ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1013

Utility-Based HTN Planning

Ilche Georgievski and Alexander Lazovik [1]

Abstract. We propose the use of HTN planning for risk-sensitive planning domains. We suggest utility functions that reflect the risk attitude of compound tasks, and adapt a best-first search algorithm to take such utilities into account.

1 INTRODUCTION

Hierarchical Task Network (HTN) planning [3] is successfully used in solving problems of various modern applications, for instance, Web services [9] and ubiquitous computing [1, 2]. Real-world domains are characterised with unpredictability of alternatives encountered during planning which relates to *risk*. The sensitivity of planning techniques to risk is especially useful for applications with large wins or loses of *resources*, such as money, power, equipment, time and even humans. If a planning technique takes indifferent attitude towards risk, the result may be an undesirable outcome. Consider the following example. A smart building is confronted with an emergency situation due to fire. The building needs to react in such a way that all its occupants will be rescued in as short a time as possible. Let us assume that the building can accomplish the deliverance in two ways. One way involves guiding occupants through a very fast track but with a high risk to encounter flames on the route to the exit. Otherwise, the planner may choose a longer track: through this track occupants can be guided to escape the building with a lesser change of danger of fire breaking out. Which track to decide on?

We propose a framework based on HTN planning that takes task utilities into account, where a utility is a function of a profit and may determine the attitude towards risk. We assume that a primitive task is associated with a function of consumption that expresses a single or combination of *properties*, such as failure rate and energy use. Thus, the possible runtime failures are not modelled directly, but we assume that some tasks are more likely to show unpredictable behaviour. Under the assumption that a compound task may have a large number of methods and many of them applicable, we use a *utility* to express the level of preference of the compound task, and a *utility function* to calculate the perceived utility of the task. The algorithm selects the best task network to work with, that is, the one with the lowest estimated value.

Related Work. To the best of our knowledge, risk sensitivity and utilities have not been addressed in HTN planning yet, but they have been investigated in other planning models that are naturally and conceptually different than HTN planning (*e.g.*, [4]). However, preferred plans have been a point of interest in HTN planning. In [10], a set of user preferences and incremental best-first search is used to find the most preferred plan. Nau *et al*. [7] require costs to be associated to operators and use rather limited branch-and-bound optimisation to search the best plan. Luo *et al*. [6] use utility functions of operators and a messy genetic algorithm to search for the best solution. In [1], context-aware adaptations seem to take specific resource-related properties into account. Finally, Kuter and Golbeck [5] use user ratings and trust in services to find the most trustful service composition in social environments. The studies that take operator costs into account to search for the "best" solution neglect the utilities of compound tasks. In the studies where the costs of compound tasks is supposed to have some decision-making role (*e.g.*, [1]), such costs are encoded manually into the domain knowledge without formal groundings of the meaning and calculation. The closest approach to ours is [5] where user ratings of atomic services are used to calculate backwards the trust of composite services.

[1] University of Groningen, The Netherlands, email: initial.surname@rug.nl

2 PRELIMINARIES

Our formalism of HTN planning adopts many definitions from [3]. A state is a set of ground predicates. A primitive task is an expression of the form $pt(\tau)$, where pt is a primitive-task symbol, and $\tau = \tau_1, \ldots, \tau_n$ are terms. A compound task is defined similarly. The set of primitive and compound tasks is a finite set of task names TN.

An *operator* o is a triple $\langle pt(o), pre(o), eff(o)\rangle$, where $pt(o)$ is a primitive task, $pre(o)$ and $eff(o)$ are precondition and effect, respectively. An operator o is *applicable* in a state s iff $pre(o) \subseteq s$. Applying o to s results into a new state $s[o] = s \cup eff(o)$.

A *task* t is a pair $\langle ct(t), M_t\rangle$, where $ct(t)$ is a compound task, and M_t is a set of methods. A *method* m is a pair $\langle pre(m), tn(m)\rangle$, where $pre(m)$ is a precondition and $tn(m)$ is a task network. A method m is *applicable* in a state s iff $pre(m) \subseteq s$. Given a task t such that $m \in M_t$, applying m to s results into a task network $tn(m) = (s[m], t)$. A *task network* tn is a pair $\langle T_n, \prec\rangle$, where $T_n \subseteq TN$, and $\prec$ is a partial order on T_n.

Definition 1 (HTN Planning Problem)**.** An *HTN planning problem* P is a tuple $\langle s_0, tn_0, O, T\rangle$, where s_0 is an initial state, tn_0 is an initial task network, O and T are sets of operators and tasks, respectively.

Definition 2 (Solution)**.** A sequence of operators $o_1, \ldots, o_n$ is a *solution* to P, if and only if there exists a task $t \in T_0$, where $tn_0 = \langle T_o, \prec_0\rangle$, such that $(t, t') \in \prec_0$ for all $t' \in T_0$ and

1. t (or o_1) is primitive and applicable in s_0 such that $o_2, \ldots, o_n$ is a solution to $P = \langle s_0[o_1], tn_0 \setminus \{o_1\}, O, M\rangle$; or
2. t is compound and there exists an applicable method m such that $tn(m) = (s_0[m], t)$, $tn' = tn_0 \setminus \{t\} \cup tn(m)$, and $o_1, \ldots, o_n$ is a solution to $P = \langle s_0, tn', O, T\rangle$.

3 UTILITIES

We require a primitive task to be provided with some function of consumption that expresses the cost of the operator as a non-negative

value, that is, $c(o) \in \mathbb{R}_{\geq 0}$. We estimate the utility of a compound task t based on its risk or consumption attitude. The following function represents a template formula for calculating the utility value of t. The definition of the estimation factor E gives a concrete utility function. We assume that each t is *acyclic*, that is, t can be decomposed only to a finite depth. Figure 1 serves as a demonstrating example of such a task, where the leaf tasks are operators associated with non-negative costs.

$$U(t) = \begin{cases} c(o), & \text{if } t \text{ is primitive;} \\ \min\limits_{m \in M_t} E(m), & \text{otherwise.} \end{cases}$$

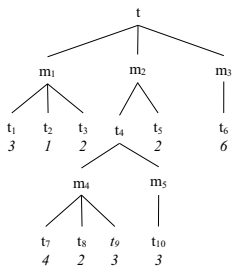

Figure 1. Example of a task

Risk-averse utility. A task is *risk-averse* if and only if it minimises the maximum expected utility value of its methods' subtasks. A risk-averse task shows pessimistic behaviour towards risk, that is, it represents the safest possible decision by using $E(m) = \max\limits_{t' \in tn(m)} U(t')$. Applying this risk-averse attitude on our example results in $U(t) = 3$.

Risk-seeking utility. A task is *risk-seeking* if and only if it minimises the minimum expected utility value of its methods' subtasks. A risk-seeking task shows optimistic behaviour towards risk, that is, it takes the highest risk to gain the best outcome by using $E(m) = \min\limits_{t' \in tn(m)} U(t')$. Considering risk-seeking attitude of the task in our example, then $U(t) = 1$.

Risk-neutral utility. A task is *risk-neutral* if and only if it minimises the average expected utility value of all subtasks for a given task's method, that is, $E(m) = \frac{\sum_{t' \in tn(m)} u_3(t')}{|tn(m)|}$. Applying the risk-neutral attitude on our demonstrating example gives $U(t) = 2$.

Consumption-aware utility. A task is *consumption-aware* if and only if it minimises the sums of utility values of its methods, that is, $E(m) = \sum_{t' \in tn(m)} U(t')$. Considering such consumption-aware attitude of the task in the demonstrating example, then $U(t) = 5$.

3.1 Algorithm

We take a node n to be a three-element structure $\langle s, tn, \pi \rangle$, where s is a state, tn is a task network, and π is a partial plan. A node n is selected for expansion based on one of the utility functions that can be seen as a heuristic function. The utility-based best-first search algorithm (Algorithm 1) takes an HTN planning problem, some resource value and a utility function, and starts by initialising the frontier. The nodes whose utility value is greater than the amount of resource are pruned from the search space. If the node has no more tasks to be decomposed, and the utility of its partial plan is less than the utility of the best plan found so far, then we consider the current one as the best plan. The function in line 10 is a decomposition method, such as the partially ordered decomposition in [3], which considers the applicability of operators and methods in the state of the current node.

Theorem 1 (Optimality)**.** Given an HTN planning problem P, a utility function U, and a resource value, if the termination of the algorithm returns a plan, then the plan is an optimal solution to P.

The proof of the theorem follows from the definition and properties of the best-first search algorithm [8].

Algorithm 1 Find a sequence of operators

```
1:  currentUtility ← ∞
2:  frontier ← ⟨s0, tn0, ∅⟩
3:  while frontier ≠ ∅ do
4:      best ← POP(frontier)
5:      if resource > U(best.tn) then
6:          if best.tn = ∅ and E(best.π) < currentUtility then
7:              print best.π
8:              currentUtility ← E(best.π)
9:          end if
10:         decompositions ← DECOMPOSE(best)
11:         frontier ← MERGE(decompositions, frontier, U))
12:     end if
13: end while
```

4 CONCLUSION

We have introduced a framework for utility-based HTN planning and utilities of compound tasks based on their risk attitude to find optimal solutions. We will further extend the framework (*e.g.*, the case of cyclic tasks), and we will demonstrate the benefits of utility-based HTN planning in the domain of smart buildings.

ACKNOWLEDGEMENTS

The work is supported by the Dutch National Research Council under the NWO Smart Energy Systems program, contract no. 647.000.004.

REFERENCES

[1] F. Amigoni, N. Gatti, C. Pinciroli, and M. Roveri, 'What Planner for Ambient Intelligence Applications?', *SMC, Part A*, **35**(1), 7–21, (2005).
[2] I. Georgievski, T. A. Nguyen, and M. Aiello, 'Combining Activity Recognition and AI Planning for Energy-Saving Offices', in *UIC*, pp. 238–245, (2013).
[3] M. Ghallab, D. S. Nau, and P. Traverso, *Automated Planning: Theory & Practice*, Morgan Kaufmann, 2004.
[4] S. Koenig and R. G. Simmons, 'Risk-sensitive planning with probabilistic decision graphs', in *KR*, pp. 363–373, (1994).
[5] U. Kuter and J. Golbeck, 'Semantic Web Service Composition in Social Environments', in *ISWC*, pp. 344–358, (2009).
[6] J. Luo, C. Zhu, W. Zhang, and Z. Liu, 'Messy Genetic Algorithm for Optimum Solution Search of HTN Planning', *JICS*, **10**(5), 1303–1313, (2013).
[7] D. S. Nau, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, 'SHOP2: An HTN Planning System', *JAIR*, **20**(1), 379–404, (2003).
[8] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, 2003.
[9] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. S. Nau, 'HTN Planning for Web Service Composition Using SHOP2', *WS*, **1**, 377–396, (2004).
[10] S. Sohrabi, J. A. Baier, and S. A. McIlraith, 'HTN Planning with Preferences', in *IJCAI*, pp. 1790–1797, (2009).