Intelligent Information Management, **2012, 4, 357-363**
http://dx.doi.org/10.4236/iim.2012.46040 Published Online November 2012 (http://www.SciRP.org/journal/iim)

Emergency Tasks Planning Based on Formal Modeling of Emergency Plan and HTN Planning System SHOP2

Jun Tian, Zhi Li
School of Manageemnt, Xi'an Jiaotong University, Xi'an, China
Email: tianjun@mail.xjtu.edu.cn, lz247@126.com

Received July 31, 2012; revised August 15, 2012; accepted September 15, 2012

ABSTRACT

A theoretical approach of ordered emergency tasks generation is proposed for dealing with a specific emergency event rapidly, exactly and effectively. According to the general principles of an emergency plan developed to response to an emergency management, a workflow model is employed to complete the formal modeling of concrete emergency plan firstly. Then the HTN planning system SHOP2 is introduced, the transformation method of domain knowledge from emergency domain to SHOP2 domain is studied. At last, the general procedure to solve the emergency decision problems and to generate executive emergency tasks is set up drawing support from SHOP2 planning system, which will combine the principles (or knowledge) of emergency plan and the real emergency situations.

Keywords: Emergency Plan; Workflow Model; HTN Planning System; SHOP2; Emergency Task

1. Introduction

Most emergency plan could predefine its launch condition and action principles, also provide important information about the emergency organization structures, the available resources and disposal process, even more, contain some domain knowledge or experiences for the special emergency response so as to give a general guidance to the response actions. But few of them could give out the detailed concrete action schemes which include the accurate tasks to be acted for real operations because of the uncertain situation when emergency event occurred. So, to generate the special tasks supported by computer systems according to the principles provided by emergency plan and the real situations of the emergency environment is the key process to deal with the emergency events in an unhurried manner under the direction of emergency plan [1]. One scientific and rational way to shape emergency disposal scheme is generated via which we have developed beforehand.

We found that the way of contingency task generation based on emergency plan is similar to one method called planning with templates [2]. Template refers to the standard operation steps to solve some typical problems. Emergency plan is just a template of dealing with emergencies. To realize the dynamic programming and generation of emergency disposal scheme based on emergency plan is the thing we need to do. Currently, in dynamic task planning area, more commonly used method is based on the *HTN* (Hierarchical Task Network) planning technology, and many research efforts have been done on such method. US naval laboratory developed HICAP [3] (A planning editor based on hierarchical interactive case planning structure). Its core is to use SHOP (Simple Hierarchical Ordered Planner), a kind of planner based on *HTN* planning technology, to program and generate emergency evacuation plan. Biundo (2001) [4] combined planning method based on state space planning method with *HTN* planning, proposed a kind of hybrid task planning method, which solved out the field modeling difficulties and search space explosion problem. It also has been applied in flood emergency management. Dana Nau (2005) [5] decomposed task using Hierarchical Task Network (*HTN*), and applied standard operating procedure of specific problem domain in planner.

This paper adopts the concept of combining logic programming with planning process firstly, and then discusses a theoretical method which employs the *HTN* planning system especially the SHOP2 into emergency decision-making process to help achieving the dynamic generation of emergency task based on emergency plan template.

2. Formal Modeling of Emergency Plan and *HTN* Planning Technology

2.1. Formal Modeling of Emergency Plan

The formal modeling of emergency plan means abstract out the emergency business process from real-world, then using a formalized way to describe those unstructured or semi-structured text in emergency plan to achieve the standardization management of contingency plan text, and to support emergency decision-making process based on emergency plan [6-8]. With respect to the application condition of HTN planning system, which should have an initial task net, this paper select a formal modeling method based on workflow to achieve formalization description of emergency plan.

Workflow decomposes work into well-defined tasks or role, and then performs these tasks according to certain rules and process. As we mentioned before, severing as emergency decision-making template, emergency plan is composed of a series of emergency plan clips. The emergency response tasks can be generated under guidance of preplan in certain emergency situation via *HTN* planning system. A particularly effective way is to formalize the emergency tasks and their correlations to a workflow model. That is to say the emergency plan template we built is going to be composed of two parts which is tasks and logic relations between them. The Petri-Net method, a tool to build workflow, could be used to model the emergency tasks described by the emergency plan clips and the condition that determine the carry out order of task normatively. And the emergency domain knowledge could be described visualized and efficient in a formal way.

