Markov-HTN Planning Approach to Enhance Flexibility of Automatic Web Services Composition

Kun Chen, Jiuyun Xu
School of Computer & Communication Engineering
China University of Petroleum
China, 257061
ck_star@126.com;
xujy@mail.hdpu.edu.cn

Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
University Road,
Leicester, LE1 7RH
United Kingdom
srm13@le.ac.uk

Abstract

Automatic Web services composition can be achieved by using AI planning techniques. HTN planning has been adopted to handle the OWL-S Web service composition problem. However, existing composition methods based on HTN planning have not considered the choice of decompositions available to a problem which can lead to a variety of valid solutions. In this paper, we propose a model of combining a Markov decision process model and HTN planning to address Web services composition. In the model, HTN planning is enhanced to decompose a task in multiple ways and hence be able to find more than one plan, taking both functional and non-functional properties into account. Furthermore, an evaluation method to choose the optimal plan and some experimental results illustrate that the proposed approach works effectively.

1. Introduction

Web services are defined as software systems which are described with functional and non-functional capabilities and can enable improved coordination among multiple computing platforms, applications, and business partners. Because a single Web service usually can not fulfil the requirements of users, Web service composition provides a mechanism to combine different services together to handle business process. Automated Web service composition is valuable in many domains, typical of e-commerce. However, with the rapid increase of Web services and more complex requirement of business process in the real world, automatic service composition requires a more flexible mechanism to deal with unexpected exceptions. AI planning for automated Web services composition has been adopted, as exemplified by the methods presented in [1-4] to handle this issue.

In [4], an HTN planning method has been suggested to handle automatic Web services composition. This method translates OWL-S Web service descriptions to a SHOP2 domain and then a business plan is achieved by decomposing complex tasks. Considering the procedure of task decomposition, this method mainly is concerned with the feasibility of task decomposition; that is can one plan be found? However, a plan may fail for various reasons, such as a service instance no longer exists or feature interaction in Web services [5]. In the real world, there usually are several possible plans which can solve one specific high-level business process. For instance, a user wants to attend an exhibition in another city in a few days. On the condition of satisfying user's requirements, he can make a choice of taking a flight or a train to the city and then attend the exhibition. In this situation, the user always wants to know what options he has and which is of the best quality (that is satisfying his non-functional criteria such as cost considerations).

This paper addresses the aspect of finding multiple composition plans and then selecting the most appropriate for a user. We propose an enhanced approach for Web services composition based on the combination of HTN planning and a Markov decision process model. With this approach, several highly suitable Web service plans will be obtained providing different solutions to a business process using Web services composition and hence offering a much more

flexible solution to the customer. To make sure these plans are indeed some of the best solutions available we use an evaluation mechanism to illustrate the optimal solution amongst those multiple solutions using a Markovian decision process. In this way, the optimal solution not only meets the requirements of the business process in its functional aspects, but also satisfied the expectations that the solution is of the best quality based on requirements considering non-functional aspects.

The rest of the paper is organized as follows: in section 2, an overview of Web services composition using HTN planning is introduced. section 3 gives an overview of the composition model and section 4 details the process of model solving. In section 5 a case study is introduced and experimental results are presented. Finally, we conclude and provide an outline of further research.

2. An Overview of Web services Composition using HTN Planning

HTN (Hierarchical Task Network) is a technique of AI planning based on control knowledge with a closed world assumption (informally, that means that all "building blocks" are known a-priori). HTN planning provides hierarchical abstraction with a powerful strategy to deal with the complexity of large and complicated real world planning domains. The purpose of an HTN planner is to produce a sequence of actions that perform some activity or task.

Considering composing Web services using HTN planning, generally the planning domain, planning problem and the process of planning are to be described within the Web service domain. The description of a planning domain includes a set of operators (which will be web service operations), and also a set of methods, each of which is a prescription for how to decompose a task into its subtasks (smaller tasks). The description of a planning problem will contain an initial state which is the same as that of classical planning, but instead of a goal formula, the problem specification will contain a partially ordered set of tasks to accomplish. The process of HTN planning proceeds by using the methods to decompose tasks recursively into smaller and smaller subtasks, until the planner reaches primitive tasks that can be performed directly using the planning operators. For each non-primitive task, the planner chooses an applicable method, instantiates it to decompose the task into subtasks, and then chooses and instantiates methods to decompose the subtasks even further. When the constraints on the subtasks or the interactions among them prevent the plan from being feasible, the planning system will backtrack and try alternative methods. More details discussed on HTN planning found in [6].

