Journal of Web Engineering, Vol. 13, No.5&6 (2014) 450-468
© Rinton Press

LEARNING-BASED WEB SERVICE COMPOSITION IN UNCERTAIN ENVIRONMENTS

YU LEI[1,2], WANG ZHILI[2], MENG LUOMING[2], QIU XUESONG[2], ZHOU JIANTAO[1]

[1] *Inner Mongolia Engineering Lab of Cloud Computing and Service Software,*

Inner Mongolia University

[2] *State Key Laboratory of Networking and Switching Technology,*

Beijing University of Posts and Telecommunications
yuleiks@bupt.edu.cn

Received July 26, 2013
Revised August 30, 2014

Web service composition has two kinds of uncertain factors, including uncertain invocation results and uncertain quality of services. These uncertain factors affect success rate of service composition. The web service composition problem should be considered as an uncertain planning problem. This paper used Partially Observable Markov Decision Process to deal with the uncertain planning problem for service composition. According to the uncertain model, we propose a fast learning method, which is an uncertainty planning method, to compose web services. The method views invocations of web service as uncertain actions, and views service quality as partially observable variables. The method does not need to know complete information, instead uses an estimated value function to approach a real function and to obtain a composite service. Simulation experiments verify the validity of the algorithm, and the results also show that our method improves the success rate of the service composition and reduces computing time.

Key words: Web service composition; optimal policy; success rate of service composition; partially observable markov decision process; reinforcement learning algorithm
Communicated by: M. Baedke & M. Bieber

1 Introduction

With the development of web service technology, composing services to meet needs of customers has become an inevitable trend. 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 main reasons. For example [1], a web service needs to request many products from markets or suppliers, but it does not know whether markets or suppliers can provide enough products. Therefore the uncertain quantity of product in the suppliers' inventory leads to uncertain execution results of the web service. In another words, different estimations on product quantities of suppliers and markets will lead to different invocation results, i.e., if suppliers are very unlikely to meet the order requirements, the service will first request products from markets. However, if markets are very unlikely to meet the order requirements, the service will first request products from suppliers, which makes the service process different.

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. The last response time of web services cannot be a reasonable criterion, because the load of the system varies over time. Moreover, the average response time of web services cannot be an adaptive criterion in environments emphasizing efficiency. Probability distribution solves our problem when guaranteeing long-term and stable QoS values are difficult, and unstable QoS values affect the success rate of services composition and the quality of composed services.

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.

Therefore, 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 uses Partially Observable Markov Decision Processes (POMDP) [3] to study the problem of web service composition (i.e. view it as an uncertainty planning).

Partially Observable Markov Decision Processes is a decision-making method solving a planning problem, just using partial information of environments due to lack of information.

POMDP is an extension of Markov Decision Processes (MDP). In the MDP model, we make decisions based on actual states of a system, but in many cases, the exact states of the system are difficult to obtain. For example, sensors measuring a complex mechanical system are often affected by noise pollutions, which lead to the difficulty of obtaining exact states of the system. Whereas, POMDP assumes that the information of the system states is just partially knowable and cannot be observed directly, thus it models the system with the incomplete information, and then makes decisions based on the current incomplete information. POMDP has a wide range of application areas [4], such as road maintenance, machinery reparation, robot control, machine vision, mobile target acquisition, target identification, distribution of arms and so on. This paper, which is inspired by reinforcement learning [5], proposes a learning method based on time to compose web services in uncertain environments.

Contributions in this paper:

As far as we know, we are the first to use POMDP to model the problem of service composition. We transfer the model to adapt the service composition in the uncertain environment.

After comparing many 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 introduces an example of supply chain services. The sixth section shows extensive experiment results of the service composition. The final section summarizes the paper.

2 Current Practice and Research

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.

Markov Decision Processes can be defined as a six-tuple: $M=\{S, A, T, R, H, S_0\}$. S is a collection of system states. S_0 is an initial state of the system states. A is a collection of actions. $T: S\times A\rightarrow S$ is a state transition probability. $R: S\times A\rightarrow R$ denotes that performing action A at state S will return a reward R. H is a planning stage. The process of solving MDP decision problems are to find out an optimal policy, $\pi(s)\rightarrow a$. The optimal policy elaborates which action in each state should be taken, so that the largest cumulative rewards will be obtained. Usually, a value iteration method or a policy iteration method is used to obtain the optimal policy.