An emergency task completed directly by single emergency entity is called as the ***Primitive Task***. A process consisting of only primitive tasks is defined as ***simple task process***. A task that cannot be accomplished directly is called as ***Complex Task***. Being equal to primitive task process, the process constituted by a set of complex tasks is defined as ***complex task process***. Also, the emergency plan template is composed of two types of templates—simple process template and complex process template.

According to the concept explanation of workflow by Wil Van Der [9], a complex task process built by workflow model based on Petri-Net is composed of tasks and routing structure among tasks, so the emergency task and routing structure are the two basic elements of emergency plan template based on workflow. Specific to expression of workflow based on Petri-Net, the complex task process can be divides into four types: Sequence Control Structure, Parallel Control Structure, Choice Control Structure and Repeat Control Structure (**Figures 1-4** show) When a workflow model only contains a single routing structure, it is called as a basic workflow model. If a workflow model only contains sequence structure, it is called sequence structure workflow model.

2.2. *HTN* Planning Technology

***HTN* (*Hierarchical Task Network*)** [10] planning system is a common and effective method in the intelligent planning field. The main idea of the intelligent planning is to understand and analyze the surrounding environment, to implement reasoning on actions which have several options for selection and on limited resources provided according the goal of user, and eventually work out a plan which can make the goal.

The basic concepts of *HTN* planning are "task" and "method". One task can be an activity, and a "method" describes a kind of implementation way to complete one task. In *HTN* planning process, there are two kinds of task: primitive task and non-primitive task. Primitive task can be performed directly when its precondition is met. Non-primitive task or complex task cannot be performed directly and can be discomposed as a set of related subtasks.

An *HTN* planning problem can be expressed as P = (**d**, **I**, **Op**, **Me**). In it, "**d**" means initial task-nets, that is a group of mission and its constraints needed to be made a plan; "**I**" describes for initial state; "**Op**" means available operating set; "**Me**" indicates usable method set. Each method (Me) describes how to decompose a complex task into subtask, and each operator (**Op**) describes

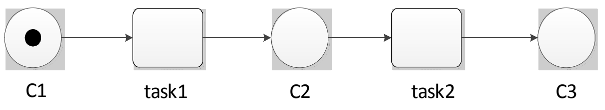

Figure 1. Sequence control structure.

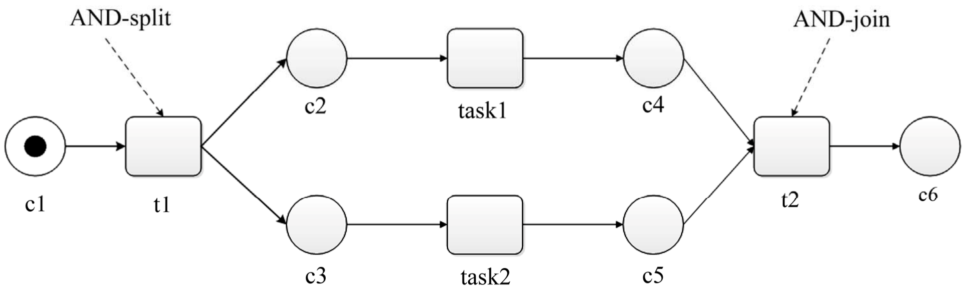

Figure 2. Parallel control structure.

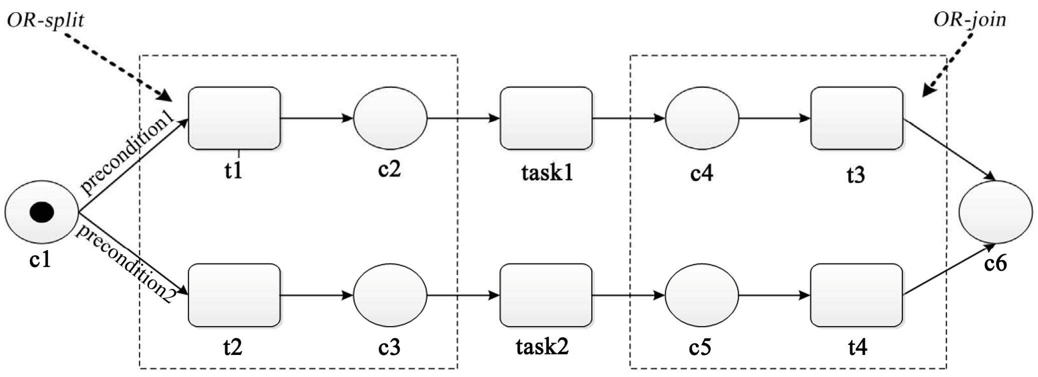

Figure 3. Choice control structure.

