There Can Be Composite Services When You Believe

Yu Lei

***Abstract*—Web services on the internet are distributed, heterogeneous, autonomous and dynamic, which leads to uncertainty of web services. These uncertain factors affect success rate of service composition. The web service composition problem should be considered as an uncertain planning problem. Therefore, we use an uncertainty planning method to deal with the uncertain planning problem for service composition. Without knowing complete information, our method uses an estimated value function to obtain a composite service. We give details and examples that how to compose services by our method. In addition, we shows relations among user requirements, service invocations and Quality of services. The experiments verify the validity of the algorithm.**

***Index Terms*—Quality of Web service; incomplete information; partially observable markov decision process; temporal difference learning**

I. INTRODUCTION

Who knows what composite services you can achieve, when you believe somehow you will. You will when you believe. They occasionally happen when you ask, and it is easy to give in to your fear, but when you are blinded by unstable QoS, cannot see your way safe through the workflow, POMDP is still a resilient voice, says mashup is very near.

Web services on the internet are distributed, heterogeneous, autonomous and dynamic, which leads to two kinds of uncertainty of web services. Uncertainty of web services refers to the uncertain results of invoking services and the uncertain QoS (Quality of Services) values, which are hard to precisely predict and affect the process of service composition.

For the first uncertainty, behaviors of web services or results of web services are the main reasons [1]. For the second uncertainty, values of QoS, such as response time and availability etc., are uncertain because other factors such as network delays constantly change QoS values. This phenomenon exits and is generally solved by providing an average value [2], but we suggest that using uncertain QoS values with probability distribution is more realistic.

One more significant and similar example is e-Marketplace, which is a dynamic supply chain for textile industry [24]. In addition, uncertain and dynamic environments are widespread, especially in wireless networks, and some researchers [19-21] already have studied on the service composition in their specific areas.

This paper is submitted on 2015.12 for review.
Yu Lei is with the Inner Mongolia University, Hohhot, China.

When taking into account the uncertain and dynamic nature of real-world cases, the following considerations have practical significance for reliable services composition: how to compose services based on implicit knowledge of business logics and how to model uncertain QoS values to ensure that composite services have higher QoS.

Because of the uncertainty of web services and QoS constraints, web service composition must take into account the results of services to dynamically adjust web service composition. Therefore, web service composition should not only be a composition plan under certain specific restrictions, but also be an optimal policy of web service composition in certain conditions. Focusing on uncertainty and QoS constraints of web services, this article views the problem of web service composition as an uncertainty planning by Partially Observable Markov Decision Processes (POMDP) [3].

Partially Observable Markov Decision Processes is a decision-making method solving a planning problem, which is an extension of Markov Decision Processes (MDP). POMDP assumes that the information of the system states is just partially knowable and cannot be observed directly, thus it models the system with incomplete information, and then makes decisions based on the current incomplete information. POMDP has a wide range of application areas [4]. Inspired by reinforcement learning [5], we use a learning method to compose web services in uncertain environments.

Contributions in this paper:

Integrating with reinforcement learning, we give details of how to use POMDP to model the problem of service composition in the uncertain environment.

After comparing three learning algorithms from both theoretical perspective and experimental perspective, and indicating sufficiency of these methods, we propose a method and verify the performance and adaption of this algorithm in the domain of service composition.

The rest of this paper is organized as follows. The second section introduces related research on the service composition based on MDP. The third section describes service composition based on POMDP. The fourth section proposes a learning algorithm to address the service composition problem (i.e. the POMDP problem). The fifth section shows extensive experiment results of the service composition. The final section summarizes the paper.

II. RELATE WORK

There are a variety of service composition methods based on the Artificial Intelligence planning, such as classic planning which is based on state-space [6], neoclassic planning which is based on propositions and constraints satisfaction, heuristics

and control policy planning which is based on heuristics function [7], logic planning which is based on model checking [8], graph planning which is based on Graphplan [9, 10], and uncertainty planning which is based on Markov Decision Processes.

Gao et al. [11] modeled service composition based on MDP, viewing a task collection as a state, and viewing success rate of invocations as transition probabilities. They compared and analyzed two algorithms that can solve MDP, which are the forward value iteration algorithm and the afterward value iteration algorithm, also they proved that the forward value iteration algorithm is more suitable for dynamic web services environments.

We are tracking the studies of Doshi. Doshi et al. [1] proposed a scene that a retailer requests products by querying product inventory from many providers, and then they used MDP to compose services that have the uncertain business logic, afterwards used a Bayesian learning method to learn the transition probability among services. Based on the previous literature, they [12] further proposed a composition method of hierarchical semi-Markov decision process, where they considered the service execution time and added the ability to decompose tasks. Recently they [13] proposed a method combined with risk preferences to adapt a composite service to different types of users.

