Collaborative Planning Using Hierarchical Task Network

Teeradaj Racharak

***Abstract*—Planning is one of the critical components in human being's decision making processes. It is a reasoning paradigm where people have to choose and organize actions to satisfy their expected outcomes. In the field of Artificial Intelligence, Automated Planning and Scheduling has become an immense research. Collaborative planning is one of the important planning problems as working together through the act of making choices is fundamental for human nature. This fact is reflected in an emergence of collaborative tools for people's participation. Such tools include social networking sites, instant messengers, email and mailing list, and so on. Unfortunately, these collaborating tools are still functioned based on the notion of human creativity involvement without an automatic planning system. This paper presented the framework to represent collaborative planning problems using HTN formalism, determine the plans, and evaluate the most preferred plan. Three main components are HTN planner(s), Plans validator, and Plan selector. The paper also provided the methodology to solve the problems under the assumption that planning knowledge is decentralized. This is described in terms of communication protocols for decentralized cooperative agents.**

***Index Terms*—Artificial intelligence, collaborative planning representation, htn planning, multi-agent system**

I. Introduction

Planning is one of the critical components in human being's decision making processes. It is a reasoning paradigm where people have to choose and organize actions to satisfy their expected outcomes. In the field of Artificial Intelligence, Automated Planning and Scheduling has become an immense research topic. A diversity of formalisms to find plans were written down by [1] and [2].

Despite diverse formalisms exists, it is a common practice for ones to develop plans together because planning problems are often rich in solutions. Let consider the following examples.

Example I.1: Consider a person, $person_1$, who is at school and wants to be relaxed after an exam. Suppose the $person_1$ has only alternative, which is to play games. Then, a preferred plan for $person_1$ must be an action *play games(.,.).*

Example I.2: Consider two people, $person_1$ and $person_2$, who are at school and want to be relaxed after an exam. Suppose person2 also has only one alternative, which is to watch movies. Then, a preferred plan must be either an action *play games(.,.)* or an action *watch movies(.,.).*

According to the above examples, working together through the act of making choices is fundamental for human nature and the health of individuals and society. This fact is also reflected in an emergence of collaborative tools for people to participate social activities. Such tools include social networking sites (e.g., Facebook, Google+), instant messengers (e.g, WhatsApp, Line), email and mailing list, etc.

The fundamental idea of collaboration is based on the very basic idea of *recursive interaction of knowledge and mutual learning between working people*. In real-life collaboration, this recursive interaction and mutual learning is gained via a method called *Participation*, in which requires an environment to connect people and encourage the sharing of knowledge. In term of Knowledge Management, a similar mechanism is also described as a *Community of Practices* ([3]) to allow a collaboration of people and achieving common outcomes.

Despite diverse formalisms to solve planning problems exist, collaborating tools are still functioned based on the notion of human creativity involvement without an automatic planning system. The difficulties are pointed out as following:

- Planning problems are often rich in solutions because solutions come from distinct people. For the sake of simplicity, solutions coming from distinct people will be referred as *knowledge base* in the paper.
- The same solutions (or plans) may not be able to apply on different sets of collaborating people. This happens since different people may have different preferences. For the sake of simplicity, the solutions will be referred as *plans* in the paper.

The objectives of the paper are to investigate collaborative planning process of people to develop a plan and formulate through a structural method. Developing framework based on the notion of collaborative planning will be proposed in the paper. Major objectives include as follows:

- Present a framework to represent collaborative planning problems, determine the plans, and evaluate the most preferred plan based on the notion of Hierarchical Task Network formalism.
- Provide the methodology to solve the problems under the assumption that planning knowledge is decentralized.

II. Hierarchical Task Network Planning

The Hierarchical Task Network, or HTN, has become popular since it provides a convenient way to write problem-solving recipes corresponding to how a human domain expert might think about solving a planning problem. Its objective is to know how to perform some set of tasks.

As there are various extensions of HTN planning formalism, this work focuses on a special case of HTN planning named *Ordered Task Decomposition*, found by [4]. This special HTN planning always build plans forward from the initial state of the world. Hence, it is obvious to conclude

Manuscript received September 18, 2012; revised December 18, 2012.

