Uncertain Composition of Web Services via Non-Deterministic Planning

Sen Niu[1], Guobing Zou[1], Yanglan Gan[2], Zhimin Zhou[1], Bofeng Zhang[1]

[1] School of Computer Engineering and Science, Shanghai University, China
[2] School of Computer Science and Technology, Donghua University, China
{sniu, gbzou}@shu.edu.cn, ylgan@dhu.edu.cn {zmzhou, bfzhang}@shu.edu.cn

Abstract

Web service composition (WSC) is the task of combining a set of single Web services together to create a more complex and cross-organizational composite service. Recently, many researchers have been done on WSC. However, most of these approaches did not take into account the inherent uncertainty of Web services that is the most important nature characteristic due to service deployment and invocation within a real and dynamic Internet environment. Therefore, this paper focuses on uncertain Web service composition (U-WSC) problem by three different non-deterministic effects of Web services. We proposed a comprehensive framework that models a U-WSC problem to a fully observable non-deterministic planning (FOND) problem using our automatic planning transition mechanism. The transformed uncertain planning problem can be solved by a highly efficient off-the-shelf non-deterministic planner which finds a plan to satisfy a composition request. We conducted some experiments based on a case study in an e-commerce application via an off-the-shelf non-deterministic planner called myND. The results of empirical experiments validate the feasibility of our proposed approach for dynamic composition of Web services with functional uncertainty.

Keywords: Web service, Uncertain service composition, Planning transition, Non-deterministic planning

1 Introduction

Web services are implemented as self-adaptive, self-descriptive, modular and well interoperable software components. They can be published by service providers and invoked by service requesters anywhere over the Internet [12]. As the development of cloud computing [24-26] and service-oriented architecture, more and more enterprises and organizations prefer to keep their principal business as Web services and publish them on the web. However, most of Web services are designed to only provide simple functionality. Thus, in most cases single service cannot meet the requirements of real world applications with complex business processes. When no single service has the capability to satisfy a functionality requirement, WSC techniques can be applied to compose several correlative services together for the purpose of fulfilling a complex service requester's business demands [20]. As a result, how to effectively and efficiently compose existing Web services is still an open research issue.

Automated planning in artificial intelligence has proved to be one of the most promising techniques for Web service composition (WSC). Several works in AI planning have addressed different aspects of WSC [7-8, 13-14, 18, 22-23], where a WSC problem is modeled as a classic planning problem. More specifically, available Web services in a Web service repository are transformed to construct a classic planning domain descriptive in Planning Domain Definition Language (PDDL). Also, a composition request can be formalized as a classic planning domain problem in PDDL. In such a case, a WSC planning problem can be solved by a classic automated planner, which generates a composition plan to satisfy a service request. We observe that most of the current investigations suppose that Web services are stateless with deterministic effects. In most cases, however, Web services are actually stateful with multiple uncertain features in Web environment, including non-deterministic effects, uncertain QoS, and failures occurrence. Therefore, with the consideration of uncertainty in Web service composition, uncertain Web service composition (U-WSC) has become a big research challenge in service-oriented real applications.

Investigations on uncertain Web service composition have been made in recent years. A replanning strategy, based on classic planning is proposed in [17]. When unexpected events that cause failures occur, a classic planner is used to recompose available services from an initial state to goal specifications. However, it is difficult to adapt for real-

*Corresponding Author: Guobing Zou, E-mail: gbzou@shu.edu.cn
DOI: 10.3966/160792642018051903008

time applications due to its high time computational complexity and low efficiency, as the number of Web services increase on the Web.

Taking into account initial states uncertainty, WSC problem is formulated as a conformant planning problem [6], which is solved using the FF conformant planner based forward heuristic algorithm. This kind of approaches only considered the uncertainties about initial states, whereas uncertain effects within an action have not been realized. In addition, U-WSC problem is modeled as a partially observable non-deterministic planning problem [2], where Web services descriptive in WS-BPEL are transformed as state transition systems (STSs) and the problem is fed into a Model Based Planner (MBP) to find a solution. The method realizes the interactions between Web services by belief states and STS. However, the strong assumption is that MBP thoroughly reply on predefined Web services with one kind of uncertain action effect. Thus, how to automatically and efficiently find a solution to an uncertain Web service composition problem that are with multiple action effects has become an important research issue to be solved.

To address the above challenges, we proposed a novel approach to solve a U-WSC problem by non-deterministic planning. We first model a U-WSC problem as a fully observable non-deterministic planning (FOND) problem. Then, Web services and their associated operations are converted into corresponding actions with multiple uncertain effects in PPDDL by our domain planning transition mechanism. Accordingly, an uncertain service composition request is also converted into an uncertain planning domain problem in PPDDL by our problem planning transition strategy. Finally, we solve the transformed U-WSC planning problem with a highly efficient state-of-the-art uncertain planner that finds an uncertain composition solution to our original U-WSC problem.

We have conducted an empirical experiment on a case study in an e-commerce application to validate the effectiveness of our proposed approach to uncertain Web service composition based on the planner called myND. The results demonstrated the feasibility of our approach that can be easily deployed with slight modification for real world applications.

The remainder of this paper is organized as follows. In Section 2, we formulate the U-WSC problem and go through preliminaries. Section 3 describes the framework of our approach. In Section 4, we present uncertain composition of Web services using non-deterministic planning. An empirical experiment on a case study is conducted in Section 5. Finally, Section 6 reviews related work on Web service composition, while Section 7 concludes the paper and discusses the future work.

2 Preliminaries

We first formulate uncertain Web service, U-WSC problem by a set of definitions, and then present the descriptive languages WS-BPLE for uncertain Web service and PPDDL in FOND planning.

2.1 Problem Formulation

Web service is inherently uncertain, since they are deployed and invoked in dynamic Internet environment. An uncertain Web service is defined as follows.

Definition 1 (Uncertain Web Service). An uncertain Web service *ws* consists of a finite set of operations, denoted as $ws = \{op_1, op_2, op_3, \cdots\}$. Among all the operations in *ws*, there exists at least one operation, $\exists op_i \in ws$, which is an uncertain operation.

The uncertainty of a Web service is expressed by a set of uncertain operations that are with multiple possible execution effects. The definition of an uncertain operation is as below.

Definition 2 (Uncertain Operation). Given a Web service $ws = \{op_1, op_2, op_3, \cdots\}$, for $\forall op_i \in ws$ it is a 2-tuple $< I, O >$, where $I = \{I^1, I^2, I^3, \cdots\}$ is an input interface parameter set. $O = \{O_1, O_2, O_3, \cdots\}$ is an uncertain output set with multiple possible execution states, where $\forall O_j = \{o_j^1, o_j^2, o_j^3, \cdots\}$ is an execution output state of the operation by a finite set of output interface parameters.

Here, we denote $op.I$ and $op.O$ as input parameters and execution output states of op, respectively. For an operation $op \in ws$, the uncertainty lies in its multiple execution results. Thus, when $j = 1$ is satisfied, the operation is certain; otherwise, the operation is uncertain with $j \geqslant 2$.

Example 1. Let us assume that there are three Web services w_1, w_2 and w_3 published by service providers on the Internet. Their operations and associated input and output parameters are shown in Table 1.

Table 1. Uncertain operations and their associated input and output parameters

Operation	Service	Inputs	Outputs
op_1	w_1	$\{p_1, p_2, p_3\}$	$\{p_4\}, \{p_5, p_6\}$
op_2	w_1	$\{p_4\}$	$\{p_7\}$
op_3	w_2	$\{p_5, p_6\}$	$\{p_8, p_9\}$
op_4	w_3	$\{p_8\}$	$\{p_{10}\}, \{p_{11}\}$
op_5	w_3	$\{p_{11}\}$	$\{p_{12}\}$

Form above example, we observe $w_1 = \{op_1, op_2\}$, $w_2 = \{op_3\}$ and $w_3 = \{op_4, op_5\}$. Since the operation op_1 has two execution output sets $\{p_4\}$ and $\{p_5, p_6\}$, it is an uncertain operation. Similarly, op_4 is also an

uncertain operation, because it has two different execution output states. Other operations have an output set that are certain operations. Therefore, Web services w_1 and w_3 are both uncertain Web services, while w_2 is a certain Web service.