Chen et al. [14] transferred the service composition problem into the artificial intelligence planning problem, and then combined MDP with Hierarchical Task Network (HTN) to solve the problem. The method first decomposed service requirements into sub-requirements by means of HTN, and then used MDP to solve the sub-requirements. The article indicated that the fewer are constraints of a service, the bigger is the probability that the service is selected. Wang et al. [15] proposed an adaptive approach for service composition, which mapped services to actions of MDP and constructed a map of service network. Each path generated by the method is a workflow path. Furthermore, to prove the adaptability of the composition method, the paper also conducted experiments and analysis on learning efficiency. Wang et al. [16] went a step further by combining the Semi-Markov Processes with the preference logic to solve the service composition problem. Semi-Markov Processes was used for modeling service execution time, and the preference logic was used for modeling the preference degree that different users prefer the different QoS criteria. Li et al. [17] extended a finite state machine model and introduced a probabilistic computation tree logic to denote MDP properties, such as the states accessibility and the probability of success for service composition, and then used a probabilistic model checker, PRISM, to analyze and verify reliability and cost of composed service. Fan et al. [18] proposed a new method for measuring QoS to adaptively update QoS values, and designed a reliable composition algorithm of web service based on MDP.

The current insufficiency of the service composition methods based on MDP is the assumption that QoS values can be accurately provided, but due to uncertain network environments and uncertain invocation results of Web services, this assumption is too optimistic. Therefore, this paper proposes a new service composition method based on learning algorithm to deal with this problem.

III. Problem Formulation

In our problem of web service composition, the possible structure of the composite service is predefined, and multiple services with the same functionality have been grouped already. Our goal is to select a best business process and its services, and the principle of selection is based on rewards that will be described below.

A. Model parameters in *POMDP*

Web service composition can be modeled by POMDP as follows:

S: State, a finite set of states of the world [15]. Suppose we have a service for an example. The state is whether or not the service is our needed service:

S = {Yes, Not}

A: Action, namely web service invocation. Results of web service invocations are uncertain.

The actions can be to do nothing, invoke the service or query a service center about this service.

A = {No-op, Invoke, Query}

T: $S \times A \times S \rightarrow [0,1]$. T is a state transition function, expressing the probability distribution that the current services transit to the next services after an invocation of the current services.

The state transition functions:

T(StartState, No-op, EndState)		
	EndState	
StartState	Yes	Not
Yes	0.98	0.02
Not	0.00	1.00

T(StartState, Query, EndState)		
	EndState	
StartState	Yes	Not
Yes	0.98	0.02
Not	0.00	1.00

T(StartState, Invoke, EndState)		
	EndState	
StartState	Yes	Not
Yes	1.00	0.00
Not	1.00	0.00

R: $S \times A \times S \rightarrow \mathbb{R}$, R is a reward function, which means the reward that we take action A in the current service S and transit to the next service S'.

The reward function:

R(Action, StartState)		
	StartState	
Action	Yes	Not
No-op	+2	-10
Query	-6	-6
Invoke	-18	-16

Z: Observations that are likely to be observed. They are also a collection of QoS ranks. The QoS criteria values have different dimensions and ranges, and the relative importance of the criteria is not assigned, hence the QoS criteria must be normalized to one single dimension (or utility) [22] to compare which service is better than others.

Z = {Query-Yes, Query-Not, Invoke-Yes, Invoke-Not}

O: S ×A ×Z→[0,1]. O is an observation function.

The observation probability functions:

O(No-op, State, Observation)				
	Observation			
State	Query-Yes	Invoke-Yes	Query-Not	Invoke-Not
Yes	0.970	0.000	0.030	0.000
Not	0.025	0.000	0.975	0.000

O(Query, State, Observation)				
	Observation			
State	Query-Yes	Invoke-Yes	Query-Not	Invoke-Not
Yes	0.000	0.999	0.000	0.001
Not	0.000	0.010	0.000	0.990

O(Invoke, State, Observation)				
	Observation			
State	Query-Yes	Invoke-Yes	Query-Not	Invoke-Not
Yes	0.250	0.250	0.250	0.250
Not	0.250	0.250	0.250	0.250

H: Stage, implied by the discount factor [23]. It refers to the number of planning steps. To achieve maximum benefits, plans should not only take into account the current action, but also the following actions. Discount factor is $\gamma \in [0,1]$, which specifies the decreased discount value when the steps increase. γ^h is the discount value in step h.

B: Belief state [23]. It is a vector of states. The belief state B consists of all states. $B = \{b_0, b_1, \ldots, b_n\}$, and b_0 is initial belief that denotes where the service process should start. Because states are partially observable, some observations are perceived only after we take an action. In our example, we have:

Pr(s = Yes) = 0.75
Pr(s = Not) = 0.25

B. *Partially Observable Service Composition*

POSC (Partially Observable Service Composition) is a POMDP problem, or a belief-MDP problem. POSC = {B, A, Z, Γ, ρ}. The service composition can be described by this 6-tuples, and a generated optimal policy is a composed service. As we mentioned earlier, B is a belief state in the continuous space, meaning the belief degree that we select one service. The value of belief state is $b(s) \in [0,1]$, where $\sum_s b(s) = 1$. All belief states form a geometric simplex with $|S-1|$ dimensions. A and Z respectively are the finite sets of actions and observations, the same as those of MDP. $\Gamma: B \times A \times Z \rightarrow B$ is a belief state transition function, meaning the probability of being the belief state B', when taking the action A in the current belief B and then obtaining the observation Z. $\rho: B \times A \rightarrow R$ is a reward function in the belief space, namely $\rho(b,a) = \sum s\ R(s,a) \times b(s)$.