Teeradaj Racarak is with the Development Engineer at Octosoft, Thailand, and as a Special Instructor at the Department of Computer Science, Rajamangala University of Technology Thanyaburi, Thailand (e-mail: r.teeradaj@ gmail.com).

that an *ordered task decomposition* planner plans for tasks in the same order that the tasks will later be performed.

Example II.1: Consider a task of arranging simple travel as an example of HTN planning problem. This task can be decomposed into arranging transportation, accommodations, and local transportation. Each of these tasks can also be decomposed based on a variant of transportation and accommodations until primitive actions are reached that can be performed directly using the planning operators.

Definition1 (HTN Planning Problem): An HTN planning is a 4-tuple $P = (s_0, w_0, O, M)$ where s_0 is the initial state, w_0 is a task network, O is a set of operators, and M is a set of methods.

A *task* represents an activity to perform. It consists of a task symbol and a list of arguments. A task is *primitive* if its task symbol is an operator name and its parameters match, otherwise it is *nonprimitive*. In the second example, *arranging transportation* and *arranging accommodation* are nonprimitive tasks, whereas *booking flight* and *booking car* are primitive tasks.

Each operator indicates how much a primitive task can be performed. It is described by a triple $o = (name(o), pre(o), eff(o))$, corresponding to the operator's name, preconditions, and effects. Preconditions are restricted to a set of literals which are supposed to hold. Effects are described as STRIPS-like Add and Delete lists. An operator o is *applicable* in s if its name matches and it can accomplish a ground primitive task in a state s. In the second example, ignoring the parameters, operators might include: *pay*, *book-train*, *book-hotel*, and *book-flight*.

Each method indicates how to decompose a *nonprimitive* task in an ordered set of subtasks, each of which can be either *nonprimitive* or *primitive*. It is described by a 4-tuple $m = (name(m), task(m), precond(m), network(m))$, corresponding to the method's name, correspondent nonprimitive task, preconditions, and task network whose tasks are called *subtasks*. A method m is *relevant* for a task t if there is a substitution $\sigma(t)$ such that $\sigma(t) = task(m)$. Generally, several methods can be relevant to a particular nonprimitive task t, in which lead to different decompositions of t. In the second example, the method with name *by-flight-trans* can be used to decompose the task *arrange-trans* into subtasks of booking a flight and paying.

Definition2 (Task Network): A task network is an acyclic digraph $w = (U, E)$, in which U is the node set, E is the edge set, and each node $u \in U$ contains a task t_u. w is ground if all of the tasks $\{t_u \mid u \in U\}$ are ground; otherwise, w is unground. w is primitive if all of the tasks $\{t_u \mid u \in U\}$ are primitive; otherwise, w is nonprimitive.

Definition3 (Solution to HTN Planning Problem): Given HTN planning problem $P = (s_0, w, O, M)$, a plan $\pi = (a_1, \ldots, a_n)$ is a solution for P, depending on these three cases: (i) if w is empty, then π is a solution for P if π is empty. (ii) if a task node $u \in w$ is primitive and has no predecessors, then π is a solution for P if a_1 is applicable to t_u in s_0 and $\pi = (a_2, \ldots, a_n)$ is a solution for $P' = ((s_0, a_1), w \setminus \{u\}, O, M)$. (iii) if a task node $u \in w$ is nonprimitive and has no predecessors, then there exists a sequence of task decompositions that can applied to w to produce a primitive task network w', where π is a solution for w'.

III. Collaborative Planning

A. An Overview

Consider a situation where three people, namely, $person_1$, $person_2$, and $person_3$, are collaborating to plan for a party. Unfortunately, each person holds different knowledge bases. Those are, $person_1$ knows that to *organize a party*, we need to *play games* and *have dinner*; $person_2$ knows that we can *play drinking games* if we wish to *play games* and we can *have pizza* if we wish to *have dinner*; $person_3$ has no knowledge related to the party domain.

One of the feasible dialogues among them can be as follows:

$Person_1$: "We can play a game and have dinner together for the party."

$Person_2$: "If we wish to play a game, why do not we play a drinking game?"

$Person_2$: "Also, if we wish to have dinner, why do not we have pizza?"

$Person_3$: "I have no idea, but all sound great."

$Person_1$, $person_2$, $person_3$: "Sake and beer are available for us. So, we can use it for our game."

$Person_1$, $person_2$, $person_3$: "Seafood pizza is also available for us. So, we can have seafood pizza."