Definition 3 (Uncertain Service Repository). A Web service repository with uncertainty is a set of available services. We denote it as $W = \{ws_1, ws_2, ws_3, \cdots\}$, where $\exists ws_i \in W (i = 1,2,3,\cdots)$, it is an uncertain Web service.

An uncertain Web service repository contains all the available services published by service providers on the Internet. In the example 1, the Web service repository is represented by $W = \{w_1, w_2, w_3\}$. It consists of three available Web services, where w_1 and w_3 are uncertain services and w_2 is an uncertain one.

Definition 4 (Operation Applicability). Given an uncertain operation $op =< I, O >$ and a service state with a set of Web service interface parameters, $S = \{p_i, p_j, p_k, \cdots\}$, op is applicable to S, denoted as $op \preccurlyeq S$, if $op.I \subseteq S$ is satisfiable.

The definition describes the applicability of an operation op to a service state S, if input parameters of the operation are subsumed in the state. In the example 1, if we give a service state $S = \{p_8, p_9\}$, then op_4 is the applicable to S since the input parameters of op_4, i.e., $\{p_8\}$, is subsumed to S.

Definition 5 (Uncertain Dependency). Given two uncertain operations $op_1 =< I, O >$ and $op_2 =< I, O >$, where $O = \{O_1, O_2, O_3, \cdots\}$ is a set of uncertain execution output states. $op_1 \triangleright op_2$ is denoted as the uncertain dependency between op_1 and op_2, if the condition $\exists op_1.O_j \subseteq op_2.I$ is satisfiable, where O_j is one of the execution output state. That is, we have $\exists op_1.O_j$ and $op_2.I \preccurlyeq op_1.O_j$.

From the relationship of uncertain dependency between two operations, if input parameters of an operation are subsumed by a random execution output state of another uncertain operation, then the two operations are dependent with uncertainty. In the example 1, op_1 and op_3 as well as op_4 and op_5 are satisfiable with uncertain dependency.

Definition 6 (Uncertain Request). An uncertain request, R, is a 2-tuple $R = \{R_{in}, R_{out}\}$, where $R_{in} = \{r_{in}^1, r_{in}^2, r_{in}^3, \cdots\}$ is a parameter set provided as initial condition and $R_{out} = \{r_{out}^1, r_{out}^2, r_{out}^3, \cdots\}$ is an uncertain output parameter set provided by a service requester as the goal specifications.

Note that R_{out} always includes all possible execution output states, e.g. success and failure execution states. We assume that $R = \{\{p_1, p_2, p_3\}, \{\{p_8\}, \{p_9\}\}\}$ is set by a service requester in Example 1, where $\{p_1, p_2, p_3\}$ is designated as initial input parameters and $\{\{p_8\}, \{p_9\}\}$ is a goal output result.

Definition 7 (U-WSC Problem). A U-WSC problem is defined as a 3-tuple, $< W, R_{in}, R_{out} >$, where W is an uncertain service repository, R_{in}, R_{out} are respectively the initial state and uncertain goal specifications.

In the example 1, we define a U-WSC problem as $U - WSC =< W, R_{in}, R_{out} >$, where the details of each component are as follows. We have $W = \{w_1, w_2, w_3\}$, $R_{in} = \{p_1, p_2, p_3\}$, and $R_{out} = \{p_{10}, p_{11}\}$. The input parameters are $\{p_1, p_2, p_3\}$ and the desired goal execution output state is $\{p_{10}, p_{11}\}$.

From the above definitions, our research goal is to model a U-WSC problem as an uncertain Web service composition planning problem, which is a classic fully observable non-deterministic planning problem. By doing so, our U-WSC problem can be solved with the leverage of off-the-shelf uncertain planners that can find a solution with multiple uncertain execution paths.

2.2 Uncertain Service and Planning Descriptions

As the fundamental description for Web services with uncertainty, WS-BPEL (Web Services Business Process Execution Language) [1] is an XML based description language, which can enable users to describe business process activities as Web services and define how they can be connected to accomplish specific tasks.

WS-BPEL. In WS-BPEL Web service description, a set of atomic communication operations are combined within a workflow that defines the process implemented by the service. There are two kinds of activities, basic activities and structured activities. The basic activities are mainly responsible for implementing certain atomic functions. Table 2 describes the key activities and their semantics.

Table 2. The key basic activities and their semantics in WS-BPEL

Basic activity	Semantics
invoke	invoke a service operation
reply	send a response
receive	receive the request
assign	copy data
exit	terminate
empty	do nothing

In terms of the structured activities, they describe how a business process is created by composing the basic activities. It consists of three kinds of structured activities, including ordinary sequential control between activities (i.e., sequence, switch and while),

concurrency and synchronization (i.e., flow), and nondeterministic choice based on external events (i.e., pick).

PPDDL. PPDDL (Probabilistic Planning Domain Description Language) was initially used for the probabilistic track in the 4th International Planning Competition. The formal definition of PPDDL and its semantics is given in [3]. It was further extended with an additional non-deterministic statement, i.e. (one of $e_1, e_2, \cdots, e_n$), where each effect e_k 's is an uncertain effect. The semantics is that when executing an effect, it is chosen and applied to the current planning state [14].

3 The Framework of Our Approach

We developed an approach for uncertain composition of Web services using the techniques of non-deterministic planning. Our planning transition strategies as well as uncertain planning of finding a composition solution are integrated into the framework as illustrated in Figure 1.

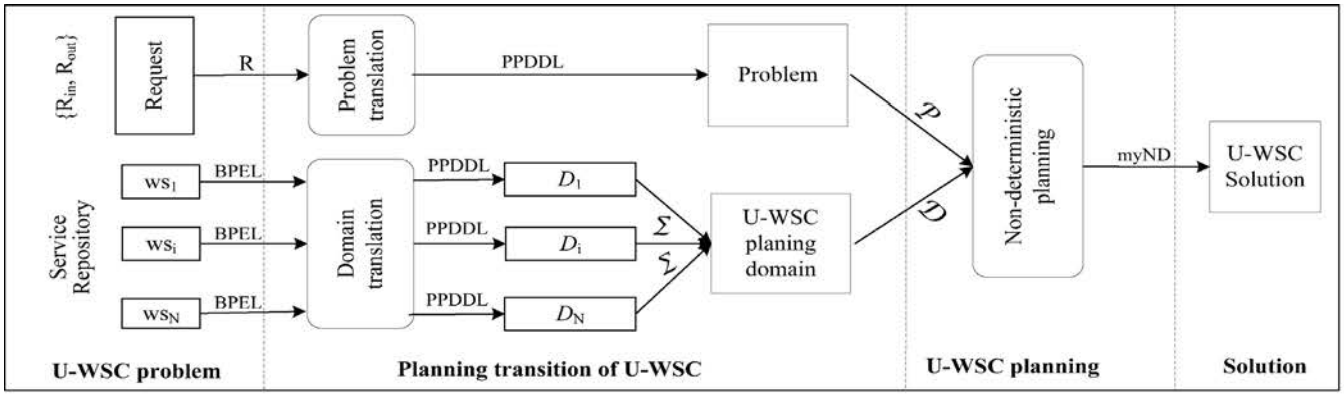


Figure 1. The framework of the approach for uncertain composition of Web services

The input of our framework is a U-WSC problem which involves an uncertain service repository and a composition request, while its output is a solution with multiple uncertain execution paths. Internally, the framework goes through three major steps. (1) Convert an uncertain service repository into a U-WSC planning domain D. (2) Translate an uncertain composition problem into a planning problem P in PPDDL. (3) Apply an efficient non-deterministic planner to solve the transformed U-WSC problem and find a solution with uncertainty.