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.

3 Problem Formulation

3.1 Web service composition modeled by POMDP

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.

3.1.1 Preliminary of POMDP

Partially Observable Markov Decision Processes can be defined as: ***M={S, A, T, R, Z, O, H, B}***, where ***S*** is a collection of system states. ***A*** is a collection of actions. ***T*** is a state transition probability. ***R*** denotes a reward of performing action ***A*** in state ***S***. ***Z*** is a collection of observations. ***O*** is a probability of getting observation ***Z*** when performing action ***A*** in state ***S***. ***H*** is a planning stage. ***B*** is a system's belief. POMDP's goal is to find an approximate optimal policy, which describes that each state should take what action, so that the cumulative rewards are the maximum.

3.1.2 Model parameters

Web service composition can be modeled by POMDP as follows:

S: State, a finite set of states of the world [15]. Web services constitute states. Service composition can be regarded as a task flow from an initial state to a final state, if we select a service successfully, we go to the next service, or we continue to select until successfully. States are denoted as blue rectangles in Fig.1.

A: Action, namely web service invocation. Results of web service invocations are uncertain. Actions are denoted as circles in Fig.1.

T: $\boldsymbol{S}\times\boldsymbol{A}\times\boldsymbol{S}\rightarrow[0,1]$. ***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. $T(s,a,s')=P(s'|s,a)$ denotes the probability that service ***S*** invokes action ***A*** to transit into a new service ***S'***. POMDP imports uncertainty of the invocation results into the planning by the state transition function. State transition function has Markov property, which means that the next state only depends on the current state and the action taken by the current state. Generally, service reliability can serve as transition probabilities, and some researchers use the Bayesian learning methods to obtain the probabilities instead. Transitions are denoted as arrows in Fig.1.

R: $\boldsymbol{S}\times\boldsymbol{A}\times\boldsymbol{S}\rightarrow R^1$, ***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'***. Reward function is an additional factor of POMDP compared with MDP and other classical planning models. POMDP let the planning objective be maximum benefits by means of reward function. The maximum benefits refer to long-term rewards, rather than rewards of simply taking one step. Rewards reflect user preferences for web services, and it could be domain-related preferences (preferences on the different suppliers). The example of rewards is shown in table 1.

Z: Observations that are likely to be observed. They are also a collection of QoS ranks. In the uncertain environments, due to the complex factors such as network load etc., we cannot get comprehensive QoS information at some time, so the QoS information is partially observable. QoS ranks are comprehensive values of all QoS criteria. Five major QoS criteria are execution time, cost, reliability, availability and reputation. 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. Moreover, QoS criteria of web service have the following features: generalization, ambiguity, dynamism and relevance. Unfortunately, these features make normalized QoS uncertain, so we turn normalized QoS values into different QoS ranks. We specify three ranks for each service, and each rank has a possibility. More ranks can be specified according to statistical data if more fine control is needed. For instance, we have six group values of QoS criteria, which can be normalized into six QoS values, say, 1, 2, 1, 2, 3, 2. If three ranks are specified, the first rank is 3 with 1/6 probability, the second rank is 2 with 3/6 probability, and the third rank is 1 with 2/6 probability. Data of table 1 are generated in accordance with this method.

O: $\boldsymbol{S}\times\boldsymbol{A}\times\boldsymbol{Z}\rightarrow[0,1]$. ***O*** is an observation function. It is the probability distribution of observations when we take an action in the current state. $O(a,s',z)=Pr(z|a,s')$ is the probability of the QoS rank ***Z*** when we take action ***A*** to transit to service ***S'***. For example, for one service, we assume that the possibility of being the first rank of QoS is 30%, the possibility of being the second rank of QoS is 70%, and the possibility of being the third rank of QoS is 0%. This probability distribution should be from statistical data. In contrast, in the MDP problem with fully observable variables, ***Z*** is equal to ***S***, which means the state is entirely observable. Therefore, observation function is an additional factor of POMDP compared with MDP model, making POMDP more realistic.