$Person_3$: "I do not like sake."

For the sake of simplicity to follow the paper, examples are always derived from this working example.

B. Dialogue Context Analysis

According to the working example, the solutions are explored through the notion of *deliberative dialogue*. The *deliberative dialogue* is a form of discussion aimed at finding the best course of actions. The purpose of *deliberative dialogue* is not intended to solve a problem, but to explore the most preferred course of action. A deliberative question generally takes the form *What should we do if we wish to satisfy the goals?*. Cooperative agents may prepare solutions if they have answers. After each cooperative agent is triggered by the question, the solutions are then explored through the deliberation process.

Indeed, finding solutions for a collaborative planning problem through the deliberation process needs a lot of considerations. These considerations can be broken down into the following list:

- What are predefined goals of a planning problem?
- How the knowledge base of cooperative agents is organized?
- How the finished deliberation process is determined?
- How the valid plans of cooperative agents are determined?
- What kind of constraints is used for a collaborative planning?

Solutions to the above questions are done through three major steps: *Planning*, *Validating plans* and *Selecting plans*. Its solutions are roughly figured out under the assumption that each cooperative agent is structured according to the notion of *Hierarchical Task Network*. In other words, a cooperative agent is represented by a 4-tuple $P = (s_0, w_0, O, M)$ where s_0 is the initial state, w_0 is the initial task network, O is a set of operators, and M is a set of methods. The *Hierarchical Task Network* was chosen as the representation of a cooperative agent is because the purpose of a

collaborative planning problem is to arrange a series of tasks by the agents.

1) Planning: Cooperative agents take goal as inputs to a planning problem. The goal can be a sequence of any arbitrary nonprimitive tasks. For an example, a gang of tourists may wish to roam around Bangkok together, so they wish to know a list of places to visit together, A gang of teenagers may wish to play exciting playthings together, etc.. In this working example, the gang of three people wishes to organize a party together.

   Goals: *organizing a party(.,.).*

- After the goal is set, the solutions are then explored through the notion of *deliberative dialogue*. Based on the assumption that cooperative agents organize planning knowledge like *Hierarchical Task Network*, knowledge base of an agent is organized in term of *HTN operators* and *HTN methods*.
- HTN operators express primitive actions which an agent can execute. These primitive actions are considered as ability of an agent, which may be varied by each agent. An example, an agent who has a car will have an operator *drive(Location*1, *Location*2*).*
- HTN methods express sequences of nonprimitive actions an agent have to do in order to solve a given problem. These sequences of nonprimitive actions are considered as recipes. Like HTN operators, HTN methods may vary by each agent. An example, an agent who wishes to travel by a plane knows that the solution is decomposed to travel from a source location to an airport, buy a ticket to a destination airport, fly from a source airport to a destination airport, and go from a destination airport to a destination location.
- On each times of the dialogue deliberation, HTN planner(s) is constructed according to cooperative agents and plans for the initial task network. In particular, the initial task network is equal to a predefined goal at the first times of the dialogue deliberation. Otherwise, the initial task network is equal to the latest developed (partial) plans.
- Despite collaboration is a recursive process, cooperative agents stop doing deliberation if each action belonging to the developed plan is primitive action. When the deliberation is stopped, Validating plans is the next procedural step.
- Theorem1 (Cooperative Agents as HTN planners): A cooperative agent is described as a 2-tuple $A = (O, M)$ where O is a set of HTN operators and M is a set of HTN methods. An HTN operator describes what an agent can do, whereas an HTN method describes recipes for arbitrary problems.

2) Validating plans: A sequence of primitive actions which is a plan for particular cooperative agents may not be a plan for the other. This can happen if some primitive actions in the sequence are not executable by every cooperative agent. In this case, such a sequence of primitive actions is not called a plan for a collaborative planning problem. Considering a situation where two cooperative agents are planning for a party as an example. Suppose neither of them can execute a primitive action drink, then the plans can be any sequences of actions which do not contain the primitive actions drink.