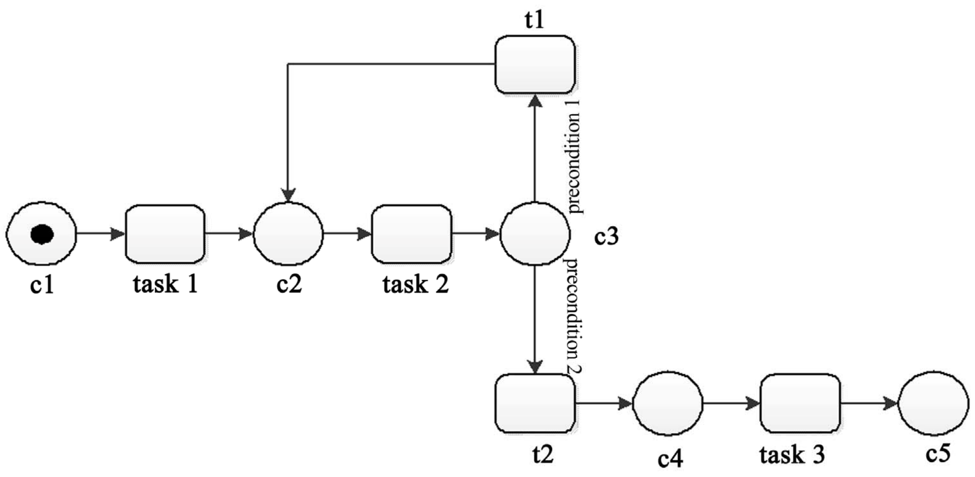

Figure 4. Repeat control structure.

how to realize a primitive task. *HTN* planning is to employ method to decompose complex task into subtask, and this process will continue till all the subtasks are primitive task. *HTN* methods generally describe the "standard operating procedures" that one would normally use to perform tasks in some domain.

SHOP2 planning domain description consists of a series of operator, method and axiom. Planning forward from the initial state, decomposing the given tasks, and ordering in accordance with the same executive order of tasks, SHOP2 can get the planning result [11]. By performing these tasks in this order, SHOP2 can know the current state of each step in the planning process, which can eliminate uncertainty and reduce reasoning complexity. By this way, those actual expressions in the world will be added to the planning system more easily, which also could have the ability of external program calls. In order to complete planning process in a given planning domain, the planning domain knowledge must be given. The SHOP2 knowledge base comprises operations and methods, and be composed of non-action facts and axiom. The operation describes how to complete a primitive task, while the method explains how a particular complex task being decomposed into a series of semi-ordered subtask. The following elements show the characteristics of SHOP2 planning system.

1) TSAK. A task represents an activity to perform. Syntactically, a task consists of a task symbol followed by a list of arguments. A task may be either primitive or compound. A primitive task is one that is supposed to be accomplished by a planning operator: The task symbol is the name of the planning operator to use, and the task's arguments are the parameters for the operator. A compound task is one that needs to be decomposed into smaller tasks using a method; any method whose head unifies with the task symbol and its arguments may potentially be applicable for decomposing the task.

2) OPEARTORS. Each operator indicates how a primitive task can be performed.

Each operator "O" has a head **head (o)** consisting of the operator's name and a list of parameters, a precondition expression **pre (o)** indicating what should be true in the current state in order for the operator to be applicable, and a delete list **del (o)** and add list **add (o)** giving the operator's negative and positive effects. An OPERATOR "O" may have the form: (**h (o) Pre Del Add**).

3) METHODS. Each method indicates how to decompose a compound task into a partially ordered set of subtasks, each of which can be compound or primitive. The simplest version of a method has three parts: The task for which the method is to be used, the precondition that the current state must satisfy in order for the method to be applicable and the subtasks that need to be accomplished in order to accomplish that task. More generally, a method" m" may have the form: (***head*** (***m***) Pre_1T1 Pre_2T2…)

Where, ***head*** (***m***) is a task called the head of m, each $\boldsymbol{Pre_i}$ is a precondition expression and each $\boldsymbol{t_i}$ is a partially ordered set of subtasks. The meaning of this is analogous to an "if-then-else". It tells that if $\boldsymbol{Pre_1}$ is satisfied then $\boldsymbol{t_1}$ should be used, otherwise if $\boldsymbol{Pre_2}$ is satisfied then $\boldsymbol{t_2}$ should be used, and so forth. For keeping the description simple, the paper assume that there is only one precondition expression pre (m) and one set of subtasks sub (m).