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. ***H*** can be divided into finite steps and infinite steps. Each step has a certain amount of benefits, and later benefits need to be assigned a smaller discount to make the total benefits convergent. This meets the principle of economics that the benefits decrease over time. 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={$b_0, b_1, \dots, b_n$}, 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. But these observations used for describing states are not complete information to make the next decision. In this circumstance, in order to get the current state of the environment, we must maintain a complete sequence of actions and observations of each step to infer state changes. This complete sequence is called history, its form is defined by: $h_t = \{a_0, z_0, a_1, z_1, \dots, a_t, z_t\}$, where t denotes the current time point. However, with passing of time, history will become longer and longer until it is difficult to maintain. Fortunately, the state transition function satisfies the Markov property, which means the last belief state can fully decide the next belief state. Belief states are sufficient statistics of history, therefore, through the belief state we can convert POMDP in the discrete space into belief-MDP in the continuous space, which will be described below.

3.1.3 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. ***Γ:B×A×Z→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***. ***ρ: B×A→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:

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

where 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, $E[\sum_h \gamma^h \times R(s_h, a_h)]$. According to the criterion, we can calculate out what the reward is, when we take an action in a state. That reward is called value function.

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.

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

3.2 Steps solving POMDP

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:

$$b' = P(s'|b,a,z) = \frac{P(s',b,a,z)}{P(b,a,z)} = \frac{P(z|s',b,a)P(s'|b,a)P(b,a)}{P(z|b,a)P(b,a)}$$

$$= \frac{P(z|s',a)P(s'|b,a)}{P(z|b,a)} = \frac{P(z|s',a)\sum_{s \in S} P(s'|s,a) \times b(s)}{P(z|b,a)} \tag{5}$$

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

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.

4 Using Learning Algorithms to Generate Composite Services

4.1 Non-learning methods solving POMDP

POMDP can be divided into a finite stage planning and an infinite stage planning, and a policy generated from the finite stage planning is not steady, because its optimal action may depend on a successive state. This paper uses POMDP with infinite stages. Up to now, several POMDP exact algorithms have been proposed, however the time and space complexity of these exact algorithms are very high. Littman and Cassandra [23] respectively proved that solving POMDP with exact algorithms is difficult, impossible in polynomial time. Therefore, besides some approximate algorithms, we need an effective algorithm to solve the service composition problem modeled by POMDP, and that is reinforcement-learning method.

4.2 Learning methods solving POMDP

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. The biggest advantage of reinforcement learning to solve MDP or POMDP problems is that it can find the optimal policy without knowing the state transition function and the reward function.

Reinforcement learning is the basis of our method. Trial-and-error search and delayed reinforcement are the most important characteristics of reinforcement learning. Reinforcement learning is divided into passive learning and active learning. For passive learning, the policy is known and fixed, whose task is to obtain a state utility (state-action pair), in other words, its aim is to evaluate this policy. For active learning, the policy is unknown, whose task is to take necessary actions to obtain the optimal policy. Active learning is suitable for service composition. Monte Carlo and Temporal difference are common methods of this type.

4.2.1 Monte Carlo method

Monte Carlo method learns the policy by estimating the value function *V(s)*. On the premise of unknown state transition function and reward function, it repeatedly uses formula 7 (an estimated value function) in each learning cycle to progressively approach formula 3 (the real value function).

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{7}$$

In the formula 7, α is a learning rate, a predefined constant. R_t is a reward, $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$, it indicates the learning of Monte Carlo method starts only after the end of a learning cycle.

4.2.2 Temporal Difference method

Temporal Difference method combines Monte Carlo with dynamic planning. Similar to Monte Carlo, it still uses rewards in a complete learning cycle. In the learning process, the iteration method (formula 8) is used to estimate the real value function (formula 3), also without knowing environment model.