Solving POMDP problems is to select an approximate optimal policy. Policy π is a function mapping belief state b ($b \in B$) into action a ($a \in A$):

$$\pi(b) \rightarrow a \tag{1}$$

For methods that not use belief states to infer the states, instead maintain a historical sequence, the policy also can be defined as:

$$\pi(b_0, h_t) \rightarrow a \tag{2}$$

h_t is the history at the moment t, b_0 is the initial belief state. It's necessary to consider future h steps before making a decision. In addition, two problems need to be considered, that are how to decide a policy is the optimal policy, and how to compare two policies. Therefore, a criterion must be specified to decide what the optimal policy is. This paper uses the maximum-expected total-discount-benefits criterion to measure the policy, $E[\sum_h R(s_h, a_h) \times \gamma^h]$.

Value function is a function mapping each action in a state into a reward so that we can see which action is better than others [23], thus it indicates how good one service invocation is. The optimal value function of POMDP reflects, from a long-term point of view, how many benefits a service contributes, as shown in formula 5. All of optimal value functions constitute an optimal policy.

$$V_t^*(b) = \max_a [\rho(b,a) + \gamma \sum_{z \in Z} P(z|b,a) V_{t-1}^*(b')] \tag{3}$$

Optimal policy is a strategy that decides to take which action in each step (or state) [23], thus it indicates how good one composite service is. Its strategy is equal to a composite service. Its value reflects the overall rewards of this composite service, as shown below.

$$\pi_t^*(b) = \arg\max_a [\rho(b,a) + \gamma \sum_{z \in Z} P(z|b,a) V_t^*(b')] \quad (4)$$

The optimal policy may be:

Pr(State=Yes)	Action
0.000 to 0.237	invoke
0.237 to 0.490	query
0.490 to 0.512	query
0.512 to 0.713	no-op
0.713 to 0.912	no-op
0.912 to 0.989	no-op
0.989 to 1.000	no-op

IV. Generate Composite Services based on Learning Algorithms

A. Iterative formula

Belief state B' at time t can be denoted by belief state B at time t-1 by means of recursive calculation. When we take action A and then obtain observation Z, belief state B will be updated to B' based on Bayes rule, as follows:

$$\begin{aligned} b' = P(s'|b,a,z) &= \frac{P(s',b,a,z)}{P(b,a,z)} \\ &= \frac{P(z|s',b,a)P(s'|b,a)P(b,a)}{P(z|b,a)P(b,a)} \\ &= \frac{P(z|s',a)P(s'|b,a)}{P(z|b,a)} \\ &= \frac{P(z|s',a)\sum_{s\in S}P(s'|s,a)\times b(s)}{P(z|b,a)} \end{aligned} \quad (5)$$

Where $P(z|b,a)$ can be obtained by Equ.6.

$$P(z|b,a) = \sum_{s' \in S} P(z|s',a) \times \sum_{s \in S} P(s'|s,a) \times b(s) \quad (6)$$

By iteratively updating belief state, web services can be composed according to the following circulate procedure: (1) when state is s_{t-1} and belief state is b_{t-1}; (2) we select actions a_{t-1}; (3) obtain observations z_{t-1} and rewards r_{t-1}; (4) move to a new service s_t in accordance with state transition function T(s,a,s'), and then belief state is updated to b_t. New service s_t is selected for the next service.

B. Using reinforcement learning

Reinforcement learning is a kind of Machine Learning algorithm in the Artificial Intelligence domain. It uses the observed reward to learn an optimal policy (or approximate optimal policy), in order to make the cumulative reward maximum.

When our method selects an action, it needs to reach a compromise between exploring (exploration) new strategies and taking advantage of (exploitation) the optimization strategies that are already retrieved, in order to avoid local optimal solutions.

We propose a Time-based Learning method (TL) that is suitable for service composition. The method decides how and when to explore and exploit. Its basic idea is to decline the number of exploration over time and to increase the number of exploitation over time.

$$TL(t) = \max_{a_{t+1}} \begin{Bmatrix} \frac{1}{\ln t}\left[Q(b_{t+1},a_{t+1}) - Q(b_t,a_t)\right] \\ \ln t \times Q(b_{t+1},a_{t+1}) \end{Bmatrix} \quad (7)$$

In the formula above, TL function will obtain the maximum value from the exploration (left part of max) and the exploitation (right part of max), i.e. the times of trying new services will decrease, and the times of using retrieved services will increase over time. Time-based learning method is shown below:

Algorithm: Time-based learning method

Inputs: the current belief b_{t+1} and reward r_{t+1}

Outputs: an action a_{t+1}

Variables: Q(b,a), which is a table storing belief b and action a.

t, a point in time.

α, a learning rate.

b_t, a_t and r_t are the previous belief, action and reward, initially null

1. Steps:
2. Begin
3. If b is not null
4. Then

5. $TL(t) = \max_{a_{t+1}} \begin{Bmatrix} \frac{1}{\ln t}\left[Q(b_{t+1},a_{t+1}) - Q(b_t,a_t)\right] \\ \ln t \times Q(b_{t+1},a_{t+1}) \end{Bmatrix}$
6. $Q(b_t,a_t) \leftarrow Q(b_t,a_t) + \alpha[r_{t+1} + \gamma TL(t) - Q(b_t,a_t)]$
7. Increase t
8. End If
9. If b_{t+1} is terminal
10. Then b_t, a_t, $r_t \leftarrow$ null
11. Else
12. $b_t \leftarrow b_{t+1}$
13. $a_t \leftarrow a_{t+1}$
14. $r_t \leftarrow r_{t+1}$
15. End If
16. End Begin

In the 11th-14th line, we iteratively update the belief, action and reward. A strategy determines a composite service. We compose services by learning strategies.