- At this point, the task is to remove sequences of primitive actions which are not called plan for a collaborative planning problem. As HTN operators express an agent's abilities to execute any arbitrary actions, this step just simply remove the sequences of primitive actions which are not executable by every agent.
- Theorem2 (Plans to a collaborative planning problem): Let $A_i = (O_i, M_i)$ be the i^{th} agent for a collaborative planning problem. Considering on n agents, a sequence of primitive actions $\pi = \langle a_1, a_2, a_3, \ldots, a_n \rangle$ is called a plan to a collaborative planning problem if and only if $a_1, a_2, a_3, \ldots, a_n \in \bigcup_{i=1}^{n} O_i$.

3) Selecting plans: In this plans selection step, all the valid plans have already been explored. The task now is to select the most satisfied plan for all cooperative agents to execute together. One may argue that there are no absolute principles for this kind of problems. An example, the most satisfied plan can be a plan whose supportive reasons cannot be defeated. Also, the most satisfied plan can be plans in which most cooperative agents are agree to execute. In this work, the latter one was selected as a criterion to select the most satisfied plan. Therefore, this paper formulated how constraints are used to determine whether an agent will execute an action.

- Theorem3 (The most preferred plan to a collaborative planning problem): Let $\pi = \langle a_1, a_2, a_3, \ldots, a_n \rangle$, where $a_1, a_2, a_3, \ldots, a_n$ are primitive actions, be a plan. A number of cooperative agents who are not preferred to execute a plan π is denoted by $w(\pi)$. Therefore, π is the most preferred plan to a collaborative planning problem if and only if $w(\pi)$ is minimal.

IV. Representing Collaboration Using HTN-Based Planning

This section discusses a way to represent collaborative planning problems using HTN formalism. First-order literals and its logical connectives are used to describe states and actions like the representation of HTN formalism.

In this representation, an agent is described as a 2-tuple $A = (O, M)$ where O is a set of HTN operators and M is a set of HTN methods. An HTN operator describes a primitive action an agent can do and an HTN method describes a recipe an agent know how to decompose a nonprimitive action (Theorem 1).

Let A^i denote the i^{th} cooperative agent from n agents. Therefore, a set of every cooperative agent of a collaborative planning problem is represented by $A_c = \bigcup_{i=1}^{n} A_i$ where n is a number of cooperative agents. As a consequence, a set of every cooperative agent's HTN operators is represented by a label $O_c = \bigcup_{i=1}^{n} O_i$ where n is a number of cooperative agents and a set of every cooperative agent's HTN methods is represented by a label $M_c = \bigcup_{i=1}^{n} M_i$ where n is also a number of cooperative agents.

Definition4 (Collaborative Planning using HTN-based planning): A collaborative planning with multiple cooperative agents is defined as 5-tuple $P = (s_{c0}, w_c, O_c, M_c, \preceq)$ where $s_{c0} \in s_c$ is the initial state, w_c is the task network, O_c is a set of every cooperative agent's operators, M_c is a set of every cooperative agent's methods, and $\preceq$ is a binary relation that is reflexive and transitive on plans. A plan π is a solution to P if

and only if π is a plan for $P' = (s_{c0}, w_c, O_c, M_c, \leq)$ and there does not exist a plan π' such that $\pi' \leq \pi$ according to the set of every cooperative agent's preference formulae $pref_c$.

V. Computing Valid Plans

To this point, all plans have already been explored. The task now is to remove all invalid plans. According to Theorem 2, a sequence of actions $\pi = <a_1, a_2, a_3, \ldots, a_n>$ is *valid* if and only if these actions belong to the set of all planning operators O_c defined by a given problem.

Definition5 (Determining valid plans): A sequence of primitive actions $\pi = <a_1, a_2, a_3, \ldots, a_n>$ is called a *plan* if and only if $\{a_1, a_2, a_3, \ldots, a_n \in O_c\}$ where O_c is a set of all planning operators defined by a given problem.

VI. Computing Preferred Collaborative Plans

This section proposes a basic desire weight and a collaborative preference-based evaluation function which are used to search for the most preferred resulting plans based on the notion of *user-specified preferences*. This paper adapted the idea of [5].

Definition6 (Basic Desire Weight): Let φ be a basic desire formula and let α be a plan. The weight of the plan α with respect to the basic desire φ is a function defined as the following figure.

$$w_{\varphi}(\alpha) = \begin{cases} 1 & \text{if } \alpha \not\models \varphi \\ 0 & \text{otherwise} \end{cases}$$

Fig. 1. Basic desire weight formula.