More specifically, we first convert each Web service in WS-BPEL into a non-deterministic planning subdomain D_i in PPDDL. With the integration of N subdomains $D_1, D_2, \cdots, D_N$, we transformed an uncertain service repository W into a U-WSC uncertain planning domain D. Then, we translate a composition request with a set of input parameters and desired output goal specifications into a planning problem P in PPDDL. Finally, the U-WSC planning problem is fed into a non-deterministic planner myND which automatically finds a composition solution to the given composition request.

4 U-WSC Non-Deterministic Planning

Here, we mainly aim at solving a U-WSC problem via non-deterministic planning. To achieve this goal, uncertain planning transition strategies are proposed to model original problem to a U-WSC planning problem with possibly multiple execution effects of an action in terms of functional uncertainty. Before the planning translation process, a set of formulations around the definition of U-WSC planning problem are as below.

4.1 U-WSC Planning Problem

To apply the non-deterministic planning technique to solve a U-WSC problem, the formalization of a U-WSC planning problem is defined as follows.

Definition 8 (U-WSC State). In a U-WSC problem setting, a set of finite planning variables and predicates $P = \{p_1, p_2, p_3, \cdots\}$ can be extracted from service repository W, R_{in} and R_{out}. A U-WSC state comprises of a set of grounded proposition by P.

In a U-WSC problem, predicates are defined to describe all possible planning states. The predicates can be exacted from input and output parameters of Web services. All possible planning state is also called the U-WSC state.

Definition 9 (U-WSC Action). A U-WSC action is defined as a triple, $a = (name(a), pre(a), eff(a))$, where $name(a)$ is the action's name, $pre(a)$ is a set of propositions as action preconditions, and $eff(a)$ is a set of positive propositions as action effects.

A U-WSC action corresponds to an operation of an uncertain service, i.e., its preconditions and effects are translated from the input and output of an operation. As a result, a U-WSC action has multiple kinds of effects, including certain, uncertain, and conditional

ones.

Definition 10 (U-WSC Planning Problem). A U-WSC planning problem is a 5-tuple, denoted as (S, A, γ, s_0, g), where S is a finite set of U-WSC states, A is a finite set of U-WSC actions, $\gamma: S \times A \to 2^S$ is the non-deterministic transition function between S and A, s_0 and g are the initial state and desired goal specifications, respectively.

A U-WSC planning problem can be expressively represented as a classic fully observable non-deterministic (FOND) planning problem. As illustrated in Figure 1, it consists of a U-WSC planning domain D and a planning problem P in PPDDL.

4.2 Mapping Rules in Uncertain Planning Transition

Given a U-WSC problem $< W, R_{in}, R_{out} >$, we extract each uncertain operation from each Web service ws in W and translate it into an uncertain action a. The mapping rules from an uncertain operation in WS-BPEL and a composition request to its corresponding action and planning problem in PPDDL are shown in Figure 2.

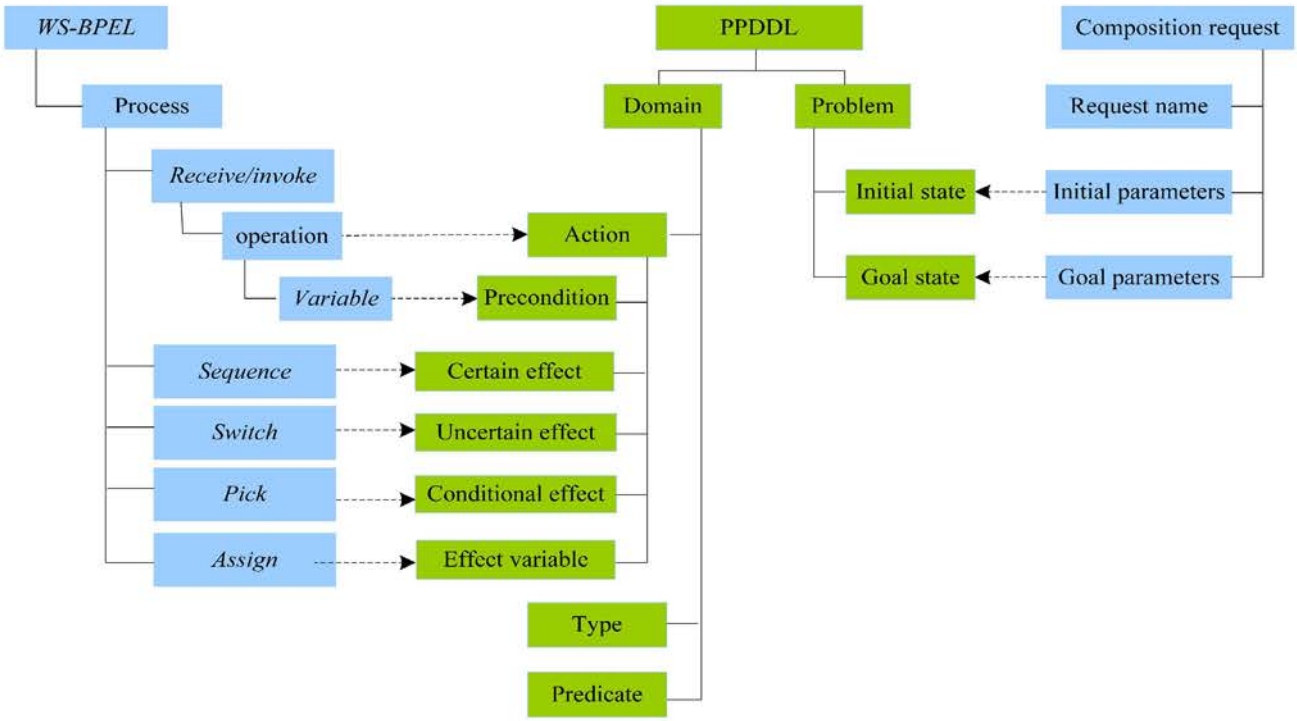


Figure 2. The mapping rules from an uncertain operation and composition request to an action and planning problem

In terms of the basic activities in WS-BPEL, including operations of "receive" and "invoke", they are converted to corresponding U-WSC actions in PPDDL. The planning variables in these operations are translated into action's preconditions. As for the structured activities in WS-BPEL, action effects are divided into three cases, including the transition from "Sequence" to certain effects, "Switch" to uncertain effects, and "Pick" to conditional effects of an uncertain action. We elaborate uncertain planning domain and problem transition processes in subsequent section.

4.3 U-WSC Planning Domain Transition

Based on the mapping relationships from an uncertain operation in a service to an action in Figure 2, we apply a set of transition rules, $Rules = \{R_1, R_2, R_3, R_4, R_5\}$, to U-WSC planning domain transition.

Definition 11 (Transition Rule Set). Given an uncertain Web service ws in WS-BPEL, the planning domain transition maps the service ws to a U-WSC planning subdomain D_i using the following transition rule set $Rules = \{R_1, R_2, R_3, R_4, R_5\}$, where

R_1: If "invoke" or "receive" occurs, create a new U-WSC action a;

R_2: If "sequence" occurs, $eff(a) \leftarrow eff(a) \cup C$, where C are certain effects in a;

R_3: If "switch" occurs, $eff(a) \leftarrow eff(a) \cup U$, where U are uncertain effects in a;

R_4: If "pick" occurs, $eff(a) \leftarrow eff(a) \cup L$, where L are conditional effects in a;

R_5: If "assign", "empty" or "terminate" occurs, add variables V to $eff(a)$.

Given an uncertain service repository W, we propose a planning domain translation algorithm as shown in Algorithm 1, which translates Web services in WS-BPEL to a non-deterministic planning domain in PPDDL with multiple execution output effects in actions.