V. SCENARIO

We give an example of service composition in the uncertain environment. Supposing there are two Web services, and each has one method, say method A and method B. One of the two methods can return data required by invokers. The invokers have following choices and corresponding rewards.

Table.1 Rewards, actions and states

Rewards (R)	Method B returns the needed data (Nb)	Method A returns the needed data (Na)
Invoke method A (Ia)	-100	80
Invoke method B (Ib)	90	-100
Invoke nothing (In)	-1	-1

We assume the probability that method B can return the needed data is *p*, and the probability that method A can return the needed data is 1-*p*. Then we have the reward of invoking one method:

$$R(Ia) = R(Ia,\ Nb)\times p + R(Ia,\ Na)\times(1-p) = -100p + 80(1-p)$$

$$R(Ib) = R(Ib,\ Nb)\times p + R(Ib,\ Na)\times(1-p) = 90p - 100(1-p)$$

According to the above values, we draw two lines corresponding to *Ia* and *Ib* separately in the following figure. This figure shows that the expected rewards and optimal actions depend on the value of *p*. If *p* = 0, then R(*Ia*) = 80 and R(*Ib*) = -100, thus we should invoke method A.

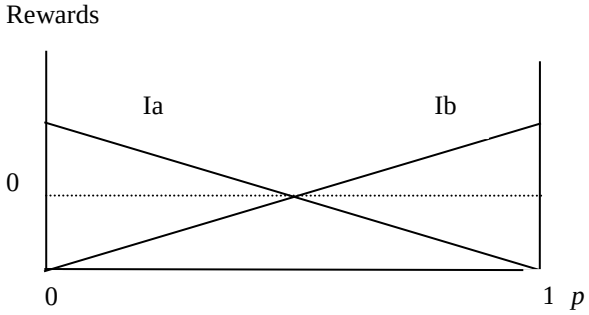

Fig.1 Initial rewards

Supposing we have two ranks of QoS statistic values, say good QoS (*Qg*) and medium QoS (*Qm*). If the probability of good QoS of method B is 0.7, then the probability of medium QoS of method B is 0.3. Likewise, we give the following assumption:

$$\begin{cases} P(Qg \mid Nb) = 0.7 \\ P(Qm \mid Nb) = 0.3 \\ P(Qg \mid Na) = 0.3 \\ P(Qm \mid Na) = 0.7 \end{cases}$$

If we obtain the value of *Qg*, then we change our belief of invoking method B. This change can be described by:

$$p' = P(Nb \mid Qg) = \frac{P(Qg \mid Nb)P(Nb)}{P(Qg)} = \frac{0.7p}{0.7p + 0.3(1-p)}$$

If we cannot sure which method is better, we can stay and obtain more accurate and latest values either from the Web service or a central server. In this case, the reward of staying is -1 and we have new rewards of invoking method A and method B. The rewards of actions are given below.

$$\begin{aligned} &R(Ia \mid Qg) \\ &= R(Ia, Nb)P(Nb \mid Qg) + R(Ia, Na)P(Na \mid Qg) \\ &= -100p' + 80(1-p') \end{aligned}$$

$$\begin{aligned} &R(Ib \mid Qg) \\ &= R(Ib, Nb)P(Nb \mid Qg) + R(Ib, Na)P(Na \mid Qg) \\ &= 90p' - 100(1-p') \end{aligned}$$

$$R(In \mid Qg) = -1$$

Likewise, if we obtain the value of *Qm*, then we change our belief and rewards. These changes can be described by:

$$\begin{cases} R(Ia \mid Qm) = R(Ia, Na)P(Na \mid Qm) + R(Ia, Nb)P(Nb \mid Qm) \\ R(Ib \mid Qm) = R(Ib, Na)P(Na \mid Qm) + R(Ib, Nb)P(Nb \mid Qm) \\ R(In \mid Qm) = -1 \end{cases}$$

Our aim is to find the optimal action with maximum rewards in certain QoS. Through the following formula we can achieve our aim.

$$\begin{aligned} &\sum_j \left[\max_i R(I_i \mid Q_j) \right] P(Q_j) \\ &= \max \begin{bmatrix} R(Ia \mid Qg) \\ R(Ib \mid Qg) \\ R(In \mid Qg) \end{bmatrix} \times P(Qg) \\ &+ \max \begin{bmatrix} R(Ia \mid Qm) \\ R(Ib \mid Qm) \\ R(In \mid Qm) \end{bmatrix} \times P(Qm) \end{aligned}$$

We can deduce that the probability of *Qg* is P(*Qg*) = 0.7*p* +0.3*p*(1-*p*). Therefore, we have new rewards after choosing stay as the following shows.

$$\sum_j \left[\max_i R(I_i \mid Q_j) \right] P(Q_j) = \max \begin{bmatrix} -100 + 80(1-p) \\ -43p - 46(1-p) \\ 33p + 26(1-p) \\ 90p - 100(1-p) \end{bmatrix}$$

Consequently we have the expected rewards as the following figure shows. This figure means we have better choices than we have in Fig.1. After staying and obtaining more values, this figure shows the price of information. In this figure, we should choose one optimal action from two immediate actions and one discounted future action.