The OWL-services language (OWL-S), is a set of ontologies for describing the properties and capabilities of Web services. Currently, Web services mostly are described by OWL-S since it supports effective automation of various Web services related activities including service discovery, composition, execution, and monitoring (it provides a richer framework than WSDL). Especially, the structure of OWL-S is propitious to exploit AI planning techniques for automatic service composition by treating service composition as a planning problem. In OWL-S, services can be described as composite or atomic processes with preconditions and effects. The concept of composite process decomposition in OWL-S process ontology is very similar to the concept of task decomposition in HTN planning. Hierarchical modelling is the core of the OWL-S process model to the point where the OWL-S process model constructs can be directly mapped to HTN methods and operators. Thus, HTN planning is especially promising for OWL-S Web services composition, which has been shown in [4, 7, 8].

3. The Formal Model of Markov-HTN planning for Web services composition

So far, many existing Web services composition models based on HTN planning are not powerful enough to support the idea that consider the functional aspects together with the non-functional, and also find multiple optimal plans in the planning process. Therefore, we put forward a formal model extended on the basis of composition model used by SHOP2 in [4]. The formal approach of Markov-HTN planning is able to support a choice of Web services composition plans, and considers the non-functional aspects of Web services, which enhances the flexibility of automatic Web services composition. The definition of the formal model is as follows.

Definition 1 (Markov-HTN planning Model for Web Services Composition).

The OWL-S Web services composition problem is defined as $< S, T, D, Q, P >$. In the 5-tuple model,

- S is the initial state of the problem.
- T is the task list, which contains the tasks that the user needs to solve.
- D is the description of a planning domain includes a set of operators and a set of decomposition methods, and D can be translated

from a collection of OWL-S process models.

- Q is a set of QoS vectors, the attributes of which include the response time, cost, availability and reliability.
- P is a set of optimal solutions which are available in the solution space.

On the basis of the above definition, solving the 5-tuple can return an optimal plan $P_{optimal} = (O_1 O_2 ... O_n)$, that is, a sequence of instantiated operators that will achieve T from S in D, and an optimal combination sequence with the best quality with respect to the non-functional aspects.

The model solving consists of three processes. First, the initialization of the description of the planning domain; second, the search for the best plans in the solution space based on HTN planning and thirdly, the evaluation of the optimality in the availability plans.

4. The Approach of Web service composition using Markov-HTN Model

4.1. Initialization for the description of the planning domain

The 5-tuple model for OWL-S services composition is based on HTN planning. So, the beginning to do is translating the description of OWL-S services to a description of planning domain.

Let $K = \{K_1, K_2, \ldots, K_m\}$ be a collection of OWL-S process models. Then, Let $D = TRANSLATE - PROCESS - MODEL(K)$. This process is achieved by using the translating algorithm provided in [4]. Details of the translation and assumptions the translation based on are all kept unchanged.

After the completion of this process, the element D in the 5-tuple model is described by a set of operators and a set of decomposition methods. Each operator is a description of what needs to be done to accomplish some primitive task, and each method tells how to decompose some compound task into a set of partially ordered subtasks. The control knowledge base for HTN planning consists of operators and methods. The definition of operators and methods are the same with its description in SHOP2 domain description [4,6], as follows:

Definition 2 (Operator). An operator is an expression of the form (h(v→) Pre Del Add) where

- h(v→) is a primitive task with a list of input parameters v→.
- Pre represents the operator's preconditions.
- Del represents the operator's delete list which is described as a conjunction of logical atoms that will become false after operator's execution.
- Add represents the operator's add list which is described as a conjunction of logical atoms that will become true after operator's execution.

Definition 3 (Method). A method is an expression of the form (h(v→) Pre_1 T_1 Pre_2 T_2 …) where

- h(v→) is a compound task with a list of input parameters (v→).
- Each Pre_i is a precondition expression.
- Each T_i is a partially ordered set of subtasks.

In the process of model solving, the part that initializes the description of the planning domain does not always need be done. The description of the planning domain, element D in the 5-tuple model, needs to be updated synchronously, only when the collection of OWL-S process models is changed.