Algorithm 1: U-WSC planning domain transition

Input: an uncertain service repository $W=\{ws_1, ws_2, ..., ws_N\}$;
Output: a U-WSC planning domain D;
1. $D(A,T,P) \leftarrow \varnothing$;
2. $T \leftarrow$ {: *requirement: typing: non-deterministic*};
3. Foreach $ws_k \in W$ do
4. Apply transition rule R_5, extract variables V;
5. $P \leftarrow P \bigcup V$;
6. Invoke Algorithm 2, extract $A(ws_k)$;
7. $A \leftarrow A \bigcup A(ws_k)$;
8. Extract subdomain D_k;
9. Endfor
10. Combine $D_1, D_2, \cdots, D_N$, we get $D = \bigcup_{i=1}^{N} D_i$;
11. Return D;

The algorithm works as follows. It takes an uncertain service repository in WS-BPEL as an input and the output is a U-WSC planning domain D in PPDDL. We iterate each service ws_k in WS-BPEL that is translated into a set of uncertain actions with all possible non-deterministic effects. By applying transition rule R_5, we can extract variables for predicates P. Then, uncertain actions are extracted from ws (in Algorithm 2) as a subdomain D_k. Finally, we combine these subdomains as a while into a U-WSC planning domain D.

In Algorithm 2, we extract actions from an uncertain service ws_k in WS-BPEL by the transition rules. When basic activities "receive" or "invoke" occurs, we apply R_1 to create a new U-WSC action a and set its precondition extracted from BPEL. When basic activities "assign", "empty", or "terminate" occurs, we apply R_5 to extract variables of these activities that are added into the effects of a. Subsequently, when a structured activity "sequence" occurs, we apply R_2 to add certain effects C to $eff(a)$, recursively. If a structured activity "switch" occurs, we apply R_3 to add uncertain effects U with "one of" statements to $eff(a)$, recursively. Finally, when a structured activity "pick" occurs, we add conditional effects L with "when (condition)" statements to action effects $eff(a)$.

Algorithm 2: U-WSC actions extraction

Input: an uncertain Web service ws_k; domain transition rules $Rules = \{R_1, R_2, R_3, R_4, R_5\}$;
Output: U-WSC actions set $A(ws_k)$;
1. $A(ws_k) \leftarrow \varnothing$;
2. While (ws_k in WS-BPEL is not NULL) do
3. If (R_1 *is satisfiable*) then
4. $a \leftarrow \varnothing$;
5. $pre(a) \leftarrow op.inputvar$;
6. EndIf
7. If (R_2, R_3, R_4 and R_5 *is satisfiable*) then
8. $eff(a) \leftarrow eff(a) \bigcup C$;
9. $eff(a) \leftarrow eff(a) \bigcup U$;
10. $eff(a) \leftarrow eff(a) \bigcup L$;
11. Add Variables V to $eff(a)$;
12. EndIf
13. $A(ws_k) \leftarrow A(ws_k) \bigcup \{a\}$;
14. EndWhile
15. Return $A(ws_k)$;

Based on the above algorithms, an uncertain Web service repository W in WS-BPEL gets translated into a corresponding U-WSC planning domain D in PPDDL.

4.4 Uncertain Planning Problem Transition

Given an uncertain service composition request $R =< R_{in}, R_{out} >$, we devise an Algorithm 3 that is used for generating a non-deterministic planning problem P in PPDDL.

Algorithm 3: Uncertain planning problem transition

Input: An uncertain composition request $R =< R_{in}, R_{out} >$;
Output: A PPDDL problem P;
1. $P(s_0, g) \leftarrow \varnothing$;
2. ForEach $r_{in}^i \in R_{in}$ do
3. $s_0 \leftarrow s_0 \bigcup \{r_{in}^i\}$;
4. ForEach $r_{out}^j \in R_{out}$ do
5 $g \leftarrow g \bigcup \{r_{out}^j\}$;
6. Return P;

In Algorithm 3, it takes an uncertain composition request R as an input and outputs a planning problem P in PPDDL. A PPDDL problem comprises of two parts, i.e., initial state s_0 and goal specifications g. We initially set each of them as $\varnothing$. For each parameter in R_{in} or R_{out}, we put them it into initial state s_0 or goal specifications g, respectively.

4.5 Time Complexity Analysis

Let U-WSC $=< W, R_{in}, R_{out} >$ be a U-WSC problem, where $W = \{ws_1, ws_2, \cdots ws_N\}$ stands for an uncertain Web service repository, including N Web services, $R_{in} = \{r_{in}^1, r_{in}^2, \cdots r_{in}^i\}$ is an input parameters set as initial state s_0 and $R_{out} = \{r_{out}^1, r_{out}^2, \cdots r_{out}^i\}$ is a set of output parameters as goal state g. A U-WSC planning problem consists of a U-WSC planning domain and a planning problem. The former translates each operation in $op \in ws$ as a U-WSC planning domain action a. The latter models $< R_{in}, R_{out} >$ as planning problem P.

The time complexity of U-WSC planning domain transition is determined by the mapping from all of the operations in W to planning actions in A. Its time complexity can be calculated by $T_{domain} = O(\sum_{ws \in W}(T_w + \sum_{op \in ws}(1+|op.I|+|op.O|+4) + T_w))$, where $op.I$ is the number of input variables, $op.O$ is the number of variables in operations and T_w is dominated by the number of services in uncertain Web service repository W. We suppose that $K = \max_{op \in ws}\{|op.I|+|op.O|\}$ is an upper bound on the number of inputvar and variables. We use N and M to denote the number of services in W and the maximum number of operations involved in each service ws. Thus, the time complexity of U-WSC planning domain translation can be recalculated by $T_{domain} = O(\sum_{ws \in W}\sum_{op \in ws}(|op.I|+|op.O|+5)+2*N) = O(N*M*(|op.I|+|op.O)+2*N) = O(N*M*K+2*N)$. In a large Web service repository, since we have $M \ll N$, $K \ll N$, the time complexity of domain transition is $T_{domain} = O(N)$.

The time computational complexity of generating a planning problem is dominated by the size of initial and goal state parameters in $< R_{in}, R_{out} >$. Considering the worst case, there are not any repeated parameters existing among Web services. The time complexity is bounded by $T_{problem} = O(|R_{in}|+|R_{out}|) = O(\sum_{j=1}^{i}(|r_{in}^j| + |r_{out}^j|)) = O(2*i)$. In a composition request, we have $i \ll N$, where N is the number of Web services in W. Thus, the time computational complexity of planning problem translation for uncertain Web service composition is $T_{problem} = O(N)$.

From the time complexity computation and analysis, our proposed approach of U-WSC planning problem translation are almost linear time algorithms in regard to the numbers of services in a given Web service repository. Thus, a U-WSC problem can be efficiently translated into a U-WSC planning problem in polynomial time.

4.6 Finding an Uncertain Solution

By applying U-WSC planning domain and problem translation algorithms, Figure 3 illustrates the translation process from a U-WSC problem to a U-WSC planning problem.

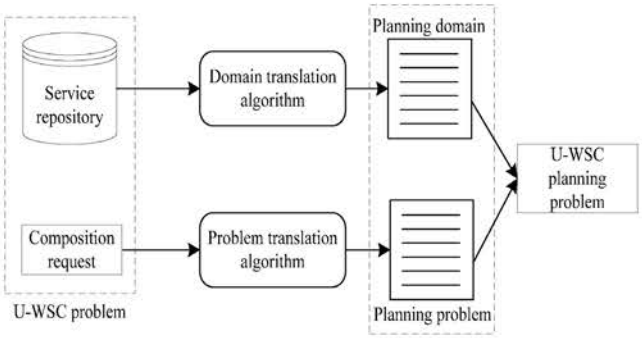


Figure 3. Translation from a U-WSC problem to U-WSC planning problem

In Figure 3, given an uncertain service repository and a composition request, we translate it in WS-BPEL into a U-WSC planning problem $< P, D >$ in PPDDL. Thus, the planning domain and planning problem in PPDDL compose the U-WSC planning problem.

Given a U-WSC planning problem $< P, D >$ in PPDDL, its solution plan consists of a set of available paths, each of which can execute from the initial state to a possible output state with the invocations of U-WSC actions in D.