4) AXIOMS. The precondition of each method or operator may include conjunctions, disjunctions, negations, universal and existential quantifiers, implications, numerical computations, and external function calls. Furthermore, axioms can be used to infer preconditions that are not explicitly asserted in the current state. For example, (**head tail**) says if head is true than tail is true. The tail of the clause may contain anything that may appear in the precondition of an operator or method.

The emergency task generation dynamic planning problem is ultimately going to be mapped to an SHOP2 planning problem. The SHOP2 planning problem is expressed as a triad (***s***, ***T***, ***D***), among which "***s***" indicates initial condition, "***T***" indicates a set of task, and "***D***" is the description of SHOP2 domain knowledge. Enter the triad as input into SHOP2 planning system, after automatic planning process, SHOP2 would return a set ***P*** = ($\boldsymbol{p_1\ p_2\ \ldots\ p_n}$). The "***P***" returned by SHOP2 means an ordered emergency task sequences which are generated based on the emergency plan under a certain emergency circumstance. Moreover, we want to emphasize that the set ***P*** is a collection of primitive task pi, which means $\boldsymbol{p_i}$ can be performed directly according the description of operators. Thus when all the primitive tasks $\boldsymbol{p_i}$ in *P* are performed under the given order, the emergency response work can be done.

3. Planning and Generation of Emergency Response Task

3.1. Conversion of Domain Knowledge

In order to achieve the planning generation of emergency task based on emergency plan which modeled on workflow through HTN planning technology, To convert the domain knowledge of emergency plan expressed in a workflow form to planning domain knowledge of the SHOP2 planning system [11] is a key problem to solve.

As mentioned in an earlier paragraph, the task processes which are based on emergency plan template can be divided into two kinds: The simple task process (SP) and the complex task process (CP). The CP can be further divided into four types based on routing structure of the workflow model [12]. So the conversion algorithm and an overall conversion program based on different structure are given as follow:

3.1.1. Conversion of Simple Task Process Template

Translate-Simple-process (***S***)

Input: *simple task process* ***S***

Output: *operator* ***O***

Procedure:

1) *take the name of* ***S*** *and parameter set* ***V*** *as the operator's name and parameter set*;

2) *take the precondition* **prec** *of* ***S*** *as the precondition* ***prec*** *of operator*;

3) *take the negative literal in effects of* ***S*** *as Delete Col* ***Delete*** *of operator*;

4) *take the positive literal in effects of* ***S*** *as Add Col* ***Add*** *of operator*;

Return: *O* = (:*operator* (*name P*) *prec Delete Add*)

3.1.2. Conversion of Complex Task Process Template

3.1.2.1. Translate-Sequence-Process (C,SC)

Input: complex task process ***C***, *Sequence Control Structure* ***SC***

Output: *method* ***M***

Procedure:

1) *take the name of* ***C*** *and parameter set* ***V*** *as the name and parameter set of the complex task waiting to be decomposed*;

2) *take the precondition* **prec** *of* ***C*** *as the decompose condition of method*;

3) *take the tasks in* ***SC*** *as subtasks of* ***C****, then perform them in order*;

Return: *M* = (: *method* (*name P*) *Prec* (t_1 t_2))

3.1.2.2. Translate-Parallel-process (C, PC)

Input: *complex task process* ***C***, *Parallel Control Structure* ***PC***

Output: *method* ***M***

Procedure:

1) *take the name of* ***C*** *and parameter set* $\boldsymbol{v}^{\rightarrow}$ *as the name and parameter set of the complex task waiting to be decomposed*;

2) *take the tasks in* ***PC*** *as subtasks of* ***C***, *then perform them in random order*;

Return: *M* = (: *method* (*name P*) *prec* (: *unorderd* t_1 t_2))

3.1.2.3. Translate-Choice-process (C,CC)

Input: *complex task process* ***C***, *Choice Control Structure*

CC

Output: *method* ***M***

Procedure:

1) *take the name of* ***C*** *and parameter set* $\vec{v}$ *as the name and parameter set of the complex task waiting to be decomposed.*

