On the Decidability of HTN Planning with Task Insertion

Thomas Geier and Pascal Bercher
Institute of Artificial Intelligence, Ulm University, Germany

IJCAI 2011, International Joint Conference on Artificial Intelligence, 19-22 July 2011, Barcelona, Spain

We give a simplified propositional HTN formalization inspired by the formalization of Erol et al. [2] and show
that the plan existence problem is still undecidable despite the simplifications.

HTN planning with insertion (hybrid planning) is decidable; from the proof of decidability, we obtain an upper complexity bound of EXPSPACE for the plan existence problem for propositional hybrid planning.

Abstract

Definition (Task Network)

A task network \( n = \langle T, \prec \rangle \) is a partially ordered sequence of tasks:

- \( T \) is a finite set of non-empty set of tasks
- \( \prec \subseteq T \times T \) is a strict partial order on \( T \)

Definition (Planning Problem)

A planning problem is a 6-tuple \( P = \langle V, C, O, M, o, n \rangle \) and

- \( V \) is a finite set of state variables
- \( C \) is a finite set of compound tasks
- \( O \) is a finite set of primitive tasks, for all \( o \in O \), \( \text{prec}(o) \cup \text{add}(o) \cup \text{del}(o) \) is a subset of \( C \times (C \times \mathbb{N}) \)
- \( M \subseteq C \times TV \) is a finite set of decomposition methods
- \( o_n \in C \) is the initial task
- \( o_f \in C \) is the final state

Note that the part of the planning problem that is usually called the plan existence problem for propositional hybrid planning is given by: \( P = \langle V, C, O, M, o, n \rangle \)

Possible Modifications

1. Decomposition:
   - Given a task network \( n = \langle T, \prec \rangle \), use method \( (c, o_n) \in M \) to replace \( c \in T \) by \( o_n \). Then adjust ordering constraints.
2. Insertion:
   - Insert primitive tasks from \( \mathcal{O} \)
   - Insert ordering constraints

Figure 2: decomposition methods

Here, the decomposition methods describe two context-free grammars (CFGs), their languages are \( L(G) = \langle a,a,b \rangle \) and \( L(G') = \langle a,b \rangle \).

Solution Criterion

1. A task network \( n \) is an HTN solution iff:
   - \( n \) is obtained via decomposition
   - There is an executable linearization of \( n \)’s tasks
2. A task network \( n \) is a hybrid solution iff:
   - \( n \) is obtained by decomposition followed by insertion
   - There is an executable linearization of \( n \)’s tasks

Solution

Given a task network \( n = \langle T, \prec \rangle \) and a task \( o \in O \),

1. remove all cycles from the decomposition tree
2. replace the removed elements using task and ordering insertion

Any hybrid solution to \( P \) can be constructed using at most \( |P| \) decompositions where \( |P| \) is the number of tasks inside the largest method. (maximum branching factor of the decomposition tree)

Possible Modifications

- The plan existence problem is decidable for hybrid planning.

Theorem

Proof Idea:

- To establish an upper bound on the size of shortest solutions, enumerate all hybrid planning instances and check whether they are solutions.
- If no solution has been found then we know that no solution exists at all.

Proof — Bounding Decomposition

We apply the idea of the pumping lemma for context-free grammars to task decomposition:

1. remove all cycles from the decomposition tree
2. replace the removed elements using task and ordering insertion
3. remove all task sequences that produce loops in the state space that do not contain tasks from \( n \)
4. obtain a solution which contains at most \( n^{(2^{|P|}+1)} \) tasks

Proof — Bounding Task Insertion

Given a task network \( n \) with a task, we have to insert at most \( n^{2^{|P|}} \) additional tasks to turn it into a solution, if this is possible at all.

Theorem

1. To obtain the bound, fix a (totally ordered) solution that contains \( n \), thus it can be obtained from \( n \) via insertion
2. remove all task sequences that produce loops in the state space and that do not contain tasks from \( n \)
3. obtain a solution which contains at most \( n^{2^{|P|}+1} \) tasks

Proof — Bounding Solutions

For every planning problem \( P \), there exists a hybrid solution with at most \( n^{2^{|P|}+1} \) tasks if such a solution exists at all.

The bound follows from the bounds on task decomposition and the bound on task insertion.

Decidability Criteria

- Imposing certain restrictions on planning problems can make HTN planning decidable:
  - Completion: Decomposition tree is acyclic
  - Intuition: search space becomes finite.

- Categorization: All methods are totally ordered (cf. SHOP system [5])
  - Intuition: the combinations of possible states before and after the abstract task are finite.

- Categorization: Methods contain at most one compound task (regular)
  - Intuition: the plan existence problem for propositional hybrid planning is still undecidable despite the simplifications.

- Categorization: Methods contain at most \( n \) compound tasks (decomposable)
  - Intuition: the plan existence problem for propositional hybrid planning is still undecidable despite the simplifications.

Conclusion & Discussion

- Allowing task insertion makes HTN planning decidable resulting in the following complexity classes [1, 3, 4]:

<table>
<thead>
<tr>
<th>Computational Complexity</th>
<th>HTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>hybrid</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>propositional</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>

One can think of HTN planning with insertion having a different meaning than HTN planning:

- HTN planning: specifies, what must not be in a plan
- Hybrid planning: specifies, what has to be in a plan

References

[2] Kutluhan Erol, James Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierar-