$); // invoke algorithm 4
7          if $N \neq \varnothing$ then
8              Add $N$ to the queue $Que$;
9              Expand the implicit graph with $N$;
10     label($n$,expanded);
11 if $Que == \varnothing$ then
12     $\pi \leftarrow TraverseSolution(G')$;
13 return $\pi$;
```

Given a U-WSC planning problem $< D, I, g >$, the BHUC algorithm can find more efficiently generate a composition solution with all possible execution plans. The main process is shown in Algorithm 3. The algorithm has two main steps. (1) Build the implicit search graph, including node expansion and labeling at lines 1-10; (2) Traverse the implicit search graph and generate the uncertain composition solution at lines 11-13.

The algorithm starts an implicit search graph with the initial node which is added to the queue at lines 1-2. Then, the main loop builds the implicit search graph where the node is expanded in queue iteratively until the queue is empty at lines 3-10. The body of the main loop proceeds by expanding the successor nodes and pushing them in the queue. The first node is selected and removed from the queue at line 4, and then we check if the node is a goal node and expand the node at lines 5-6. The node expansion process is shown in Algorithm 4. At lines 7-9, the successor nodes are added into the queue and implicit search graph. The ancestor node is labeled as the status of "expanded" at line 10. After the completion of implicit search graph at lines 11-13, an composition solution with multiple possible execution plans can be found by traversing the whole implicit graph from initial state node to goal state node.

Algorithm 4: BHUC node expansion

```
Input: a non-goal node $n$ and a finite set of possible
       actions $A$;
Output: a set of expanded nodes $N$;
1  $A' \leftarrow \varnothing$;
2  foreach $a \in A$ do
3      if applicable($n, a$) then
4          $h(a) \leftarrow d$;
5          $A' \leftarrow a \bigcup A'$;
6  foreach $a \in A'$ do
7      Select the action $a$ with maximum depth value in
       action set $A'$;
8      $N \leftarrow Execute(a)$;
9  return $N$;
```

In Algorithm 4, an action set A' is first initialized at line 1. Then, all possible applicable actions from action set A in planning domain are selected and the function values of these actions can be drawn from the depth of each node at lines 2-5. At line 7, the action with the maximum depth value is chosen. Finally, the successor nodes set can be generated by the execution of the actions at line 8.

Following the motivating example, the input *request* is as the initial node *I*. The algorithm expands the initial node and executes the action *select_manufacturer* as same as UCLAO* algorithm. Then, add the new states in the state space until the BHUC queue is empty. Based on the current state space, the algorithm will choose the action which has maximum depth value,eg. the depth value of the action *check_avail* is more than *select_manufacturer*'s until the BHUC node is not expanded.

As a result, compared to the UCLAO* algorithm, the BHUC algorithm can more efficiently solve a U-WSC planning problem when finding an uncertain composition solution, because it does not expand any redundant nodes during the process of building implicit search graph. Different to dynamic programming, the BHUC algorithm is based on forward breadth search algorithm and we choose the depth value of the expanded action as the heuristic function.

V. Experimental Evaluation

A. Experimental setup

In order to validate the effectiveness of our proposed U-WSC approaches and compare the efficiency with state-of-the-art non-deterministic planning algorithms, we conducted some empirical experiments on a PC with Intel Dual Core 2.8 GHZ processor and 3G RAM in Windows 7.

There are four different algorithms that are highly related to uncertain planning techniques, including AO*, LAO*, UCLAO* and BHUC algorithms, have been applied for solving an uncertain service composition problem. Especially, the AO* and LAO* algorithms are based on Zero-heuristic search in myND planner [13], while UCLAO* and BHUC are the two our proposed algorithms with heuristic state space search.

We evaluate all the uncertain planning algorithms from three performance criteria. (1) The feasibility of finding a solution. Given a set of uncertain services in a Web service repository and an uncertain composition request, whether these algorithms can find a composition solution with all the possible execution plans. (2) Response time. If an algorithm can generate a composition solution, we calculate the cost of its time consumption. (3) Scalability. Along with the increasing number of Web services, whether they can still effectively and efficiently solve the problem of uncertain Web service composition.

B. Finding an uncertain composition solution

We solve the U-WSC problem on the running example using the four algorithms, including AO*, LAO*, UCLAO* and BHUC, respectively. We analyze them from three aspects: finding a solution, generating all the possible execution paths, and satisfying a composition request. The experimental results are shown in the following Table I.