2) *precond*1 = *pre*∩*pre*1, *precond*2 = *pre*∩*pre*2;

Return: *M* = (: *method* (*name P*) *prec* (: *unorderd* t_1 t_2))

3.1.2.4. Translate-Repeat-process (C,RC)

Input: *complex task process* ***C***, *Repeat Control Structure* ***RC***

Output: *a set of method* {***M*1**,***M*2**}

Procedure:

1) *take the name of* ***C*** *and parameter set* $\vec{v}$ *as the name and parameter set of the complex task waiting to be decomposed.*

2) *take the precondition* ***prec*** *of* ***RC*** *as the decompose condition of* ***M*1**;

3) *set complex task process* ***C*1,** *name is* ***name*1,** *parameter set is the parameter set* $\vec{v}$ *of* ***C***;

Return: *M*1 = (: *method* (*name P*) *pre* (*t*1 (*name*1 *P*) *t*5);

*M*2 = (: *method* (*name*1 *P*) *prec* (*t*2(*namel P*1)*()()*)

3.1.3. Overall Conversion Program

Translate-process-Model (*K*)

Input: *a set of* $\boldsymbol{K_i}$, $\boldsymbol{K_i}$ *means the simple task process template or the complex one of emergency plan.*

Output: *SHOP*2 *domain description* ***D***

Procedure:

(1) ***D*** = ∅

(2) *If* $\boldsymbol{K_i}$ *is simple task process template*, *then*

1) *Perform* 0 = *Translate-Simple-process* (K_i);

2) *Add operator model***0** *to domain knowledge model* ***D*** *of SHOP*2;

(3) *If* $\boldsymbol{K_i}$ *is complex task process template*, *then*

1) *Convert emergency plan template* $\boldsymbol{K_i}$ *to basic workflow model set* $\boldsymbol{Q}_{Set} = \{\boldsymbol{Q_1}, \cdots, \boldsymbol{Q_n}\}$, $\boldsymbol{Q_i}$ *is basic workflow model that describes the complex task process* $\boldsymbol{C_i}$;

2) *For arbitrary* $\boldsymbol{Q_i} \in \boldsymbol{Q}_{Set}$, *perform the following steps*:

a) *If* $\boldsymbol{Q_i}$ *is Sequence Control Structure then Perform*
M = *Translate-Sequence-process* (C_i,Q_i);

b) *If* $\boldsymbol{Q_i}$ *is Parallel Control Structure then Perform*
M = *Translate-Parallel-process* (C_i,Q_i);

c) *If* $\boldsymbol{Q_i}$ *is Choice Control Structure then Perform*
M = *Translate-Choice-process* (C_i,Q_i);

d) *If* $\boldsymbol{Q_i}$ *is Repeat Control Structure then Perform*
M = *Translate-Repeat-process* (C_i,Q_i);

(4) *Add* ***M*** *to* ***D***;

(5) *Return* ***D***.

3.2. Generation of Emergency Response Task

After finishing the description to planning problem, identifying the initial state of planning problem and accomplishing the conversion of domain knowledge, emergency response task can be planning generated by the SHOP2 planning system. A simplified version of the SHOP2 planning procedure is below.

Procedure SHOP2 (s,T,D)

```
P = the empty plan
  T_0 ← {t ∈T : no other task in T is constrained to precede t}
  loop
  if T = ∅ then return P
  nondeterministically choose any t ∈T0
  if t is a primitive task then
  A ← {(a,θ): a is a ground instance of an operator in D, θ is a substitution that unifies {head (a), t}, and s satisfies a's preconditions}
  if A = ∅ then return failure
  nondeterministically choose a pair (a,θ) ∈A
  modify s by deleting del(a) and adding add(a)
  append a to P
  modify T by removing t and applying θ
  T_0 ← {t ∈T : no task in T is constrained to precede t}
  else
  M ← {(m,θ) : m is an instance of a method in D,θ unifies {head (m), t},
  Pre (m) is true in s, and m and θ are as general as possible}
  if M = ∅ then return failure
  nondeterministically choose a pair (m,θ) ∈M
  modify T by removing t, adding sub (m), constraining each task in sub (m) to precede the tasks that t preceded, and applying θ
  if sub (m) = ∅ then
  T_0 ← {t ∈sub (m): no task in T is constrained to precede t}
  else T_0 ← {t ∈T: no task in T is constrained to precede t}
  repeat
  end SHOP2
```

When the planning procedure run to the end, SHOP2 returns a set *P* consist of a collection of p_i which possess a certain logic order. The p_i refers to the single emergency task that need to be done under certain emergency circumstance, while *P* is a task set that comprise all the primitive emergency task p_i, which is also called an emergency disposal scheme.