The simplest Temporal Difference is one-step forward method, namely TD(0). During iterations of the value function, it simply modifies the estimated value of adjacent states, which is more efficient then Monte Carlo due to Monte Carlo begins learning only after a complete learning cycle. Monte Carlo method uses rewards that obtained after a complete learning cycle to approach the real value function, whereas Temporal Difference uses the value function of the next state and the current reward to approach the real value function. The iterative formula of TD(0) is:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{8}$$

4.2.3 Q-learning method

Q-learning is an advanced Temporal Difference method, which estimates the value function based on actions, $Q(s_t,a_t)$, to learn the optimal policy. The iterative formula (estimated value function) of Q-learning is:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \tag{9}$$

In the belief state space, Q-learning must be appropriately changed, the iteration function is following, by which we obtain the composite service.

$$Q(b_t,a_t) \leftarrow Q(b_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} Q(b_{t+1},a_{t+1}) - Q(b_t,a_t)] \tag{10}$$

When Q-learning 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. The previous literatures use a ε-greedy policy to select the action ($0<\varepsilon<1$). The concept of the method is to greedily select, with ε probability, the retrieved action that has the maximum $Q(b_t,a_t)$ currently, and to explore, with $1-\varepsilon$ probability, new actions. This probability can ensure that each action has opportunities to be explored. Moreover, it can guarantee the actions that have maximum $Q(b_t,a_t)$ have higher priority to be selected. In other words, it accelerates the speed of convergence, and ensures the effective learning.

4.2.4 Time-based Learning method

However, the ε-greedy strategy does not take into account the time factor, i.e. the ε value does not change over time. Inspired by Q-learning, 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 function is:

$$TL(t) = \max_{a_{t+1}} \{\frac{1}{\ln t}\left[Q(b_{t+1},a_{t+1}) - Q(b_t,a_t)\right], \ln t \times Q(b_{t+1},a_{t+1})\} \tag{11}$$