4.2. Planning with Multi-decomposition for tasks

In this paper, the process of HTN planning is improved in the second step, that is searching for plans, in order to be able to produce more than one good solution within the available solution space. Specific details of the improvement focus on decomposition for non-primitive tasks when a task can be decomposed by more than one method.

The improved decomposition method changes the way of decomposing when a task can be decomposed by multiple methods. The method chooses *each* method to decompose a non-primitive task instead of choosing *any one* of the ones applicable in the current state. Also, a control strategy is embedded into the planning process to decide whether a branch will be decomposed further. The detail of improved non-primitive tasks decomposition is presented in Fig. 4.1.

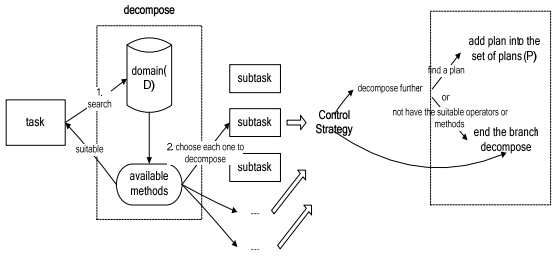

Fig. 4.1 Decomposing a non-primitive task

The improved decomposition is superior to the decomposition presented in [4] on the strategy of searching for solutions. For decomposing a non-primitive task with every available method, the current state (S) and task list (T) must be copied, and the number of the replications is the same as the number of available methods. After this, every branch can be considered by the planning method. If one branch cannot be decomposed further, that is all the subtasks are primitive tasks, the found plan will be added to the set of plans (P). In the subsequent recursive process, a similar situation that a subtask may have more than one available method to be decomposed will occur. With the number of such situations increasing, the solution space that will be searched is growing and the planning process will be more and more complex. So, we apply a control strategy to decide whether a branch will be decomposed further.

Before the definition of the control strategy, the concept of immediate reward needs to be introduced.

Immediate reward: An immediate reward is a utility value to measure the quality of a decomposition method. A method decomposes a task into primitive subtasks or non- primitive subtasks. A primitive task can be performed directly using a service operation (or planning operator in planning terms). Clearly, operations suggested by a decomposition method have a direct impact on the overall quality of the solution. On the basis of this, the immediate reward of a decomposition method can be calculated by using the QoS (Q), and the corresponding Web services are mapped into operators produced on the certainty branch, which does not have a subtask that can be decomposed by more than one method in the remaining decomposition process until planning is completed.

Since the construction of a plan is a sequence of operators, the immediate reward can be defined by formula (1), which is similar to [9, 10].

$$R = \frac{1}{N} \frac{w_3 \prod_{j=1}^{N} \text{Availability}(ws_j) + w_4 \prod_{j=1}^{N} \text{Reliability}(ws_j)}{w_1 \sum_{j=1}^{N} \text{Cost}(ws_j) + w_2 \sum_{j=1}^{N} \text{Response Time}(ws_j)} \quad (1)$$

where N is the number of Web services which are mapped into the operators produced on the certainty branch. $w_1, \ldots, w_4$ indicate the importance a service integrator (or user) gives to a particular QoS attribute. In formula (1), there is a special case that some free and fast services will lead an infinity value of R. Thus, the effect of availability and reliability are neglected. This case often occurs on the bottom tasks decomposition when the services less on certainty branch are all free and fast. So, there is a little effect on the whole plan, and which can be taken no account in a whole plan evaluation mentioned in section 4.3.

Definition 4 (Control Strategy). There is threshold value $\lambda(\lambda \geq 0)$, which is a standard to measure the immediate reward value R of a decomposition method m. If $R_m \geq \lambda$, the planner uses the method to decompose further, else if $R_m < \lambda$, the planner stops to decompose this branch. For $\lambda = 0$, all the branches will be extended.

Figure 4.2, shows a search tree for a planning problem. In node 2, the branch will not be extended by decomposition method *m23*, because $R_{m23} < \lambda$. The same applies to the branch to be extended by decomposition method *m22* in node 4.