Definition 12 (Path Satisfiability). Given an initial state R_{in} and an execution output state $r \in R_{out}$, if there exists a sequence of U-WSC actions that is denoted as $\prod = < a_1, a_2, \cdots, a_m >$ in D, which executes from R_{in} to the desired output state r, then we denote $< a_1, a_2, \cdots, a_m > \propto \{R_{in}, r\}$ as path satisfiability.

A solution plan for a U-WSC planning problem involves all possible paths that state from initial state to multiple execution output states. It is defined as below.

Definition 13 (U-WSC Solution). Given a U-WSC problem with $R_{out} = \{r_1, r_2, \cdots, r_k\}$, an uncertain composition solution to the problem is a set of action sequences $\{\prod_1, \prod_2, \cdots, \prod_k\}$. $\forall \prod_i = \{a_1, a_2, \cdots, a_k\}$, $\prod_i \propto \{R_{in}, r_i\}$ is path satisfiable.

From the U-WSC solution, we observe that with the combination of all the action sequences $\prod_1, \prod_2, \cdots, \prod_k$, a solution plan takes into account those execution paths from initial state to all possible uncertain output states.

To solve a U-WSC planning problem, we apply a state-of-the-art uncertain planner myND [9] that uses AO^* and LAO^* [5] search guided by the canonical PDB heuristic. The planner can support PPDDL with "one of" statement and solve a U-WSC planning problem to find a U-WSC solution with a set of actions

sequences, which are path satisfiable.

Note that we mainly focus on how to translate a U-WSC problem into a U-WSC planning problem, instead of designing an efficient uncertain planning algorithm to find a solution. However, we solve the translated U-WSC planning problem by an existing uncertain planner myND, because it has better performance than other uncertain planning based non-deterministic planners.

5 Empirical Experiment

To validate the feasibility and effectiveness of our proposed approach, we have conducted empirical experiments where models and transition strategies are implemented in Java for the framework of solving a U-WSC problem via the techniques of non-deterministic planning. A case study has been done from an e-commerce shopping application, whose dataset are collected from ws-Toolset [2]. We first translate a U-WSC problem to a U-WSC planning problem using our proposed transition algorithms, then compare the time consumption during transition phase and solution phase, respectively. In general, there are four main workflow patterns, including sequential, switch, parallel and iterative in service composition. In this paper, we focus on the sequential and switch patterns. However, others can be easily transformed into these two workflow patterns. The empirical experiments are conducted on a PC with Intel Dual Core 2.8 GHZ processor and 3G RAM in Windows 7.

5.1 E-commerce Shopping Problem

Our reference example aims to provide a furniture purchase & delivery composed service by combining two independent existing services, including a furniture purchase service Producer, and a delivery service Shipper. The composed service allows a service role User to ask for expected products that can be delivered at a desired location. As a consequence, the composed service interacts with three available Web services, i.e., *Producer*, *Shipper*, and *User*. The three services are interacted with each other by the composed service as illustrated in Figure 4.

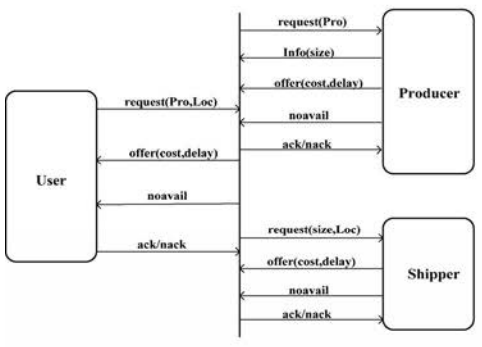


Figure 4. The interaction relationship among service roles User, Producer and Shipper

The interaction process is as follows. The Producer initially accepts a request for a given product. If the requested product is available, it provides product size. Then, if the requester acknowledges his/her interest to buy it, the Producer makes an offer with a cost and production duration. Also, this offer can be either accepted or refused by the requester. In both cases, the Producer terminates its execution with success or failure, respectively. For the Shipper, it receives a request for transporting a product of a given size to a desired location. After its checking, if the delivery is possible, the Shipper provides an offer with a cost and delivery time. It can be accepted or refused by the external service that has invoked the Shipper. As for the User, it sends his/her requests to get a given product at a given location, and then it gets either a refusal or an offer, indicating the price and the time required for the service. The User may either accept or refuse the offer from its interactive services.

The composed service is to sell a product at a destination as requested by a customer. To achieve this goal, we have to consider possible execution situations and try to reach a situation where the three interactions reach a successful state. However, the goal may be not always achievable because the product is not available or the location is out of the area of service of the Shipper. Therefore, the composed service must consider all the situations with success or failure.

5.2 Shipper Uncertain Planning Transition

With the U-WSC planning domain and problem transition, we translate the Web services User, Producer and Shipper in WS-BPEL to a U-WSC planning domain using the proposed algorithms 1 and 2. Here, we take the Shipper service as an empirical experiment, which can be converted into a set of actions around shipping activities. Part of the Shipper in WS-BPEL and its corresponding non-deterministic planning domain are illustrated in Figure 5 and Figure 6, respectively.

In Figure 5, from its business process description, the Shipper consists of a set of operations, including $\{request, not\_avail, offer\}$, where "*request*" and "*offer*" are uncertain operations and "*not_avail*" belongs to a certain operation. When the Shipper invokes "*request*", it may return $\{noAvail\}$ or offer $\{cost, delay\}$. Similarly, after the invocations of "*not_avail*" and "*offer*", they return $\{FAIL\}$ and $\{(doNothing), (FAIL)\}$, respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="Shipper"
        ...
  <sequence name="main">
    <receive partnerLink="Customer" portType="tns:Shipper_PT"
      operation="request" variable="customer_req" createInstance="yes">
    </receive>
    <switch name="checkAvailability">
      <case condition="isNotAvailable">
        <sequence>
          <sequence>
            <assign name="prepareNoAvail">
              <copy>
                <from variable="customer_req" part="key"/>
                <to variable="noAvail" part="key"/>
              </copy>
            </assign>
            <invoke partnerLink="Customer"
              portType="tns:Shipper_CallbackPT"
              operation="not_avail" inputVariable="noAvail">
            </invoke>
            <terminate name="FAIL"/>
          </sequence>
        </sequence>
      </case>
      <otherwise>
        ...
          <pick>
            <onMessage partnerLink="Customer"
              portType="tns:Shipper_PT" operation="ack" variable="ack">
              <empty name="doNothing"/>
            </onMessage>
            <onMessage partnerLink="Customer"
              portType="tns:Shipper_PT" operation="nack" variable="nack">
              <terminate name="FAIL"/>
            </onMessage>
          ...
</process>
```

Figure 5. Part of the description of the Shipper in WS-BPEL

In Figure 6, the U-WSC subdomain has three actions, including *request*, *offer* and *not_avail*. The action "*request*" has uncertain effects with "*one of*" statement, while the action "*offer*" has conditional effects with "*when*" condition statement. The action "*not_avail*" has a certain effect.

```
(define (domain shipper)
        (:requirements :typing :non-deterministic)
        (:predicates
                (size)(location)(noAvail)(cost)
                (delay)(ack)(nack)(doNothing)(FAIL)
        )

(:action request
 :precondition (and(size)(location))
 :effect (oneof
        (and(noAvail))
        (and(cost)(delay)))
)
(:action offer
 :precondition (and(cost)(delay))
 :effect (and
        (when(ack)(doNothing))
        (when(nack)(FAIL)))
)
```

Figure 6. U-WSC planning domain for Shipper in PPDDL

We can generate a PPDDL problem from an uncertain composition request with the reference of domain. Figure 7 illustrates a planning problem in PPDDL from a composition request.