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.
$\alpha$, 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.     $TL(t) = \max_{a_{t+1}}\{\frac{1}{\ln t}\left[Q(b_{t+1},a_{t+1})\text{-}Q(b_t,a_t)\right], \ln t \times Q(b_{t+1},a_{t+1})\}$
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
End Begin
```

In the 6th line, we choose a larger value, namely $TL(t)$, to replace $Q(b_{t+1},a_{t+1})$. 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. When the environment changes, the learning strategies is adapt to the changed environment.

5 Case Studies

5.1 Original problem

Supply chain service is a composite service for retailers to request products. The member of supply chain services (Fig.1) contain a Retailer (R), a Manufacturer1 (M_1, contracted manufacturer), a Manufacturer2 (M_2, non-contracted manufacturer), a Preferred Supplier (PrS), the Other Supplier (OtS), a Spot Market (SpM) and a Delivery carrier (D). In Fig. 1, the manufacturer1 and the manufacturer2 shared the supplier and the spot market, but the manufacturer1 has the preferred supplier. The possible process that the retailer requests products is: the retailer selects an appropriate manufacturer to place orders, if the manufacturer has the preferred supplier, it first queries the inventory of preferred supplier. If the inventory of preferred supplier is empty, the manufacturer queries the other supplier, or directly buys products from the spot market. After choosing the supplier, the retailer needs to select the delivery carrier that transports the products to complete the order processing. During the process, some conditions will affect success rate of service composition. For example, a query may fail because lack of products or bad QoS, then it will go back to the last service to continue querying. Moreover, the spot market is usually able to meet the requirements of orders, but it has more fluctuated costs, so it may not be the first choice.

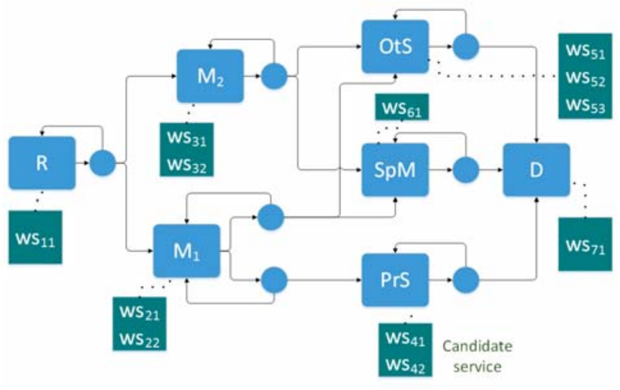

Fig.1. Supply chain services

Vendors involved in the business process use web services to close the deal, and the entire business process is a service that is going to be composed by these component services. As mentioned earlier, the uncertain invocation results and the uncertain QoS values exist in web services. In this case, the uncertainty of the invocation results is that the retailer does not know whether the manufacturer1 or the manufacturer2 is more likely to provide the necessary products. The uncertainty of QoS is reflected in the spot market. For a certain period of time, the market has sufficient and cheap products but slower response time, and for another period of time, the situation is contrary.

In Fig. 1, POMDP's states are denoted by services. POMDP's actions are web service invocations, and the service invocation results are unknown, which may or may not meet the product requirements. POMDP's observations are QoS ranks, which aggregate the execution time, cost, reliability and credibility etc. Each vendor has some candidate services denoted by ws_{ij} in the table below, and we have their statistical data of QoS. According to these data, we can calculate the ranks and probabilities.

O(1)/R denotes the probability of being the first rank of QoS and its reward, etc. O/R denotes that it is a stable QoS value in the MDP case and its reward, which is assumed by other researchers.