Definition7 (Collaborative Preference-based Evaluation Function): Let $W(\pi)$ be a collaborative preference-based evaluation function of a plan π corresponding to every cooperative agent. Then, $W(\pi) = \sum_{i=1}^{n} w_{pref}^{i}(\pi)$ where n is a number of every cooperative agent and $pref^i$ is a user-specified preferences set of a cooperative agent.

Definition8: Let π_1, π_2 be two plans. Then, a collaborative preference-based evaluation function of a plan π_1 and a collaborative preference-based evaluation function of a plan π_2 are denoted by $W(\pi_1)$ and $W(\pi_2)$, respectively. A plan π_1 is at least as preferred as a plan π_2, denoted by $\pi_1 \leq \pi_2$, if $W(\pi_1) \leq W(\pi_2)$.

Remark1: Let $\pi_1, \pi_2, \ldots, \pi_n$ be all feasible plans for a collaborative planning problem. Suppose that $W(\pi_1)$ results in the minimal value. Therefore, π_1 is the most preferred plan.

VII. Formulating Dialogue Communication

This section discusses the communication for different cooperative agents if these agents are decentralized. This is needed in order to make the collaborative planning problem more easily manageable if planning knowledge of different cooperative agents is stored in different physical planners. The formulation proposed here is based on HTN formalism as the representation is extended the notion of abstract task decomposition like HTN.

Definition9: Let P be a collaborative planning problem as defined above. Every decentralized cooperative agent is transferring a task network w back and forth. Then,

- An agent A_i is said to be *quasi-exactly solvable* with respect to a task network w_c if and only if w_c is nonprimitive and there exists a method $m \in M_i$ that is *capable* to and *relevant* for the network w_c.
- An agent A_i is said to be *quasi-exactly unsolvable* with respect to a task network w_c if and only if w_c is nonprimitive and there doesn't exist a method $m \in M_i$ that is *applicable* to and *relevant* for the network w_c.
- An agent A_i is said to be *exactly solvable* with respect to a task network w_c if and only if w_c is primitive and is a solution to the problem P.
- An agent A_i is said to *exactly unsolvable* with respect to a task network w_c if and only if w_c is primitive and is not a solution to the problem P.

VIII. An Abstract Algorithm Of Collaborative Protocols

This section describes an abstract algorithm to collaborate plans among associated agents. The algorithm is derived from the definition *Collaboration using HTNs*.

```
1: function ONRECEIVED(tasknetwork w)
2:     if w ∉ ReceivedNetwork then
3:         store w in ReceivedNetwork
4:         if w is quasi-exactly solvable then
5:             decompose w into possible
   subnetworks(w_1 ... w_k)
6:             for 1 ≤ m ≤ k do
7:                 if w_m is quasi-exactly unsolvable then
8:                     broadcast(w_m)
9:                 else
10:                    check consistency(w_m)
11:                    broadcast(w_m)
12:                end if
13:            end for
14:        else
15:            if w is quasi-exactly unsolvable then
16:                broadcast(w)
17:            else
18:                check consistency(w_m)
19:            end if
20:        end if
21:    else
22:        exhausted w
23:    end if
24: end function
```

Fig. 2. An abstract algorithm for plan collaboration.

Figure 2 describes the pseudo code for collaboratively develop plans among associated agents. Input to the algorithm is a task network w defined in the definition 5 and output is a list of feasible full plans being developed by exchanging the task networks.

The *broadcast* takes as input w where w is a task network and functions to send a task network to all the agents. The algorithm also assumes that all the agents have to know each other in advance. The *check consistency* takes as input w where w is a task network and functions to validate it against the agent's constraints.

A. Soundness and Completeness of the Proposed Algorithm

Theorem4 (Soundness): Let $P = (s_0^* \cup \bigcup_{i=1}^{n} s_0^i, w, \bigcup_{i=1}^{n} O_i,$

$\bigcup_{i=1}^{n} M_i$) be a HTN planning. Suppose one of the nondeterministic developments of the algorithm given in the figure 2 finds a plan π. Then, π solves the HTN planning problem *P*.

Proof: Let us first notice that an agent has never struggled on the same task network *w* more than twice (line 22). Therefore, there are finite numbers of times an agent will be triggered according to the algorithm.

The proof is done by induction on *n*, where *n* is the number of times an agent being triggered such that a plan $\pi = (a_1, \ldots, a_j)$ which is a solution of *P* will be collaboratively developed with others.

Base case (*n*=1): In this case, an agent does not need to collaborate a plan π with others. Thus, there are two cases:

- *w* is empty and π is empty. According to the case 1 of the definition 3, π solves the planning problem *P*.
- *w* is primitive and is either *exactly solvable* and *exactly unsolvable* with respect to the agent (line 11 and line 18). According to the case 2 of the definition 3, π solves the planning problem *P*.

Induction step: Let $n > 1$. Suppose that the theorem is true for every $m < n$. According to the inductive assumption, a full plan $\pi = (a_1, \ldots, a_j)$ can be collaboratively developed at $n - 1$ steps. Then, it follows that the agent is said to be either *exactly solvable* or *exactly unsolvable* with respect to π. According to the case 2 of the definition 3, π solves the planning problem *P*.

Proposition1. The algorithm given in the figure 2 provides a complete solution if a *preference-based HTN planner* is implemented based on *ordered task decomposition*.

Theorem5 (Completeness): Suppose that a HTN planning $P = (s_0^* \cup \bigcup_{i=1}^{n} s_0^i, w, \bigcup_{i=1}^{n} O_i, \bigcup_{i=1}^{n} M_i)$ is solvable. Then, all of the nondeterministic traces $\pi_1, \ldots, \pi_j$ based on the algorithm given in the figure 2 can be developed.

Proof: Similar to the previous theorem, let us first notice that an agent has never struggled on the same task network *w* more than twice (line 22). Therefore, there are finite numbers of times an agent will be triggered according to the algorithm.

The proof is done by induction on *n*, where *n* is the number of times an agent being triggered such that a plan $\pi = (a_1, \ldots, a_j)$ which is a solution of *P* will be collaboratively developed with others.

Base case (*n*=1): In this case, an agent does not need to collaborate a plan π with others. Thus, there are two cases:

- *w* is primitive and there exists a solution with respect to *w*. Thus, the algorithm terminates at line 18.
- *w* is nonprimitive and there exists a solution with respect to *w*. Thus, the algorithm terminates at line 16.

Induction step: Let $n > 0$. Suppose that the theorem is true for every $m < n$. There are two cases:

- $w = (w_1, \ldots, w_k)$ for some *k* and w_1 is primitive. Then, there exists a decomposition a_{11} for w_1. According to the inductive assumption, $(a_{12}, \ldots, a_{1k})$ can be collaboratively developed at step $n - 1$. Thus, the algorithm terminates a plan $(a_{11}, \ldots, a_{1k})$ at line 11.
- $w = (w_1, \ldots, w_k)$ for some *k* and w_1 is nonprimitive. Then, there exists a decomposition $D = (d_1 1, \ldots, d_1 l)$ for w_1. According to the inductive assumption, $(a_{12}, \ldots, a_{1k})$ can be collaboratively developed at step $n - 1$. Thus, the algorithm terminates a plan $(d_1 1, \ldots, d_1 l, a_{11}, \ldots, a_{1k})$ at line 8.

REFERENCES

[1] S. J. Russell and P. Norvig, "Artificial intelligence: A modern approach," *Upper Saddle River*, New Jersey: Prentice Hall, 2003.

[2] M. Ghallab, D. Nau, and P. Traverso, "Automated planning: Theory and practice," San Francisco: Morgan Kaufmann, 2004.

[3] P. Duguid, "The art of knowing: Social and tacit dimensions of knowledge and the limits of the community of practice," University of California, 2005.

[4] D. S. Nau, S. J. Smith, and K. Erol, "Control strategies in htn planning: Theory versus practice," in *AAAI-98/IAAI-98 Proceedings*. AAAI Press, 1998, pp. 1127-1133.

[5] T. C. Son and E. Pontelli, "Planning with preferences using logic programming," in *Proc. LPNMR' 04*. Springer, 2004, pp. 247-260.

Teeradaj Racharak was born in Bangkok, Thailand on 18th of September 1988. The author did his master's degree in Computer Science with specialization in Software Engineering from Asian Institute of Technology, Thailand in 2012. Currently, the author works as a software development engineer at Octosoft, Thailand, and as a special instructor at the department of Computer Science, Rajamangala University of Technology Thanyaburi, Thailand.