```
(define     (problem Shipper_problem)
            (:domain Shipper)
            (:init (and (size)(location)))
            (:goal (or (FAIL)(doNothing)))
)
```

Figure 7. The planning problem in PPDDL for Web service shipper

In Figure 7, *size* and *location* are as initial states and a set of states *FAIL* (failure) or *doNothing* (success) as goal states in the domain problem of service shipper.

5.3 Finding An Uncertain Solution

We apply a highly efficient uncertain planner myND to solve the U-WSC planning problem. Taking the U-WSC planning problem in PPDDL as input, the planner finds a solution with multiple possible execution paths. The result of U-WSC planning problem for service shipper is illustrated in Figure 8.

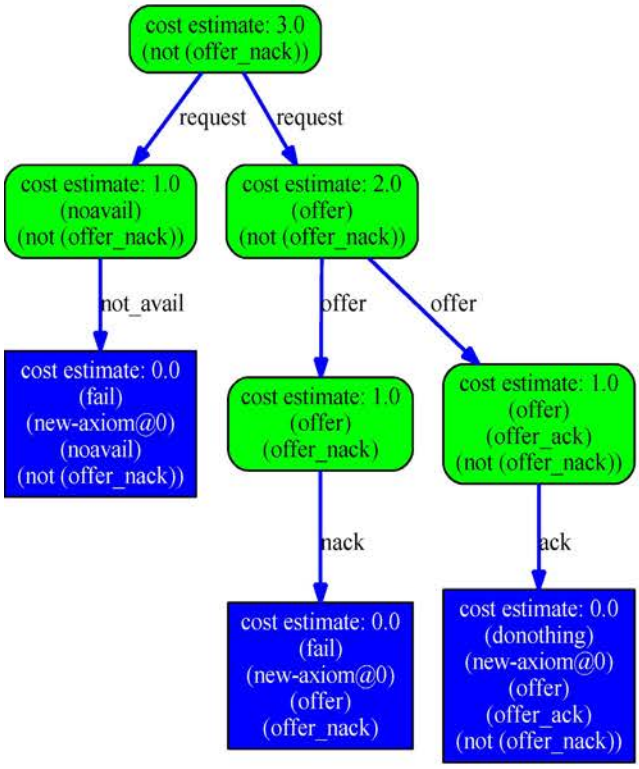


Figure 8. The process of shipper actions with uncertainty

From the result, we can easily know the all possible workflow processes of operations in shipper. There are three possible paths:

(1) $< request, offer, ack >$;

(2) $< request, offer, nack >$;

(3) $< request, notavail >$.

More specifically, when shipper gets a request, it will be to check the available. If the available is false, the shipper invokes the operation *not _avail* and return fail. Otherwise, the shipper invokes the operation *offer*. If the shipper receives a respond of *ack*, the result returns success. Otherwise, the result is fail. The process of shipper actions can be transformed into an abstract states diagram in Figure 9.

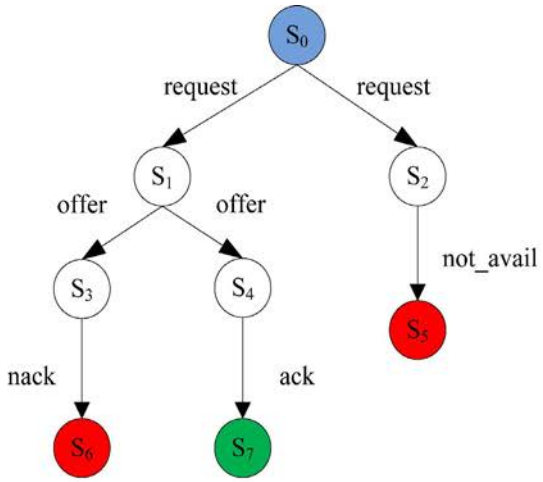


Figure 9. The abstract states diagram of shipper

In Figure 9, every state represents a set of state space in solution. The root node represents the initial state and the leaf nodes represent the states of fail or success.

We translate all Web services and user's request in E-commerce Shopping Problem into a U-WSC planning domain and a planning problem. The solution about the shopping problem is shown as abstract states diagram in Figure 10.

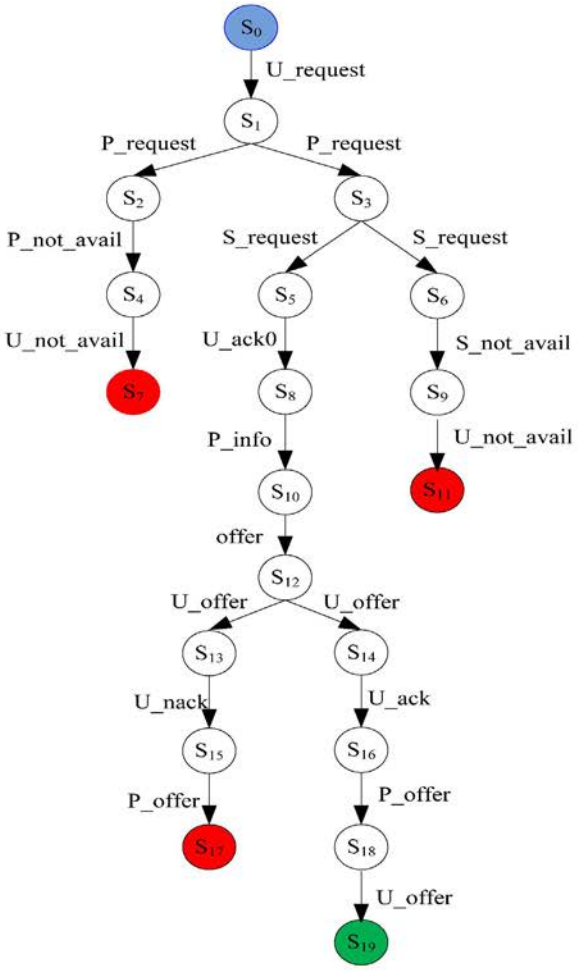


Figure 10. The solution for the U-WSC planning problem using myND

From the solution, we observe that all the paths from initial state to an execution output state reach a success or failure state. There are four paths in the U-WSC solution.

(1) <U_request, P_request, P_not_avail, U_not_avail>;

(2) <U_request, P_request, S_request, S_not_avail, U_not_avail >;

(3) <U_request, P_request, S_request, U_ack0, P_info, offer, U_offer, U_nack, P_offer>;

(4) <U_request, P_request, S_request, U_ack0, P_info, offer, U_offer, U_ack, P_offer, U_offer>.

More specifically, when the user provides the request including products and location, we invoke the Producer and check whether it is available. If it is not available, we get a failure state. Otherwise, we get information about product size, and then invoke a shipping request and check whether it is available. When the shipping can be available, we can offer the cost and delay to the User. If the User acknowledges the cost and delay, we invoke the offer operation and reach the goal state. Otherwise, the resulting reaches the possible failure states.

5.4 Performance Analysis

From the view of practicability in real-world applications, the response time is of vital importance to a U-WSC method, because it determines whether a feasible composition solution can be rapidly returned to users within a short period of time. Therefore, we employ the response time as the evaluation metric to compare the approach in transition phase and solution phase, respectively.

First, we compare our transition approach with state transition system approach [2] in terms of time consumption during planning transition phase. Our transition approach translates Web services from WS-BPEL to PPDDL, while state transition system approach translates Web services from WS-BPEL to state transition systems (STS). We chose four different Web service in WS-BPEL and translate them into PPDDL and STS 10 times for each one. We compute the average time in 10 times and the results are shown in Table 3.

Table 3. The comparison of time consumption on planning transition between two approaches

Service name / Transition time (ms) / Approach	Shipper	Producer	Hotel	Flight
BPEL2STS	401.8	386	398.4	394.4
BPEL2PPDDL	113.6	103	119.6	101.6

From the results in Table 3, we observe that the transition time consumption of BPEL2PPDDL is better than that of BPEL2STS during the process of generating different uncertain planning domains of

Web services. The average time of BPEL2STS and BPEL2PPDDL is 395.15 ms and 109.45 ms, respectively. As a result, our proposed uncertain planning transition algorithm perform more efficiently when it makes transition for uncertain Web services in WS-BPEL than STS.

Then, we compare the Zero-heuristic with Canonical PDB heuristic based on AO^* algorithm from myND planner during the phase of finding an uncertain solution.

We solve the U-WSC planning problem about the case by the AO^* algorithm based on Zero-heuristic and Canonical PDB heuristic. The response time includes preprocess time, search time and total time. The corresponding time is shown in Figure 11.

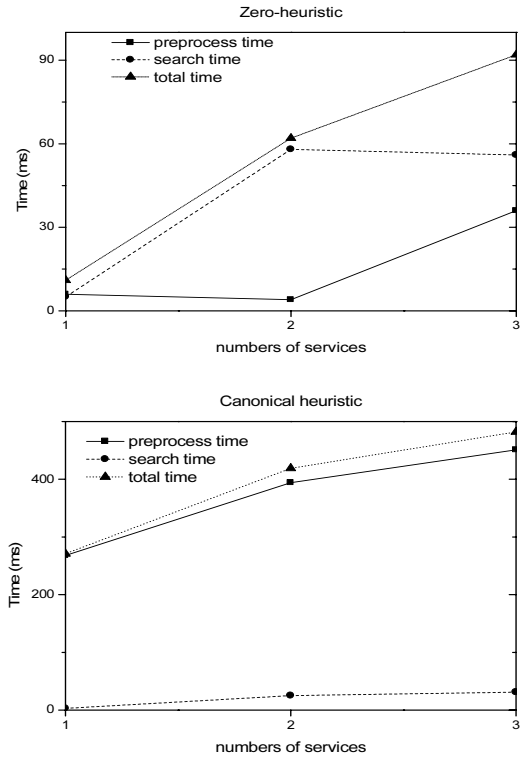


Figure 11. The response time of finding a solution between Zero-heuristic and Canonical heuristic search

From the response time of comparing the algorithms Zero-heuristic and Canonical PDB heuristic search, the time increases along with the increasing of the number of the services, because the U-WSC planning problem is getting more complex.

The experimental results of time comparison between Zero-heuristic with Canonical PDB heuristic is illustrated in Figure 12, including preprocessing time, search time and total time of finding a service composition solution to the given uncertain composition request with all possibly uncertain execution plans.

In Figure 12, the preprocess time of Zero-heuristic search is more faster than the time of Canonical PDB heuristic search, because the PDB heuristic search is based on the state relevance and need more time to prepare the relationship in states. But the search time of the Canonical PDB heuristic search is very lower than the time of Zero-heuristic search. To be more precise, the search time of using Canonical PDB heuristic search ranges from 3ms to 31ms; the search time of using Zero-heuristic search ranges from 5ms to 58ms. The total time of using Canonical PDB heuristic search is longer because of the preprocess time.

Based on the experimental results and time compare analysis, it comes to a conclusion that our approach translating a U-WSC problem to a corresponding U-WSC planning problem and solving the U-WSC planning problem using the non-deterministic planner is effective.