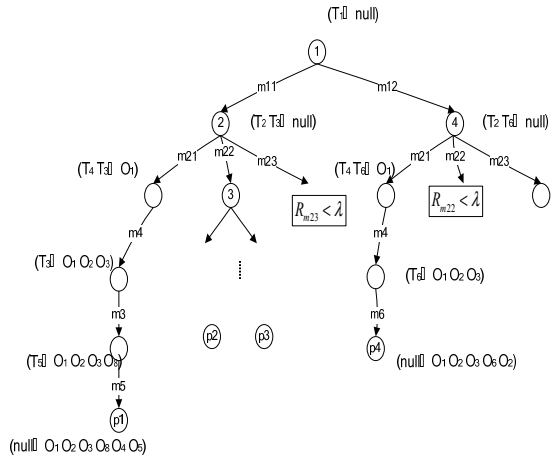

Fig. 4.2 A search tree for a planning problem

Because of the control strategy, we cannot only make it easier to reduce the size of the solution space that is searched, but can also find better solutions addressing different demands of users. To some extent, the speed to find plans and the number of plans that will be found can be controlled by changing the threshold value λ. Since the immediate reward value R measures the quality of a decomposition method, it can also be used to evaluate the quality of plans. More details will be introduced in section 4.3.

As Fig. 4.2 shows, a node $n = (T, \pi)$ in the search tree implies the current state. T is the task list which needs to be solved and π is the current partial plan. In the planning process, the state at each node can be reached by the initial state S and the current partial plan π. The node that has any child in the search tree is called a terminal node. If the terminal node has

a null task list, the corresponding π is a feasible solution for initial tasks of users and which is composed of operators.

On the basis of the given description, the algorithm for HTN planning used is as shown in Fig. 4.3.

```
Procedure CompleteDecomposition_HTN
Input: S,T,D,Q,P, plan(a sequence of Operators).
Output: true or false

1. if T != Ø
2. get the first task t from T
3. if t is a non-primitive task
       Find all the available methods M in D.
       M ← {(m,θ): m is an instance of a Method in D, as m = (h, Pre, T)
            θ unifies {h(m),t}, Pre(m) is true in S.}
4.     if M = Ø, return false.
       else
5.         make N copies of current < S,T >, N is the number of (m,θ) in M.
6.         choose a pair(m,θ) ∈ M to decompose based on copies of < S,T >.
           Loop N:
7.         compute the short reward value R of m by Q.
8.         if R ≥ λ, modify T by removing t, adding all the elements in T(m).
9.         else return false.
10. else if t is a primitive task
       Find an Operator o = (h Pre Del Add) in D,
       such that h unifies with t and S satisfies Pre.
11.    if no such o exists, return false.
       else
12.        modify S by deleting Del and adding Add.
13.        modify T by removing t.
14.        appending the Operator o to plan.
15. if T = Ø
       modify P by adding plan, and then reset plan = null.
16.    return true.
17. else
       return CompleteDecomposition_HTN(S,T,D,Q,P,plan).
```

Fig. 4.3. HTN planning algorithm for complete decomposition

Note that all branches in the planning process will be considered when a task can be decomposed by multiple decomposition methods. At that state, the immediate reward of the decomposition methods is calculated which determines whether a branch will be extended further. It may be possible that branches which could lead to better utility in further process will be cut away, but that has little consequence to the better quality plans found. In view of the reliability of plans during actual execution, the partial plan composed of the operators which are found on an anterior branch is more important than the one found on the posterior branch. Consequently the plans produced by the HTN planning algorithm for complete decomposition are ensuring better quality. Moreover, the solution space searched for large-scale and complex problems can be controlled by adjusting the threshold λ of the control strategy.

4.3. Optimality Evaluation by MDP

After the completion of the HTN planning step, several good plans can be provided to users, but it is the optimal plan that users are most concerned about. Hence, we propose a method to evaluate the optimality using a Markov decision process (MDP) is proposed. MDPs provide a mathematical framework for modelling decision-making in situations where outcomes are partly random and partly under the control of the decision maker. MDPs are useful for solving a wide range of optimization problems.

In the process of HTN planning, the choice of multiple decomposition methods can be seen as a decision-making process and the decision-making only connects with the current state. So we construct a MDP model by introducing the probability and reward value for choosing a decomposition method and solve the model to find the optimal plan. The time to choose a method is a decision-making time *t*, such as the nodes $(1,2,3,4)$ in Fig. 4.2. First, a list of four objects in MDP should be described as $(S, A, P_a(\cdot,\cdot), R_a(\cdot,\cdot))$, where,

- S is the state space.
- A is the available action set, which is the same with the available decomposition methods set.
- $P_a(s,s')$ is the probability that action *a* in state *s* at time *t* will lead to state *s*' at time *t* + 1.
- $R_a(s,s')$ is the immediate reward received after transition to state *s*' from state *s*.