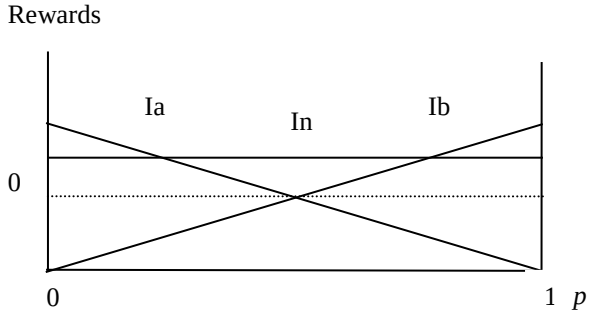

Fig.2 Optimal actions

VI. EXPERIMENTS

Our simulation computer has following configurations: 2.13GHZ Intel Core2 and 2GB RAM. In the simulation, we randomly generate transitions, rewards and observations. The related discount factor is 0.9, learning rate is 0.2, and ε is 0.6.

A. Comparative methods

One traditional method and three learning-based methods are introduced. Three learning-based methods are compared with our algorithm, and we introduce their iterative formula respectively.

1) Dynamic programming

Dynamic programming solves our problem by breaking it down into a collection of states, and then solves each states just once and stores results. The following figure shows the search paths of dynamic programming.

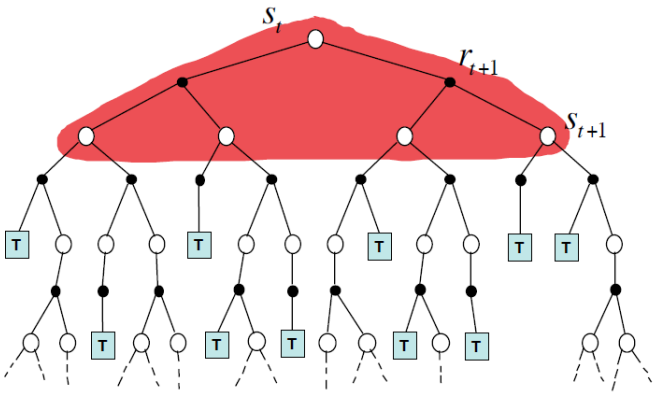

Fig. 3 search paths of Dynamic programming

Dynamic programming is not suit for our problem due to its time complexity is very high.

2) Monte Carlo method

Monte Carlo method repeatedly uses formula 8 (an estimated value function) to progressively approach the real value function.

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \quad (8)$$

In the formula 8, α is a learning rate, a predefined constant. R_t is a reward after time *t*. The following picture shows the search paths of Monte Carlo.

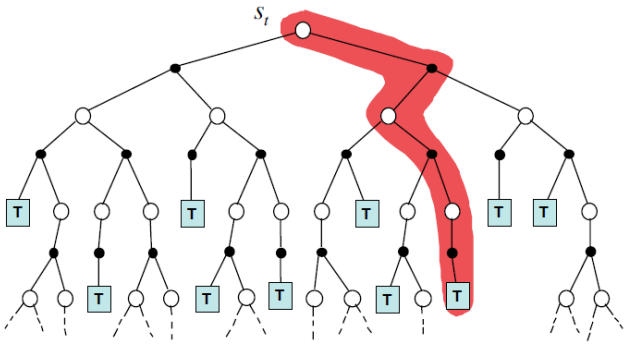

Fig. 4 search paths of Monte Carlo

3) Temporal Difference method

The simplest Temporal Difference is one-step forward method, namely TD(0). TD(0) simply modifies the values of adjacent states.

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (9)$$

The following picture shows the search paths of TD(0). TD(0) is more efficient than Monte Carlo due to Monte Carlo begins learning only after a complete learning cycle.

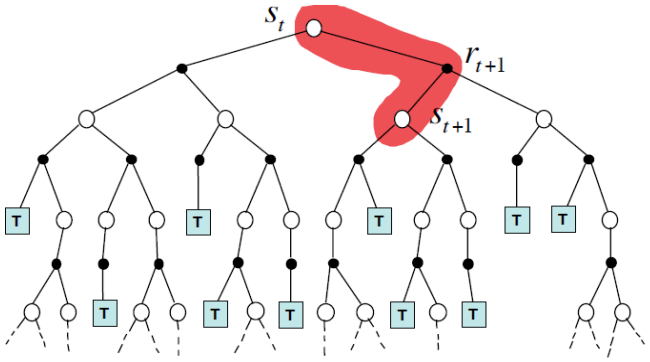

Fig. 5 search paths of TD(0)

4) Q-learning method

Q-learning estimates the value function based on actions, $Q(s_t,a_t)$, to learn the optimal policy. The estimated value function of Q-learning is:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \quad (10)$$

Q-learning method needs to reach a compromise between exploration and exploitation in order to avoid local optimal solutions. The previous literatures use a ε-greedy policy to select the action ($0<\varepsilon<1$). This probability can ensure that each action has opportunities to be explored. Q-learning method greedily selects (with ε probability) the retrieved action that has the maximum $Q(b_t,a_t)$, and explores (with 1-ε probability) new actions. It can guarantee the actions with maximum $Q(b_t,a_t)$ have higher priority to be selected.