TABLE I. Observations, rewards and services

Service	O(1)/R	O(2)/R	O(3)/R	O/R
ws11	0.51/1	0.29/1	0.2/1	1/1
ws21	0.67/6	0.33/2	0/0	1/4
ws22	0.9/7	0.1/1	0/0	1/4
ws31	0.49/8	0.33/2	0.18/2	1/4
ws32	0.38/7	0.22/3	0.4/2	1/4
ws41	0.67/4	0.33/2	0/0	1/3
ws42	0.92/5	0.08/1	0/0	1/3
ws51	0.43/3	0.57/1	0/0	1/2
ws52	0.88/3	0.12/1	0/0	1/2
ws53	0.55/2	0.45/2	0/0	1/2
ws61	0.4/1	0.37/1	0.23/1	1/1
ws71	0.42/1	0.28/1	0.3/1	1/1

5.2 Solutions of the problem

The following Figure is an optimal policy obtained by our TL algorithm.

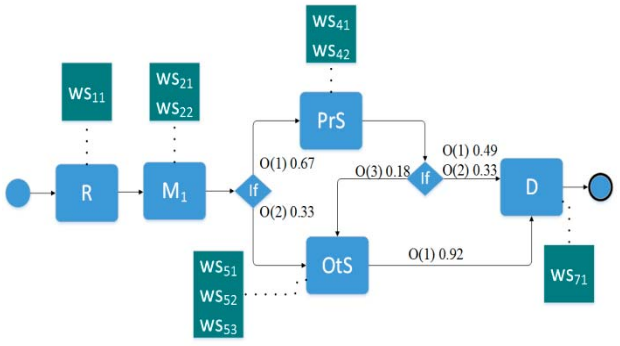

Fig. 2. An optimal policy

Based on this policy, the retailer places orders to the manufacturer1 at first. From a list of candidate services of the manufacturer1, it will select a service with a better QoS rank. Next, the manufacturer1 queries to the preferred supplier and the other supplier. If the QoS of manufacturer1 is the first rank (about 67% opportunities from statistical data), it goes to the preferred supplier, and then the preferred supplier requests to the delivery carrier. If the QoS of manufacturer1 is the second rank (about 33% opportunities), the manufacturer1 queries the other supplier. There are two possible situations that the preferred supplier requests to the delivery carrier. In the first one, where the observation of QoS is the first rank (around 49% opportunities) or the second rank (about 33% opportunities), either of them meets the QoS requirements, so the preferred supplier places orders to the delivery carrier. In the second one, the observation of QoS is the third rank (about 18% opportunities), then it turns to the other supplier. Finally the other supplier places orders to the delivery carrier, the observation of this invocation is QoS the first rank (about 92% opportunities). ws11, ws22, ws42, ws52, ws71 are selected by TL method as the best component services for the composite service in this case.

Three QoS ranks of the manufacturer1 constitute a probability distribution, and the sum of the probability of three QoS ranks is 100%. If the QoS-ranks probability distribution of the manufacturer1 does not always intend to be high QoS rank, and the QoS-ranks probability distribution of other vendors intends to be Normal Distribution, then the optimal policy may become more complex, as the following Figure.

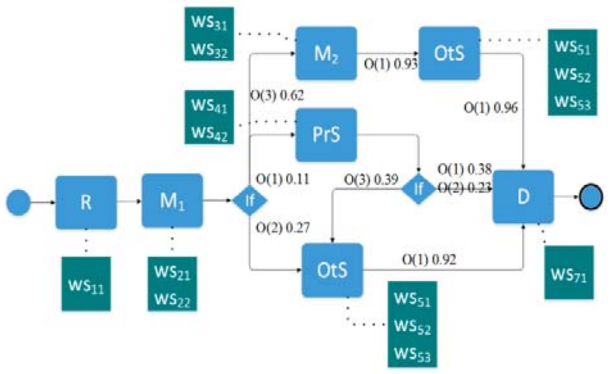

Fig. 3. A changed optimal policy

Based on the optimal policy, the optimal plan for the service composition is: the retailer places orders to the manufacturer1, and then the manufacturer1 queries to the manufacturer2, the preferred supplier and other suppliers. If the QoS of the manufacturer1 is the first rank (about 11% opportunities), it goes to the preferred supplier, and then the preferred supplier requests to the delivery carrier. If the QoS of manufacturer1 is the second rank (around 27% opportunities), it goes to the other supplier, and then the other supplier requests to the delivery carrier. Thereafter, the remaining branch of the policy is the same with Fig. 2. The difference is the time when the QoS of the manufacturer1 is the third rank (around 62% opportunities), this time it cancels the orders for the manufacturer1, and places the orders to manufacturer2 instead. When the QoS of