6 Related Work

Automated composition of Web services aims to integrate a bunch of correlative functionally independent services and combine them as a whole to satisfy a complex demand in real world applications. Recently, many approaches have been applied to tackle WSC problems, while AI planning techniques play an important role in different scenarios, including classic service composition, QoS-aware dynamic composition of Web services and uncertain composition of Web services.

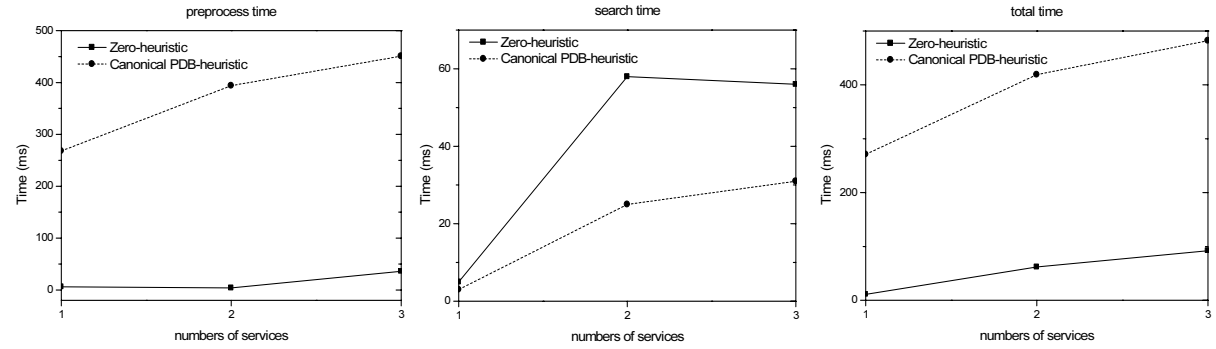


Figure 12. The time comparison in every phase

Using classic planning for Web service composition, Web service planner (WSPR) [10-11] goes through two phases including forward search and regression search to find a feasible composition solution. During the search process, a heuristic function is used to choose a service with the biggest contribution to match a subgoal. However, it cannot guarantee an optimal composition solution with the minimum number of services. A service composition algorithm is proposed by planning graph model in [19]. The process of finding a composition solution is the construction of a planning graph. It just selects a subset of services in order for new planning graph level, which possibly incurs redundant Web services in a feasible composition solution. Recently, we proposed an efficient approach for automatic composition of Web services using the state-of-art AI planners [21], where a WSC problem is regarded as a WSC planning problem. However, the approach cannot handle uncertain composition in Web services.

For QoS-aware service composition, [4] proposed a partial selection approach that is able to reduce the size of search space by dominance relationships and constraint validations at candidate level. However, the approach is based on workflow model and cannot compose the service automatically proposed an improved genetic algorithm based approach to optimize the overall QoS of the service composition [15]. The approach exploits the genetic algorithm to simulated annealing and uses the heuristics harmony search function which changes the algorithm parameters to improve the conventional techniques. In [23], We also proposed a novel approach that can automatically convert a QoS-aware composition task to a planning problem with temporal and numerical features, and then solve this planning problem using a self-developed planner. The approach can find a composite service graph with the optimal overall QoS value while satisfying multiple global QoS constraints.

The next, we review the state-of-the-art works on WSC via non-deterministic planning. Initially, Hoffmann et al. presented a planning-based approach to formalize a special case WSC problem [6]. It takes integrity constraints as background theory specified by ontology to describe domain constraints between objects and their properties. Based on integrity constrains, a WSC problem is converted into a conformant planning problem under uncertainty with all of the possible initial states. An empirical experiment has been conducted by Conformant-FF planner. However, they are lack of the consideration of uncertain effects of actions. After that, Bertoli et al. modeled a WSC problem as a partially observable non-deterministic planning problem [2], where Web services descriptive in WS-BPEL are transformed as state transition systems (STSs). The translated planning problem is fed into a planner called Model Based Planner (MBP) to find a solution. Although the method realizes the interactions between Web services by belief states and STS, it heavily reply on the Web service repository that decreases the automation of uncertain composition of Web services when large scale services are available on the Internet. Recently, our research developed an efficient approach for automatic composition of Web service with uncertainty using contingencies [20-22]. 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 services. However, we only create artificial actions for uncertain effects.

Although lots of works have been done in Web service composition, most of them mainly focus on certain Web service without any stateful execution efforts and they do not consider the uncertainty of Web service in WSC. Some efforts on uncertain Web service composition have been made in recent years and presented the solution via non-deterministic planning. However, these works are time-consuming and strongly reply on predefined Web services.

Based on above investigations, we propose a novel approach for uncertain composition of Web services via non-deterministic planning. After the transition from a U-WSC problem to a U-WSC planning problem that is solved by highly efficient uncertain planners. It can be deployed with good scalability in real applications.

7 Conclusion and Future Work

We proposed a novel approach for solving uncertain composition of Web services using the techniques of non-deterministic planning. We give a whole framework that models a U-WSC problem to a fully observable non-deterministic planning problem. Then, we convert Web services in WS-BPEL into actions with multiple uncertain efforts in PPDDL by our domain planning transition rules. Also, an uncertain composition request was converted into a planning problem in PPDDL by our problem planning transition strategy. Finally, taking the U-WSC planning problem as an input, we apply a highly efficiently uncertain planner myND to find a solution.

A case study in e-commerce real-world application is conducted to validate the feasibility and efficiency of our proposed approach. The results has demonstrated that our method via non-deterministic planning can be potentially deployed for real applications. As future work, we plan to extend non-deterministic planning algorithms in myND planner and make them more robust especially for dealing with cycle occurrence in a composition solution.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China (61303096, 61300100), Shanghai Natural Science Foundation (13ZR1454600, 13ZR1451000, 18ZR1414400), a Specialized Research Fund for the Doctoral Program of Higher Education (20133108120029), platform fund of PAPD and CICAEET, 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 also appreciate all of the anonymous reviewers for their insightful suggestions and useful comments that will significantly improve the quality of our manuscript.

Sen Niu and Guobing Zou contributed equally to this study and share first authorship.

References

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana, *Business Process Execution Language for Web Services*, Version 1.1. Specification, May, 2003.