Table I
The experimental results of finding an uncertain composition solution

Basic activity	*AO**	*LAO**	*UCLAO**	*BHUC*
Finding a solution	N	Y	Y	Y
Generating execution paths	N	Y	Y	Y
Satisfying composition request	N	N	Y	Y
Response time	–	9ms	11ms	3ms

From the experimental results in Table 1, we can find that AO* algorithm cannot find a solution for the running example. Although LAO* can find a composition solution to the problem, it still cannot satisfy the uncertain composition request, since there are cyclic actions when the execution output effects are uncertain. Our proposed two uncertain planning algorithms, UCLAO* and BHUC, can both find all the possible execution paths without any cyclic actions that are composed together as a composition solution.

From the results in table 1, the response time of LAO* algorithm, UCLAO* algorithm and BHUC algorithm consumes 9ms, 11ms and 3ms, respectively. On one hand, we can conclude that our proposed uncertain planning algorithm BHUC performs best, because it does not produce redundant planning states during the process of node expansion. On the other hand, although the response time of LAO* algorithm is a little bit shorter than that of our proposed UCLAO* algorithm, its composition solution cannot satisfy the U-WSC composition request, since there are cyclic execution paths that violate the uncertainty of Web services. As a result, our proposed algorithms can effectively and efficiently find an uncertain composition solution with all multiple execution paths, such that service requesters can choose an appropriate algorithm that can satisfy their preferences between effectiveness and efficiency.

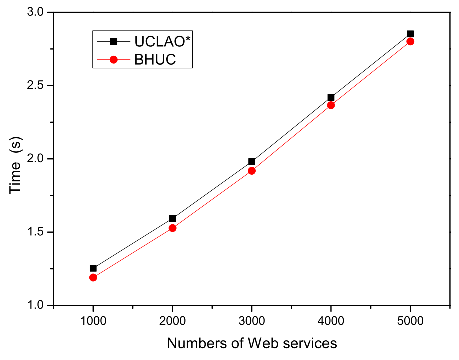
C. Response time and scalability

As an important evaluation criterion, the response time determines whether an uncertain composition solution can be rapidly returned to the users within a short period of time. To verify the scalability of the two proposed algorithms for uncertain Web service composition problem, we dynamically adjust the number of Web services in the running example service repository and check the search time of finding the composition solutions. We test our algorithms on different groups of Web service repositories which contain 60,000 services. The service repositories can be divided into four groups, including different numbers of uncertain Web service. The numbers of services in every group are from 1,000 to 5,000. Along with the increasing number of Web services, the empirical results of the search time are illustrated in Fig.2.

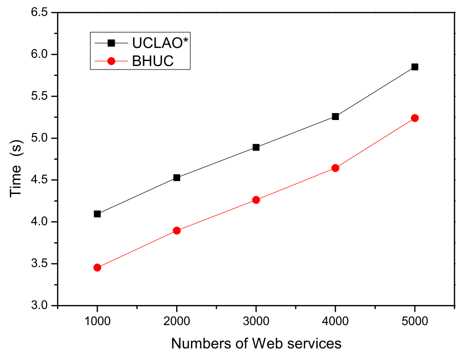
For the results of response time, all of them take longer to find a composition solution when the number of Web services in a service repository becomes larger and larger, which ranges from 1000 to 5000. The observation from the experimental results is three-fold:

(1) The response time of two algorithms UCLAO* and BHUC increases linearly along with the increasing number of certain Web services. Thus, our two proposed approaches of Web service composition are linear algorithms with the number of certain services in a Web service repository when the number of uncertain services does not change.

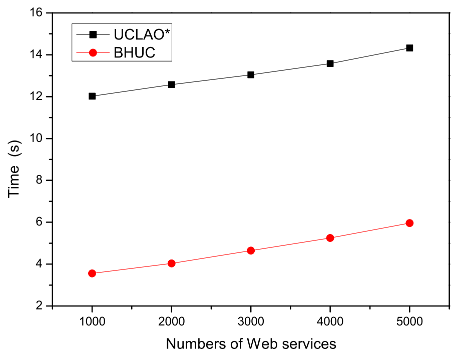
(2) Along with the increasing number of uncertain Web services, the response time of finding an uncertain composition solution using BHUC algorithm increases much slower than that of UCLAO*. In other words, the gap of response time between two algorithms becomes more and more remarkable as the number of uncertain Web services changes in different service repositories, which is shown in Fig.2. More specifically, along with the increasing number of uncertain Web services in four different service repositories, BHUC algorithm takes response time spanning from 1.19 seconds to 10.824 seconds, while UCLAO* algorithm consumes time from 1.254 seconds to 162.017 seconds. Thus, the BHUC algorithm is



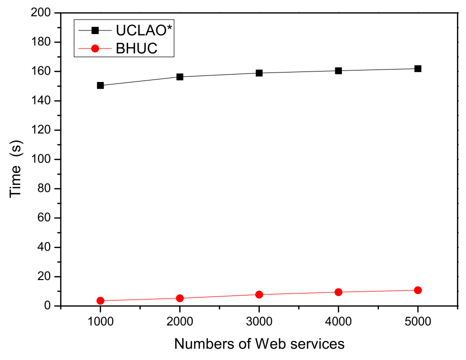
(a) The number of uncertain Web services = 3



(b) The number of uncertain Web services = 4



(c) The number of uncertain Web services = 5



(d) The number of uncertain Web services = 6

Figure 2. The response time of finding a composition solution along with the increasing number of Web services.