the manufacturer2 is the first rank (around 93% opportunities. We ignore other probabilities here for simplifications), then it goes to the manufacturer2 who will query the inventory of the other supplier in the next step. When the QoS of the other supplier is the first rank (around 96% opportunities), then it turns to the other supplier. Finally the other supplier places orders to the delivery carrier. TL method selects ws11, ws22, ws31, ws42, ws52, ws71 as the best component services for the composite service in this case.

6 Simulation Results

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. We assume each service has four candidate services in the Fig.4 and Fig.5. The two figures show that Q-learning and TD(0) are more efficient than Monte Carlo method, the reason is that the value function of TD(0) in the iteration only modifies values of adjacent states. Moreover, Monte Carlo method begins to iterate after a whole learning cycle, whereas TD(0) and Q-learning are only need to search one step forward. Q-learning is slightly better than TD (0), because Q-learning uses a ε-greedy strategy to assign a higher priority to 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.

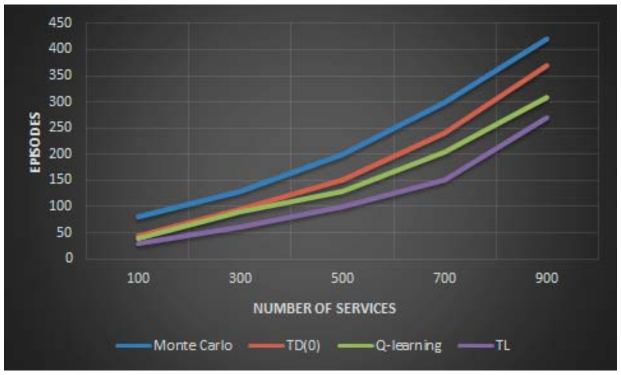

Fig. 4. Comparison of learning speed w.r.t. services

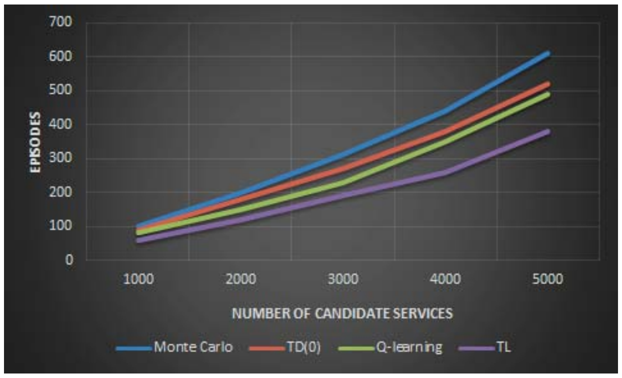

Fig. 5. Comparison of learning speed w.r.t. candidate services

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. 6 shows that our success rate of service composition in uncertain environments is higher than that obtained by [18].The reason is that, general methods based on MDP simply use the service reliability for the service transition probabilities and do not consider partially observable QoS variables, whereas TL method maps QoS ranks to partially observable variables, which is more suitable for uncertain service environments.

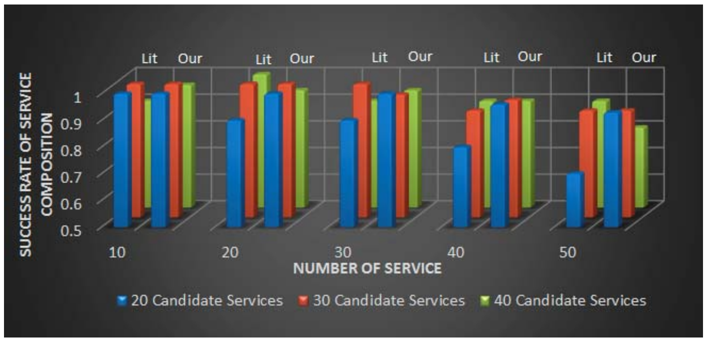

Fig. 6. Comparison of success rate of service composition

Fig. 7 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. However, TL method has lower time complexity and shorter computing time.

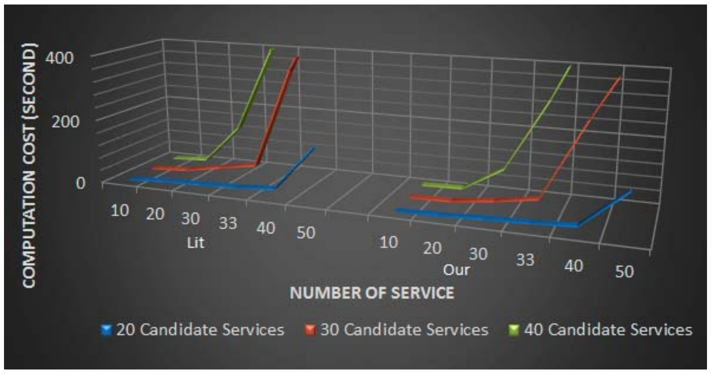

Fig. 7. Comparison of computing time

Above experiments show that more parameters make the model more realistic, TL method improves success rate of the service composition and reduces computing time.

Our method can adjust learning speed over time to accelerate convergence. 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. We randomly generated transition graphs, whose number of services is between 1000 and 4000, and number of candidate services is between 1000 and 10000. The effect of service composition using TL method is shown in Fig. 8. Comparing our results with the results of literature [15], we observed that the cumulative reward is converged to the optimal policy very fast. From Fig. 8 and Fig. 9, we can see that our method is faster than that of literature [15]. The convergence time increases polynomially with the number of services and candidate services.

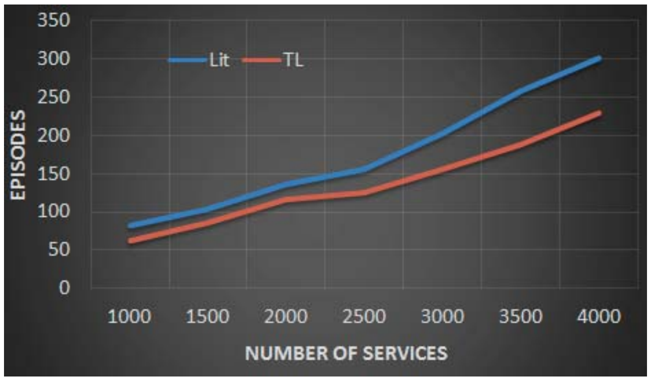

Fig. 8. Convergence time with respect to services

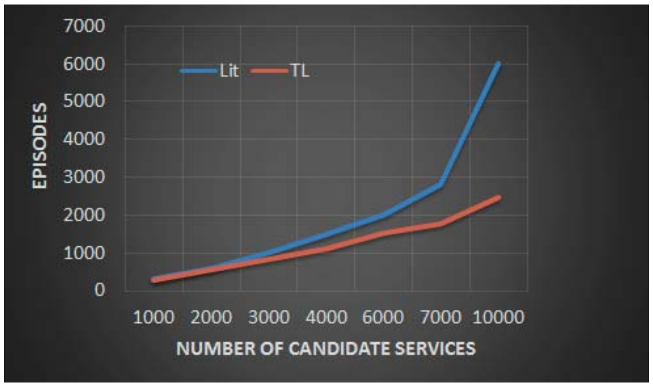

Fig. 9. Convergence time with respect to candidate services

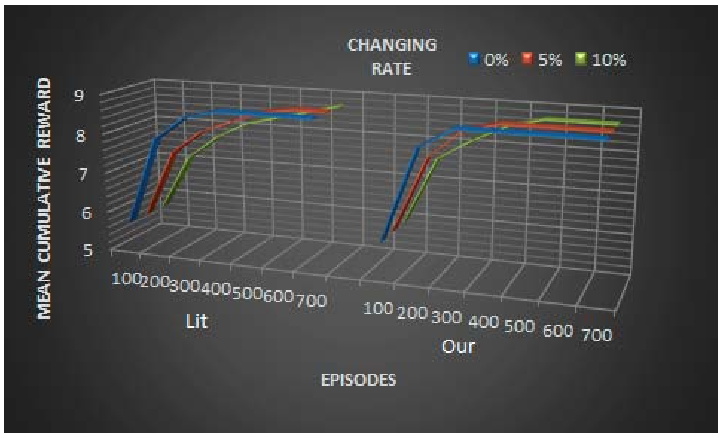

Fig. 10. Influence of changes of QoS on learning