Calculation of transition probability and reward. In the MDP process, the calculation of the transition probability and reward is the core. The probability for choosing a decomposition method in HTN planning is related to the preconditions of the method. Fewer constraints of preconditions will have less risk of failure in actual execution process. Hence, a less restrictive method has a higher probability of being selected.

In state *s*, a task can be decomposed by *k* methods *M*. Each method $m_i(1<i<k)$ in *M* has N_i parameters in its Pre_i. Then, the transition probability is defined by formula (2).

$$P_a(s|s') = P_{m_i}(s|s') = \frac{\theta(\sum_{j=1}^{k} N_j - N_i)}{\sum_{j=1}^{k} N_j}, \text{and } \theta = \begin{cases} 1, \text{if } Pre_i \subset s \\ 0, \text{otherwise} \end{cases} \quad (2)$$

The reward calculation is the same with the immediate reward introduced in section 4.2, also defined by formula (1). QoS publication helps selecting among services with the same functionality,

service composition based on QoS and evaluation of alternative execution paths for process adaptation. Moreover, QoS can be used as a basis for cost models that drive process optimization[11].

Solution to MDP by the way of policy iteration. The solution to a Markov Decision Process can be expressed as a policy π, a function from states to actions. The standard family of algorithms to calculate the policy need calculate two variables repeatedly. One is value *V*, which contains utility value of state *s*, and the other is policy π which contains actions *a*. *s'* is the next state achieved by executing an action *a* from the current state *s*. The two variables are calculated by formula (3), (4).

$$V(s) = R(s) + \gamma \sum_{s'} P_{\pi(s)}(s,s')V(s') \quad (3)$$

, where γ is a discounting factor.

$$\pi(s) = \arg\max_{a} \sum_{s'} P_a(s,s')V(s') \quad (4)$$

After completing the second process of HTN planning, the plan set P has *N* plans. So, policies, available actions A and state space S in MDP can be determined. Make each plan to be a policy, such as in fig4.2, the plan *p1* can be expressed as a policy $\pi_1 : \{(s_1, m11), (s_2, m21)\}$, and the expected utility of a policy reflects the quality of the plan, and which can be calculated by formula (5).

$$E_{\pi_i}(s) = R(s) + \gamma \sum_{s'} P_{\pi_i(s)}(s,s')E(s') \quad (5)$$

, where *s* is the state in policy π_i . Formula (5) calculates all the rewards on non-primitive tasks decomposition during the production of a whole plan. Considering the effect of each layered decomposition, the high layers is more impact for the plan than the low ones.

Then, the policy iteration algorithm is used to find the optimal policy. Specific details are described as shown in Fig. 4.4.

Policy iteration:

1. *start with an arbitraty initial policy* π
   for $i = 1, 2, \ldots$
2. *compute* $E_{\pi_i}(s)$ *for every s*,
   $\Rightarrow E_{\pi_i}(s) = R(s, \pi_i(s)) + \gamma \sum_{s' \in S} P_{\pi_i(s)}(s,s')E_{\pi_i}(s')$
3. *for every s*,
   $\Rightarrow \pi_{i+1}(s) := \arg\max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} P_a(s,s')E_{\pi_i}(s')$
4. *if* $\pi_{i+1} = \pi_i$, *then exit*. π_{i+1} *is the optimal policy*.

Fig 4.4. Policy iteration algorithm for MDP

The process will converge in a finite number of iterations and the process ends with the optimal policy (For a proof, we refer to [12]).

5. A Case Study

To demonstrate the feasibility of our composition approach, the commonly used e-travel scenario will be used. The e-travel scenario requires that first a destination is reached and then after a few days touring an exhibition is attended. Users may have more than one plan to achieve the goal. We implement a system by using our approach of Web services composition based on the Markov-HTN planning model to simulate the composition. We can get composition plans as shown in Fig. 5.1.