4. Applying Case

4.1. Formal Description of Emergency Plan

Emergency plans are usually text description of a series of emergency planning fragments which are composed of the elements and their logical relationships of the emergency destinations and emergency procedures. Accordinging to the requirements of the generation of emergent task, the task template need to be build by abstracting the element and its logical relations into a model for emergency tasks.

A flood levee risk disposal plan introduced by reference [13], the 10 emergency executive tasks can be abstracted out:

Task 1. Transport anti-flood stone to collapsing location;

Task 2. Throw stone to control the situations of flood;

Task 3. Transport stones to break mouth locations;

Task 4. Transport prefabricated wire cage to break mouth;

Task 5. Fill stone into wire cage and binding;

Task 6. Throw the banded stone cage to break mouth;

Task 7. Transport sandbags to dike danger locations;

Task 8. Building the second sandbags dike;

Task 9. Transported excavator to danger locations;

Task 10. Mine temporary flood road.

Integrating three kinds of sudden emergency situations described by the emergency plan of disposal of regional flood disaster risks, an emergency plan template based on Petri-Net workflow modeling could be got as shown in the **Figure 5**.

4.2. Task Planning Scheme Based on SHOP2

In accordance with the basic workflow model transformation algorithm and overall conversion program of plan domain knowledge, we convert the knowledge described by template into the programming knowledge of SHOP2, and respectively perform the converting program. The corresponding solution can be generated as shown in **Figure 6**.

5. Conclusions

In order to allow emergency plan to achieving its maximum impact of guiding role, this paper presents a theoretical method of emergency response task generation, which is based on formal modeling of emergency plan.

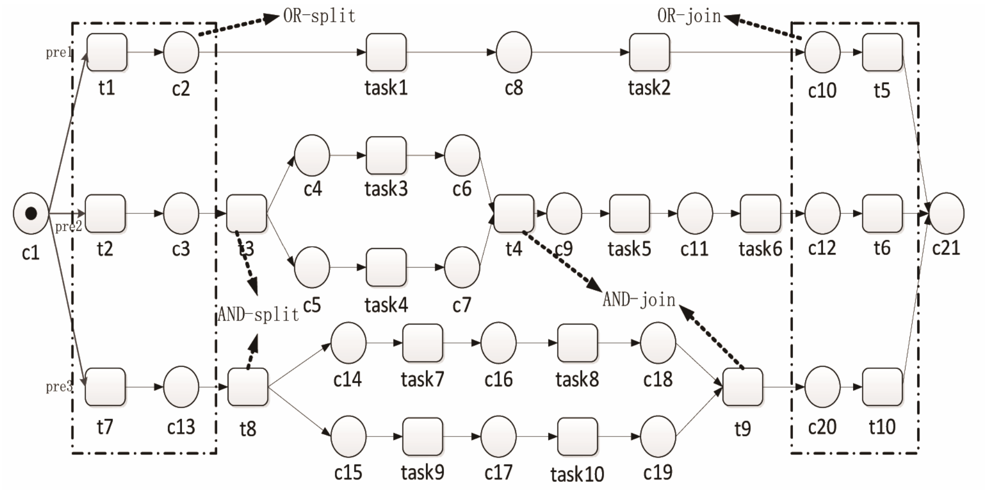

Figure 5. The emergency plan template of regional flood disaster risks disposal.

Planning Problem Instance	Generated Plan
(defproblem problem floodemergency	[1](! reserve Truck1)
;;-----the atoms for the initial state-----	[2](! move Truck1 loc3-1 loc2-1)
((emergency-happen E1 loc1-1)	[3](! load Stone1 truck1 loc2-1)
(material-at Stone1 loc2-1)	[4](! move Truck1 loc2-1 loc1-1)
(material-at Stone2 loc2-2)	[5](! unload Stone1 truck1 loc1-1)
(material-at WireBasket loc2-3)	[6](! move Ttruck1 loc1-1 loc3-1)
(material-at Sandbag loc2-4)	[7](! free Truck1)
(truck-at Truck1 loc3-1)	[8](! dump Stone1 loc1-1)
(truck-at Truck2 loc3-2)	
(truck-at Truck3 loc3-3)	
(truck-at Digger loc4-1)	
(free Truck1)	
(free Truck2)	
(free Truck3)	
(free Digger))	
;;--------------initial task net-------------	
((floodrespose A))	

Figure 6. The task planning scheme based on SHOP2.