In the dynamic network environments, the average values of QoS criteria may not always remain the same. Therefore, we regularly change the QoS values with a certain probability in the experiment. In Fig. 10, we respectively change 5%, 10% of the QoS criteria values in every 100 learning cycle, comparing to the stable QoS values (the changing rate is 0%). *Lit* is the result of literature [15] in Fig. 10, *TL* is the result of our method. Results show that *TL*'s convergent speed is faster in three cases. Fig. 10 also indicates that even though the change of QoS values increases the computing time that find the optimal solution, it does not stop the algorithms to find the optimal solution.

In the dynamic network environments, the number of web services may increase because of provisions of new services, and may reduce because of failures of existing services. In another experiments, we change the quantity of services by a certain probability. There are three kinds of circumstances, 0%, + 2% and - 2%, which means the quantity of services remains the same, the quantity increases by 2% and the quantity reduces by 2%. Below is the comparison result, it shows that the decreasing the quantity of services reduces the computing time, and increasing the quantity of services raises the computing time.

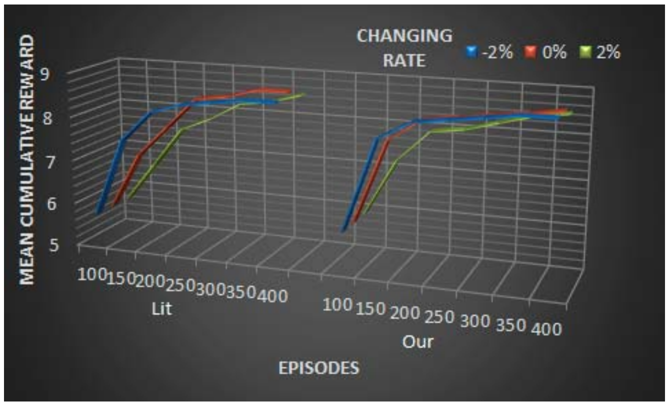

Fig. 11. Influence of changes of services on learning

7 Conclusions

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 whose advantage is no need to know complete information. It views unpredictable invocation results of Web services as uncertainty of actions, and views unstable QoS as uncertainty of partially observable variables. The proposed method is applied to a real scenario, a composition service for supply chain. Comparing to the methods based on MDP, the experiments show that our method achieves higher success rate of service composition and shorter computing time.

Acknowledgements

This work was supported by National Natural Science Foundation of China [No: 61262082], Key Project of Chinese Ministry of Education [No.212025], Inner Mongolia Science Foundation for Distinguished Young Scholars [2012JQ03], Creative Research Groups of China (No. 61121061); National High-tech R&D Program of China (No. 2013AA013502); National Key Technology R&D Program (2012BAH06B02).

References

[1] P. DOSHI, R. GOODWIN, R. AKKIRAJU, and K. VERMA. Dynamic workflow composition using Markov decision processes[J]. 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[J]. Computer Communications, 2012, 35(15): 1882-1892.

[3] C. RAPHAEL and G. SHANI. The Skyline algorithm for POMDP value function pruning[J]. 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[M]. 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[J]. 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] // 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 [C] //

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[J]. 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[J]. 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[J]. 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[J]. 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[J]. ACM Transactions on the Web, United States,2012, 6.

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

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