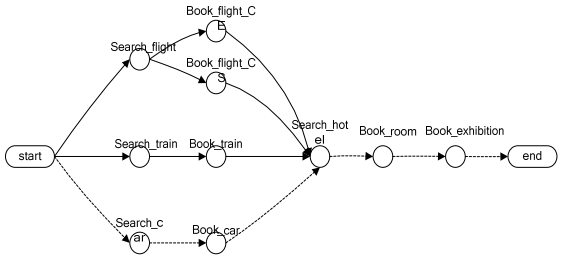

Fig. 5.1. a sample of plans for e-travel

As Fig. 5.1 shows, there are 4 plans for composing Web services in sequence; these can be found by using our improved decomposition HTN algorithm. In contrast, by using the SHOP2 algorithm, only the first plan can be found and this might not be the users' preferred option. After giving all the available plans that can fulfil the task, users can choose the one that they expect to execute, and also we use the MDP method to identify the optimal plan based on non-functional aspects. In Fig. 5.1 the optimal plan is plan_4 (marked in broken line).

MDP is an efficient method to solve an optimization problem, like the decision of choosing a decomposition method in HTN planning. In order to demonstrate the feasibility and validity for using MDP to solve the optimal plan, we experimented in our system by using 10 groups of random QoS data set to identify an optimal plan based on the structure of plans shown in Fig. 5.1 shows. In Fig. 5.2, the abscissa *X* indicates the labelling of the QoS data set, and the ordinate *Y* indicates the expected utility value of a plan. So, the point (X, Y) with different shape indicates that a plan represented by the point shape has an expected utility value *Y* calculated using QoS data set *X*. The point on the line is the optimal plan under current QoS data set. Our results show that the optimal

plan is almost always the plan with the highest expected utility value, as Fig. 5.2 shows.

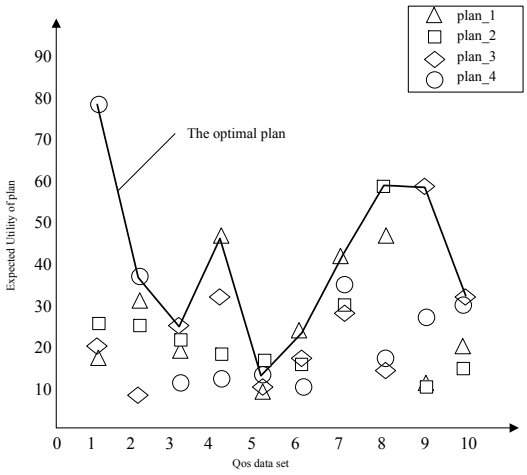

Fig. 5.2. The expected utility of plans and the optimal plan

It should also be noted that tasks with different complexity have different size of solution space. Users expect that sufficient and better plans can be provided, but if the task is so complex that the search time is too long and the plans have too much redundancy, we can reduce the search space by increasing the threshold λ. The value of λ is inversely proportional to the search time and the number of plans found.

6. Related Work

Considering the related work, Zhang jianhong et al.[13] proposed an enhanced HTN planning method combined with partial-order planning(POP) for service composition in which action decomposition are used as plan refinements in POP. Comparing to the pure HTN planning, this approach can solve certain tasks, which are novel conjunctive goals. In our approach, we also focus on the decomposition in HTN planning, improving the decomposition for non-primitive tasks, but in order to search more potential feasible solutions.

Evren Sirin and Dana Nau et al.[7] presented a HTN planning algorithm, ENQUIRER, designed for planning domains and in which the information about the initial state of the world may not be complete. By using ENQUIRER, information is discoverable through plan-time information gathering queries. In ENQUIRER, some limitations in their previous work [4] are overcome, which can make service composition sound and complete. Based on the work in [4], our approach improves the composition method to provide multiple plans and consider the non-functional properties of Web services in planning process in addition for user's flexible choice.

Incheon Paik and Daisuke Maruyama[2] suggested a combined architecture, which consisted of HTN planning and Constraint Satisfaction Problem(CSP) as an underlying problem-solving engine to automate Web service composition and especially for composition problem with scheduling with many parameters. In the architecture, a complete semantic concept for CSP is provided by using OWL, which can make solver agents automatically solve a given problem with greater flexibility and intelligently. This work focuses on the CSP for the semantic web, the CSP solver is a part of the combined architecture but independent of HTN planning. The CSP solver and solver agents solve the problem collaboratively. In this method, MDP for evaluating an optimal plan is independent of HTN planning as well. But in our work, MDP is used to select a whole plan based on multiple plans. Likewise, it can be used on single Web Service selection in other works [14].