faster enough to find an uncertain composition solution with better scalability than UCLAO*.

(3) From the perspective of realistic application, we conclude that although our proposed algorithms can always find a composition solution with all the possible execution plans that can satisfy a composition request, BHUC algorithm is more suitable for the problem of Web service composition with uncertainty in realistic application because of its completeness of finding an uncertain composition solution and good scalability, along with the increasing number of both certain and uncertain Web services in a real-world service repository.

D. Discussion

Based on the experimental results, the comparisons and analysis are summarized among our U-WSC planning algorithms, AO* and LAO* with regard to effectiveness, efficiency and scalability.

(1) Effectiveness. In terms of finding a composition solution, AO* algorithm cannot find a solution, while LAO* algorithm can find an uncertain loop solution but it cannot satisfy a service composition request with uncertain application demands. Therefore, when there are uncertain Web services registered by service providers in a service repository, current non-deterministic planning algorithms cannot be directly applied to solve a U-WSC problem. To tackle this challenge, only our two proposed algorithms, UCLAO* and BHUC, can be both exploited to effectively solve a U-WSC planning problem. The found uncertain composition solution include all the possible execution plans without any cyclic actions.

(2) Efficiency. By comparing the response time between our proposed two algorithms, BHUC algorithm is significantly faster than UCLAO*. The major reason is that BHUC algorithm reduces the state search space when expanding nodes for constructing an implicit search graph. However, the two algorithms can find a composition solution within a very short period of time in our empirical experiments for running example.

(3) Scalability. The fact is that all of the uncertain planning algorithms take much longer time to solve a composition problem along with the increasing number of Web services. However, the response time taken by BHUC algorithm rises obviously slower than that taken by UCLAO* algorithm, as the number of both certain and uncertain Web services changes. Thus, our proposed BHUC algorithm has better scalability for solving a U-WSC planning problem. Consequently, BHUC algorithm is potentially applied in real-world applications.

VI. Related Work

There are some researches about the uncertainty of Web service in Web service composition. Hoffmann et al. presented a conformant planning-based approach to formalize a special case WSC problem [7]. Based on integrity constrains, a WSC problem is translated into a conformant planning problem under uncertainty with multiple possible initial states. The planning problem is solved by Conformant-FF planner. However, it only considers the initial uncertainty rather than uncertain effects of actions. After that, Bertoli et al. modeled a WSC problem as a partially observable non-deterministic planning problem [8], that is fed into a planner called Model Based Planner (MBP) to find a composition solution. Although it realizes the interactions between Web services by belief states and STS, this approach has a disadvantage that it heavily replies on the specific information about the numbers of services and the services' functionality that decreases the automation of uncertain composition of Web services. Recently, H.wang et al. solve a web service composition using single agent or multi-agent reinforcement learning [14][15] and develop Adaptive Service Composition.

To solve this problem, in our recent work we developed an efficient approach for automatic composition of Web service with uncertainty using contingencies [16]. We translate a Web service composition problem as a WSC planning problem in PDDL, where some of contingent actions are taken into account for the uncertainty of stateful Web services. However, we only create artificial actions for several limited uncertain conditional effects of Web services.

Based on above investigations, we propose two algorithms to solve the U-WSC planning problem translated from a U-WSC problem. The proposed algorithms can effectively and efficiently find a composition solution to an uncertain composition problem with good scalability.