Q-learning accelerates the speed of convergence, meanwhile it ensures the effective learning. Q-learning is an advanced Temporal Difference method, and the difference between TD(0) and Q-learning is that Q-learning select the maximum actions as the following picture shows.

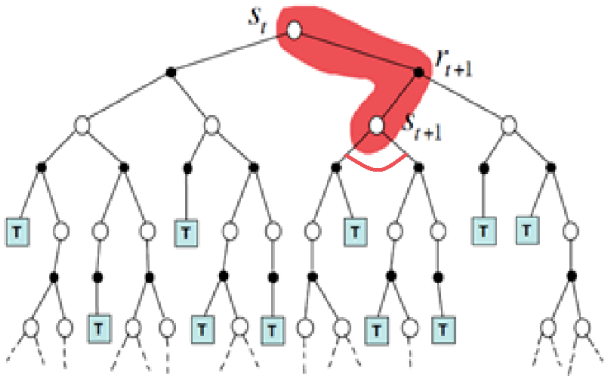

Fig. 5 search paths of Q-learning

B. Simulation Results

Each service has four candidate services in the Table 2 and Table 4, which show that Q-learning and TD(0) are more efficient than Monte Carlo method. The reason is that TD(0) only modifies values of adjacent states. Monte Carlo method begins to iterate after a whole learning cycle, whereas TD(0) and Q-learning only need to search one step forward. Q-learning is slightly better than TD (0), because Q-learning uses a maximum action to be the current optimal action. TL is better than these methods. Because, with time passing, TL tries less new services and tries more existing optimization services to find the best composition.

Table 2 Comparison of learning speed w.r.t. services

#Service	Monte Carlo (Episodes)	TD(0) (Episodes)	Q-learning (Episodes)	TL (Episodes)
100	80	45	41	30
300	130	95	90	63
500	202	155	135	102
700	310	241	205	156
900	420	377	312	271

Table 3 Comparison of learning speed w.r.t. candidate services

#Candidate Service	Monte Carlo (Episodes)	TD(0) (Episodes)	Q-learning (Episodes)	TL (Episodes)
1000	101	94	82	67
2000	203	180	150	126
3000	310	270	231	190
4000	440	382	350	262
5000	612	525	496	381

A learning cycle (Episode) is a process of learning policy, and a process of service composition. The initial policy is not optimal at the beginning. As the learning cycle increases, the policy continues to be optimized until it is convergent. Convergence means a method has learned the best policy (i.e. the best plan for service composition). The number of convergent learning cycle is the number of cycles to learn the best policy. Each learning cycle will go from the initial state to the final state.

In summary, not only the experiments support our ideas, but also the theory shows advancement of Q-learning as the following figure shows.

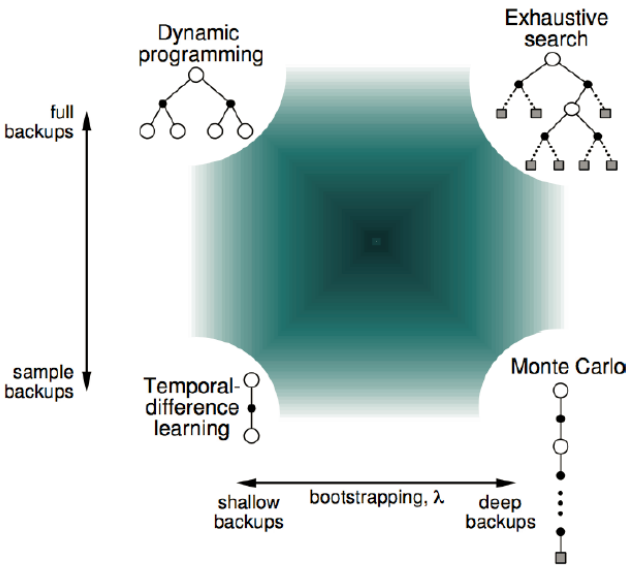

Fig. 6 comparison of learning methods

Dynamic programming and exhaustive search are full backups, but they take too much time to obtain the optimal results. Whereas Temporal difference learning and Monte Carlo method use sample backups to reduce the complexity of problem (Google AlphaGo takes the same strategy). Temporal difference learning (Q-learning) just uses efficient shallow backups, meanwhile it finally keeps convergence.

The experiments show that more parameters make the model more realistic, TL method improves success rate of the service composition and reduces computing time. We randomly generated transition graphs, whose number of services is between 1000 and 4000, and number of candidate services is between 1000 and 10000.