Prashant Doshi, Richard Goodwin et al.[15] modeled the workflow composition problem as a MDP, which handled non-deterministic behaviors of Web services in dynamic environments with the phrase of the plan execution. In that paper, a policy computed by MDP for generating workflows is capable of optimally recovering from Web service failures. While our work is about services composition based on AI planning during decomposing into the atomic tasks, in which MDP is used to evaluate an optimal plan among multiple available plans considering the non-deterministic of non-primitive tasks decomposition in HTN. It is different from work in [15].

7. Conclusion and Future Work

In this paper, a novel composition model based on Markov-HTN planning has been proposed. With this model, more than one plan can be found and the evaluation mechanism in model can give an optimal plan based on non-functional aspects.

With a choice of Web services composition plans, users can be more flexible in accomplishing their tasks in the most suitable way. They can adopt the optimal plan that our method provides, but they can also choose freely according to their own preference from a number of alternatives. Moreover, when executing the selected plan results in failure, candidate plans can ensure the tasks will be completed without constraints slacking or premises increasing.

While our method can provide multiple plans for users, we will explore a re-planning mechanism to be used when plan execution results in failure. Under this mechanism, a process of plan execution can be continued automatically from an appropriate service node and the negative impact of a failure will be minimized.

Acknowledgment Thanks to the anonymous reviewers for the comments

Reference

[1] Hilmar Schuschel,M. Weske, "Automated Planning in a Service-Oriented Architecture," Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004.

[2] Incheon Paik, Daisuke Maruyama, M. N. Huhns, "A Framework for Intelligent Web Services: Combined HTN and CSP Approach," IEEE International Conference on Web Services (ICWS'06), 2006.

[3] Rama Akkiraju, Biplav Srivastava, Anca-Andreea Ivan, Richard Goodwin,T. Syeda-Mahmood, "SEMAPLAN: Combining Planning with Semantic Matching to Achieve Web Service Composition," IEEE International Conference on Web Services (ICWS'06), 2006.

[4] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, D. Nau, "HTN planning for Web Service composition using SHOP2," Web Semantics: Science, Services and Agents on the World Wide Web 2004, pp. 377-396.

[5] M. Weiss, B. Esfandiari, Y. Luo, "Towards a classification of web service feature interactions," Computer Networks, vol.51, 2007, pp.359-381.

[6] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu,F. Yaman, "SHOP2：An HTN planning system," Journal of Artfficial Intelligence Research, vol. 20, 2003, pp. 379-404.

[7] Ugur Kuter, Evren Sirin, Bijan Parsia, Dana Nau,J. Hendler, "Information gathering during planning for Web Service composition," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 3, 2005, pp. 183-205.

[8] Naiwen Lin, Ugur Kuter, James Hendler, "Web Service Composition via Problem Decomposition Across Multiple Ontologies," 2007 IEEE Congress on Services (SERVICES 2007), 2007.

[9] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, M. L. Villani, "An Approach for QoS-aware Service Composition based on Genetic Algorithms," Genetic and Evolutionary Computation Conference(ACM), 2005.

[10] LIU Shu-Lei, LIU Yun-Xiang, ZHANG Fan, TANG Gui-Fen, JING Ning, "A Dynamic Web Services Selection Algorithm with QoS Global Optimal in Web Services Composition," Journal of Software, vol. 18, 2007, pp. 648-656.

[11] Diego Zuquim Guimarães Garcia, M. B. F. d. Toledo, "Semantics-enriched QoS policies for web service interactions," Proceedings of the 12th Brazilian symposium on Multimedia and the web(ACM), 2006, pp. 35-44.

[12] Liu Ke, "Applied Markov Decision Process," Beijing:Tsinghua University publication, 2004, pp. 38-40.

[13] Zhang Jianhong, Zhang Shensheng ,M. Y. Cao Jian, "Improved HTN Planning Approach for Service Composition," *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04)* 2004.

[14] Dongjun Cai, Zongwei Luo, Kun Qian,Y. Gao, "Towards Efficient Selection of Web Services with Reinforcement Learning Process," *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, 2005, pp. 372-376.

[15] Prashant Doshi, Richard Goodwin,R. Akkiraju, "Dynamic Workflow Composition using Markov Decision Processes," IEEE International Conference on Web Services (ICWS'04), 2004.