VII. Conclusion and Future work

This paper presents a novel approach for uncertain composition of Web services using non-deterministic planning. We model a U-WSC problem into a U-WSC planning problem. Then, two novel algorithms with heuristic state space search, UCLAO* and BHUC, are proposed to solve the U-WSC planning problem. The two algorithms can find a composition solution with all the possible execution paths. Finally, we have conducted empirical experiments on the running example in an E-commerce application. Compared with the existing non-deterministic planning algorithms, the experimental results demonstrate that the proposed uncertain planning algorithms can effectively and efficiently generate a composition solution with better scalability.

As for future work, we plan to combine the functional properties of Web services with those non-functionality criteria in uncertain Web service composition problem. Based on the QoS values of Web services, we make further extension on this work and design new algorithms to solve uncertain service composition problem with the optimal solution.

Acknowledgment

This work was partially supported by National Natural Science Foundation of China (61303096, 61300100), Shanghai Natural Science Foundation (13ZR1454600, 13ZR1451000), Chen Guang project supported by Shanghai Municipal Education Commission, the Fundamental Research Funds for the Central Universities (16D111208), and an Innovation Program of Shanghai Municipal Education Commission (14YZ017).

We thank Robert Mattmuller, Manuela Ortlieb and Malte Helmert for their open sources of AI planner myND planner. We would like to appreciate the anonymous reviewers for their insightful suggestions and constructive comments.

References

[1] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, and Y. Xiang, "Qos-aware dynamic composition of web services using numerical temporal planning," *IEEE Transactions on Services Computing (TSC)*, vol. 7, no. 1, pp. 18–31, 2014.

[2] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven web service composition in meteor-s," in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2004, pp. 23–30.

[3] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering (TSE)*, vol. 30, no. 5, pp. 311–327, 2004.

[4] Y. Li and C. Lin, "Qos-aware service composition for workfow-based data-intensive applications," in *Proceeding of the International Conference on Web Services (ICWS)*, 2011, pp. 452–459.

[5] W. Jiang, C. Zhang, Z. Huang, M. Chen, S. Hu, and Z. Liu, "Qsynth: A tool for qos-aware automatic service composition," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2010, pp. 42–49.

[6] S. C. Oh, D. Lee, and S. R. T. Kumara, "Web Service Planner (WSPR): An effective and scalable Web service composition algorithm," *International Journal of Web Services Research (JWSR)*, vol. 4, no. 1, pp. 1–22, 2007.

[7] J. Hoffmann, P. Bertoli, M. Helmert *et al.*, "Message-based Web service composition, integrity constraints, and planning under uncertainty: A new connection," *Journal of Artificial Intelligence Research (JAIR)*, vol. 35, no. 1, pp. 49–117, 2009.

[8] P. Bertoli, M. Pistore, and P. Traverso, "Automated composition of Web services via planning in asynchronous domains," *Artificial Intelligence (AIJ)*, vol. 174, no. 3, pp. 316–361, 2010.

[9] E. A. Hansen and S. Zilberstein, "Lao*: A heuristic search algorithm that finds solutions with loops," *Artificial Intelligence (AIJ)*, vol. 129, no. 1, pp. 35–62, 2001.

[10] O. Sapena and E. Onaindia, "Planning in highly dynamic environments: an anytime approach for planning under time constraints," *Applied Intelligence (AI)*, vol. 29, no. 1, pp. 90–109, 2008.

[11] B. Bonet and H. Geffner, "Action selection for mdps: Anytime ao* vs. uct," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2012, pp. 1749–1755.

[12] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "Weak, strong, and strong cyclic planning via symbolic model checking," *Artificial Intelligence(AIJ)*, vol. 147, no. 1, pp. 35–84, 2003.

[13] R. Mattmüller, M. Ortlieb, M. Helmert, and P. Bercher, "Pattern database heuristics for fully observable nondeterministic planning," in *Twentieth International Conference on Automated Planning and Scheduling (ICAPS)*, 2010, pp. 105–112.

[14] X. C. Q. Y. Hongbing Wang, Qin Wu, "Integrating gaussian process with reinforcement learning for adaptive service composition," in *Proceedings of the 13th International Conference Service-Oriented Computing (ICSOC)*, 2015, pp. 203–217.

[15] H. Wang, X. Wang, X. Zhang, Q. Yu, and X. Hu, "Effective service composition using multi-agent reinforcement learning," *Knowledge-Based Systems*, vol. 92, pp. 151–168, 2016.

[16] G. Zou, Y. Chen, Y. Xu *et al.*, "Towards automated choreographing of Web services using planning," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2012, pp. 178–184.