[2] P. Bertoli, M. Pistore, P. Traverso, Automated Composition of Web Services via Planning in Asynchronous Domains, *Artificial Intelligence*, Vol. 174, No. 3-4, pp. 316-361, March, 2010.

[3] D. Bryce, O. Buffet, 6th International Planning Competition: Uncertainty Part, *Proceedings of the 6th International Planning Competition*, Sydney, Australia, *2008*, pp. 1-6.

[4] Y. Chen, J. Huang, C. Lin, Partial Selection: An Efficient Approach for QoS-Aware Web Service Composition, *Proceedings on IEEE International Conference on Web Services (ICWS)*, Anchorage, AK, 2014, pp. 1-8.

[5] E. A. Hansen, S. Zilberstein, LAO*: A Heuristic Search Algorithm that Finds Solutions with Loops, *Artificial Intelligence*, Vol. 129, No. 1-2, pp. 35-62, June, 2001.

[6] J. Hoffmann, P. Bertoli, M. Helmert, M. Pistore, 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, May, 2009.

[7] M. Klusch, A. Gerber, Fast Composition Planning of OWL-S Services and Application, *Proceedings of the European Conference on Web Services (ECOWS)*, Zurich, Switzerland, 2006, pp. 181-190.

[8] M. Klusch, A. Gerber, M. Schmidt, Semantic Web Service Composition Planning with OWLS-XPlan, *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web (AAAI)*, Arlington, Virginia, 2005, pp. 1-8.

[9] R. Mattmüller, M. Ortlieb, M. Helmert, P. Bercher, Pattern Database Heuristics for Fully Observable Nondeterministic Planning, *Proceedings of the twentieth International Conference on Automated Planning and Scheduling (ICAPS)*, Toronto, Canada, 2010, pp. 1-8.

[10] S.-C. Oh, D. Lee, S. R. T. Kumara, Web Service Planner (WSPR): An Effective and Scalable Web Service Composition Algorithm, *International Journal of Web Services Research*, Vol. 4, No. 1, pp. 1-22, January-March, 2007.

[11] S.-C. Oh, D. Lee, S. R. T. Kumara, Effective Web Service Composition in Diverse and Large-scale Service Networks, *IEEE Transactions on Service Computing (TSC)*, Vol. 1, No. 1, pp. 15-32, January-March, 2008.

[12] J. Rao, X. Su. A Survey of Automated Web Service Composition Methods, in: J. Cardoso, A. Sheth (Eds.), *Semantic Web Services and Web Process Composition, Lecture Notes in Computer Science (LNCS)*, Springer, 2005, pp. 43-54.

[13] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, HTN Planning for Web Service Composition Using SHOP2, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 1, No. 4, pp. 377-396, October, 2004.

[14] D. Wu, B. Parsia, E. Sirin, J. Hendler, D. Nau, Automating DAML-S Web Services Composition Using SHOP2, *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, Sanibel Island, FL, 2003, pp. 195-210.

[15] A. E. Yilmaz, P. Karagoz, Improved Genetic Algorithm based Approach for QoS Aware Web Service Composition, *IEEE International Conference on Web Services (ICWS)*, Anchorage, AK, 2014, pp. 463-470.

[16] H. L. S. Younes, M. L. Littman, *PPDDL1. 0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects*, CMU-CS-04-167, October, 2004.

[17] E. Zahoor, O. Perrin, C. Godart, Disc: A Declarative Framework for Self-healing Web Services Composition, *Proceedings of the 10th IEEE International Conference on Web Services (ICWS)*, Miami, FL, 2010, pp. 25-33.

[18] W. Li, Z. Ye, X. Zhao, J. Jiang, Q. Jin, Probability Modeling and Functional Validation of Dynamic Service Composition for Location Based Services with Uncertain Factors, *Journal of Internet Technology*, Vol. 15, No. 4, pp. 653-643, July, 2014.

[19] X. Zheng, Y. Yan, An Efficient Syntactic Web Service Composition Algorithm based on the Planning Graph Model, *Proceedings of IEEE International Conference on Web Services (ICWS)*, Beijing, China, 2008, pp. 691-699.

[20] G. Zou, Y. Chen, Y. Xu, R. Huang, Y. Xiang, Towards Automated Choreographing of Web Services Using Planning, *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, Toronto, Canada, 2012, pp. 178-184.

[21] G. Zou, Y. Gan, Y. Chen, B. Zhang, Dynamic Composition of Web Services Using Efficient Planners in Large-scale Service Repository, *Knowledge-Based Systems*, Vol. 62, pp. 98-112, May, 2014.

[22] G. Zou, Y. Gan, Y. Chen, B. Zhang, R. Huang, Y. Xu, Y.

Xiang, Towards Automated Choreography of Web Services Using Planning in Large Scale Service Repositories, *Applied Intelligence*, Vol. 41, No. 2, pp. 383-404, September, 2014.

[23] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, 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, January-March, 2014.

[24] Y. Ren, J. Shen, J. Wang, J. Han, S. Lee, Mutual Verifiable Provable Data Auditing in Public Cloud Storage, *Journal of Internet Technology*, Vol. 16, No. 2, pp. 317-323, March, 2015.

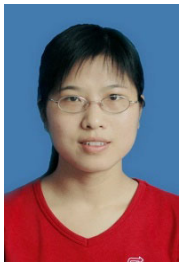
[25] Z. Xia, X. Wang, X. Sun, Q. Wang, A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 2, pp. 340-352, February, 2016.

[26] Z. Fu, X. Sun, Q. Liu, L. Zhou, J. Shu, Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing, *IEICE Transactions on Communications*, Vol. E98-B, No. 1, pp. 190-200, January, 2015.

Biographies

Sen Niu is a member of IEEE (93919367). He is currently a Ph.D. candidate in the School of Computer Engineering and Science, Shanghai University, China. His research interests include Web service composition, service computing, artificial intelligence and automated planning. He has published a paper on SCC 2016.

Guobing Zou is an associate professor in School of Computer Engineering and Science at Shanghai University, China. His research interests include service computing and data mining. He has published around 45 papers on international journals and conferences, including IEEE TSC, Knowledge-Based Systems, Applied Intelligence, Soft Computing, AAAI, and SCC.



Yanglan Gan is an associate professor in School of Computer Science and Technology at Donghua University, China. Her research interests include data mining and bioinformatics. She has published around 20 papers on international journals and conferences, including Bioinformatics, BMC Bioinformatics, IEEE/ACM Transactions on Computational Biology and Bioinformatics, and Neurocomputing.

Zhimin Zhou received his undergraduate degree from School of Internet of Things Engineering, Jiangnan University, China. He is currently a master student in School of Computer Engineering and Science, Shanghai University, China. His research interests include service computing, uncertainty of service QoS recommendation.

Bofeng Zhang is a full professor in School of Computer Engineering and Science at Shanghai University, China. His research interests include social computing and artificial intelligence. He has published around 120 papers on international journals and conferences. He served as program chair or PC for varieties of international conferences.