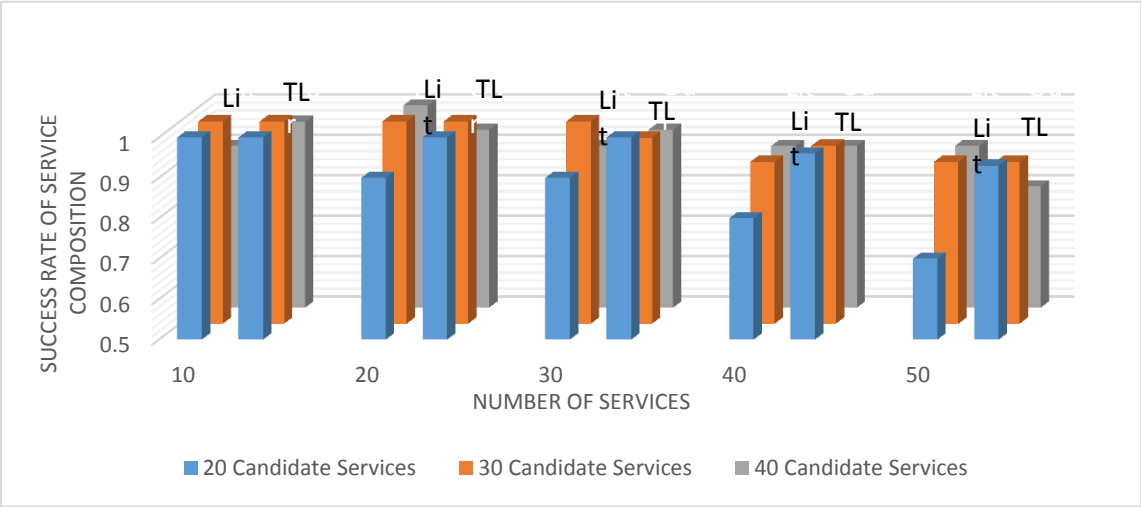

Fig. 7. Comparison of success rate of service composition

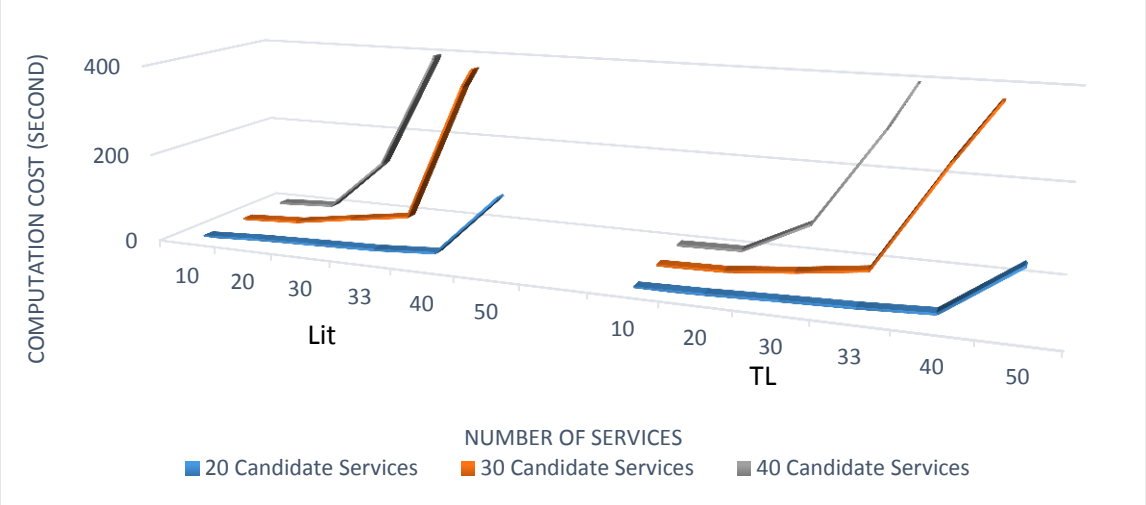

Fig. 8. Comparison of computing time

The methods used in the paper [18] and in this paper are compared in two aspects: success rate of service composition and computation cost (computing time). Lit is the result of the paper [18], TL is the result of the method this paper used. Fig. 7 shows that our success rate of service composition in uncertain environments is higher than that obtained by [18].

Fig. 8 shows that, as the number of services increases, computing time of TL method is significantly shorter than that in the [18]. With the number of services dramatically increases, the computing time of the traditional methods dramatically increase.

Comparing our results with the results of literature [15], we observed that the cumulative reward is converged to the optimal policy very fast. From Table 4 and Table 5, we can see that our method is faster than that of literature [15].

Table 4 Convergence time with respect to services

# Services	Lit (Episodes)	TL (Episodes)
1000	82	62
1500	103	86
2000	136	116
2500	155	125
3000	203	156
3500	258	188
4000	301	230

Table 5 Convergence time with respect to candidate services

#Candidate Services	Lit (Episodes)	TL (Episodes)
1000	310	282
2000	602	553
3000	1001	820
4000	1520	1112
6000	2030	1532
7000	2806	1757
10000	6080	2470

VII. CONCLUSION

To solve the web service composition problem in uncertain environments, this paper proposes a time-based learning algorithm, which is a dynamic decision-making method. Our method uses an estimated value function to obtain a composite service. The proposed method is applied to the real scenario. We give details and examples that how to compose services by our method. In addition, we shows relations among user requirements, service invocations and Quality of services. Comparing to the methods based on MDP, the experiments show that our method achieves higher success rate of service composition and shorter computing time.

ACKNOWLEDGMENT

This work was supported by Natural Science Foundation of Inner Mongolia Autonomous Region (2015BS0603), Scientific projects of higher school of Inner Mongolia (NJZY009), Programs of Higher-level talents of Inner Mongolia University (215005145143).

REFERENCES

[1] P. DOSHI, R. GOODWIN, R. AKKIRAJU, and K. VERMA. Dynamic workflow composition using Markov decision processes. International Journal of Web Services Research, 2005, 2(1): 1-17.