In the beginning, the emergency plan described in text is formal modeled as an emergency template based on workflow model using the Petri-Net technology, by which we could complete the description of emergency domain knowledge. After that, the domain knowledge of emergency plan template should be converted to the domain knowledge of SHOP2 planning system. In the end, under certain emergency circumstance, the emergency response task can be generated automatically after the solving process of SHOP2 planning system. We hope this article could give some reference to the way of emergency decision-making under certain emergency situation which requires us to formulate response solutions rapidly and exactly.

Of course, this method has its shortcomings. SHOP2 planning model could not explicitly express the synchronous relationship between time limit conditions and the task, and the task sequences generated by SHOP2 is linear action sequences which cannot adapt to the parallel execution feature of emergency task between different emergency entities. To solve these problems, future work may concentrate on the establishment of a temporal HTN planning model based on SHOP2 planning system.

6. Acknowledgements

This project is supported by the National Natural Science Foundation of China (NSFC) under Grant #70971108.

REFERENCES

[1] J. Tian and Q. Zou, “A Framework of Task-Oriented Decision Support System in Disaster Emergency Response,” *Communications in Computer and Information Science*, Vol. 35, Part 6, 2009, pp. 333-336.

[2] D. Dyer, S. Cross, C. A. Knoblock, *et al.*, “Planning with Templates,” *IEEE Intelligent Systems*, Vol. 20, No. 2, 2005, pp. 13-15. doi:10.1109/MIS.2005.29

[3] H. Munoz-Avila, D. Aha, L. Breslow, *et al.*, “HICAP: An Interactive Case-Based Planning Architecture and Its Application to Noncombatant Evacuation Operations,” *Eleventh Innovative Applications of Artificial Intelligence Conference*, Orlando, 18-22 July 1999, pp. 870-875.

[4] S. Biundo and B. Schattenberg, “From Abstract Crisis to Concrete Relief: A Preliminary Report on Combining State Abstraction and HTN Planning,” *Recent Advances in AI Planning*, *Proceedings of the* 6*th European Conference on Planning*, Toledo, 12-14 September 2001, pp. 157-168.

[5] D. Nau, T. Au, O. Hghami, *et al.*, “Application of SHOP and SHOP2,” *IEEE Intelligent Systems*, Vol. 20, No. 2, 2005, pp. 34-4l. doi:10.1109/MIS.2005.20

[6] M. Grathwohl, F. de Bertrand de Beuvron and F. Rousselot, “A New Application for Description Logics: Disaster Management,” *Proceedings of the International Workshop on Description Logics* 99, Linkoping, 30 July-1 August 1999, p. 56.

[7] W. J. Wang, F. K. Meng, Y. L. Wang, *et al.*, “Research on Ontology-Based Emergency Response Plan Template,” *Computer Engineering*, Vol. 32, No. 19, 2006, pp. 170-172.

[8] M. Hoogendoom, C. M. Jonker, V. Popova, *et al.*, “Formal Modeling and Comparing of Disaster Plans,” *Proceedings of the* 2*nd International ISCRAM Conference*, Brussels, 18-20 April 2005, pp. 97-107.

[9] W. Van Der Aalst and K. Van Hee, “Workflow Management-Models, Methods, and System,” Tsinghua University Press, Beijing, 2004.

[10] K. Erol, D. Nau and J. Hendler, “HTN Planning: Complexity and Expressivity,” *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, 1994, pp. 1123-1128.

[11] D. Nau, *et al.*, “SHOP2: An HTN Planning System,” *Journal of Artificial Intelligence Research*, Vol. 20, 2003, pp. 379-404.

[12] E. Sirin, B. Parsia, D. Wu, *et al.*, “HTN Planning for Web Service Composition Using SHOP2,” *Web Semantics*: *Science*, *Services and Agents on the World Wide Web*, Vol. 1, No. 4, 2004, pp. 377-396.

[13] T. Pan, W. Hong-Wei and W. Zhe, “Method of Modeling Knowledge for HTN Planning Based on Emergency Plan Template and Its Application,” *Chinese Journal of Computer Science*, Vol. 37, No. 10, 2010, pp. 2020-2060.