[2] LUO YuanSheng, YANG Kun, TANG Qiang, ZHANG Jianmin, and XIONG Bin. A multi-criteria network-aware service composition algorithm in wireless environments. Computer Communications, 2012, 35(15): 1882-1892.

[3] C. RAPHAEL and G. SHANI. The Skyline algorithm for POMDP value function pruning. Annals of mathematics and artificial intelligence, 2012, 65(1): 61-77.

[4] L. AGUSSURJA and H. C. LAU. A POMDP Model for Guiding Taxi Cruising in a Congested Urban City. Lecture Notes in Artificial Intelligence, 2011, 7094: 415-428.

[5] K. A. Yau, P. KOMISARCZUK. Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues. Journal of Network and Computer Applications, 2012, 35(1): 253-267.

[6] WU Bin, DENG Shuiguang, LI Ying, WU Jian, and YIN Jianwei. AWSP: An Automatic Web Service Planner Based on Heuristic State Space Search. Proceedings of IEEE International Conference on Web Services, USA: IEEE, 2011: 403-410.

[7] P. RODRIGUEZ-MIER, M. MUCIENTES and M. LAMA. Automatic web service composition with a heuristic-based search algorithm. Proceedings of IEEE International Conference on Web Services, ICWS, Washington, DC, USA: IEEE, 2011: 81-88.

[8] FENG Yuzhang, A. VEERAMANI and R. KANAGASABAI. Automatic DAG-based service composition: A model checking approach. Proceedings of IEEE International Conference on Web Services, ICWS, Honolulu, HI, USA: IEEE, 2012: 674-675.

[9] JIANG Wei, HU Songlin, D. Lee, GONG Shuai, and LIU Zhiyong. Continuous Query for QoS-Aware Automatic Service Composition. Proceedings of IEEE International Conference on Web Services, (ICWS), USA: IEEE, 2012: 50-57.

[10] YAN Yuyong, CHEN Min and YANG Yubin. Anytime QoS optimization over the PlanGraph for web service composition. Proceedings of Annual ACM Symposium on Applied Computing, Trento, Italy, USA: ACM, 2012:1968-1975.

[11] GAO Aiqiang, YANG Dongqing, TANG Shiwei, and ZHANG Ming. Web service composition using markov decision processes. Proceedings of International Conference on Advances in Web-Age Information Management, WAIM, Hangzhou, China, USA: IEEE, 2005: 308-319.

[12] ZHAO Haibo and P. DOSHI. Composing nested Web processes using hierarchical semi-Markov decision processes. Proceedings of AAAI, Boston, MA, United states, USA: IEEE, 2006: 75-83,.

[13] HARNEY J, DOSHI P. Risk sensitive value of changed information for selective querying of web services. Proceedings of 8th International Conference on Service Oriented Computing, ICSOC, December 7-10, 2010, San Francisco, CA, United states, United states: Springer Verlag, 2010: 77-91.

[14] CHEN Kun, XU Jiuyun and S. Reiff-Marganiec. Markov-HTN planning approach to enhance flexibility of automatic Web services composition. Proceedings of IEEE International Conference on Web Services, ICWS, Los Angeles, CA, USA: IEEE, 2009: 9-16.

[15] WANG Hongbing, XUAN Zhouy, ZHOU Xiang, Liu Weihong, and Li Wenya. Adaptive and dynamic service composition using Q-learning. Proceedings of International Conference on Tools with Artificial Intelligence, ICTAI, Arras, France, USA: IEEE, 2010:145-152.

[16] WANG Hongbing and GUO Xiaohui. An Adaptive Solution for Web Service Composition. Proceedings of World Congress on Services (SERVICES-1), USA: IEEE, 2010: 503-510,.

[17] LI Lixing, JIN Zhi, LI Ge, ZHENG Liwei, and WEI Qiang. Modeling and Analyzing the Reliability and Cost of Service Composition in the IoT: A Probabilistic Approach. Proceedings of IEEE International Conference on Web Services (ICWS), USA: IEEE, 2012: 584-591.

[18] FAN Xiaoqin, JIANG Changjun, WANG Junli, and PANG Shanchen. Random-QoS-aware reliable web service composition. Ruan Jian Xue Bao/Journal of Software, 20, 2009:546-556.

[19] WU Qing, LI Zenbang, YIN Yuyu, et al. Adaptive Service Selection Method in Mobile Cloud Computing. China Communications, 2012, 9(12): 46-55.

[20] SUN Liang, YANG Dong, QIN Yajuan, et al. Energy-Aware Service Selection Method Based on Sharing Routes in Wireless Sensor Networks. China Communications, 2011, 8(8): 25-33.

[21] SUN Qibo, WANG Wenbin, ZOU Hua, et al. A Service Selection Approach Based on Availability-Aware in Mobile Ad Hoc Networks. China Communications, 2011, 8(1SI): 87-94.

[22] M. ALRIFAI, T. RISSE and W. NEJDL. A hybrid approach for efficient web service composition with end-to-end QoS constraints. ACM Transactions on the Web, United States,2012, 6.

[23] A. R. CASSANDRA. Exact and approximate algorithms for partially observable markov decision processes. Brown University, 1998:447.

[24] H. YANG and S. FONG, Optimizing dynamic supply chain formation in supply mesh using CSET model, Information Systems Frontiers, 2012:1-20,