Tight Bounds for HTN Planning with Task Insertion (Extended Abstract)*

Ron Alford  
ASEE/NRL Postdoctoral Fellow  
Washington, DC, USA  
ronald.alford.ctr@nrl.navy.mil

Pascal Bercher  
Ulm University  
Ulm, Germany  
pascal.bercher@uni-ulm.de

David W. Aha  
U.S. Naval Research Laboratory  
Washington, DC, USA  
david.aha@nrl.navy.mil

Abstract
Hierarchical Task Network (HTN) planning with task insertion (TIHTN planning) is a variant of HTN planning. In HTN planning, the only means to alter task networks is to decompose compound tasks. In TIHTN planning, tasks may also be inserted directly. In this paper we provide tight complexity bounds for TIHTN planning along two axis: whether variables are allowed and whether methods must be totally ordered.

Background & Motivation
The Hierarchical Task Network (HTN) planning paradigm is centered around planning the execution of tasks (Erol, Hendler, and Nau 1996). Fundamental to HTN planning is the task network, which is a partially-ordered multiset of task names, each representing an activity to be accomplished. Tasks are either primitive, corresponding to an action that can be taken whenever its precondition is met, or non-primitive, which must be iteratively refined into an executable sequence of actions. There are two types of refinement available to an HTN planner: either imposing an ordering on two tasks, restricting the task network’s partial order, or by decomposing a non-primitive task. For every non-primitive task name, there is an associated set of methods, each containing a task network. We decompose a non-primitive task name by replacing it with the task network from one of its associated methods.

Although HTN planning has found numerous practical applications (Nau et al. 2005), it can be difficult to write complete and effective sets of methods (Shivashankar et al. 2013). A principled approach to this problem is that of HTN planning with Task Insertion (TIHTN Planning), which adds an additional refinement operator beyond that of the HTN formalism: the ability to insert any task name into the task network without regard to the decomposition hierarchy (Geier and Bercher 2011). Task insertion is also allowed by other hierarchical planning approaches, such as hybrid planning (Kambhampati, Mali, and Srivastava 1998; Biundo and Schattenberg 2001) – a planning approach fusing HTN planning with Partial-Order Causal-Link (POCL) planning. Task insertion can also be seen as the underlying formalism in hierarchical goal systems such as GoDel (Shivashankar et al. 2013).

Somewhat surprisingly, allowing task insertion reduces the worst case complexity of deciding the plan existence problem. While even propositional HTN planning is undecidable, propositional TIHTN planning is known to be in EXPSPACE (Geier and Bercher 2011). Here, we lower that bound for the general case and further investigate the influence of variables (or lifting) and the ordering of the task networks given in the decomposition methods. We give a colloquial overview of our results. For formal definitions and proofs we refer to the full paper (Alford, Bercher, and Aha 2015b).

Results and Comparison
One of the key insights into TIHTN planning is that any solution can be mimicked by a mixture of task insertion and acyclic decomposition, where no task name is ever decomposed using a method that contains the same task name as an ancestor (Geier and Bercher 2011). We can exploit this insight to define acyclic progression for TIHTN problems using the progression planning technique commonly used in HTN planning (Alford et al. 2012).

Under acyclic progression, a TIHTN planning algorithm needs only to consider a small portion of the task network at the time. Using that technique we obtain tighter bounds for TIHTN planning given only totally-ordered methods.

Theorem 1. TIHTN planning for propositional problems with totally-ordered methods is PSPACE-complete. For lifted problems (having variables) with totally-ordered methods, TIHTN planning is EXPSPACE-complete.

For partially-ordered TIHTN problems, acyclic progression may need decompose all or most of the task network before imposing ordering constraints. The number of required task insertions, however, is limited by the number of expressible states.

Theorem 2. Propositional TIHTN planning is NEXPTIME-complete; lifted TIHTN planning is 2-NEXPTIME-complete.

Partially-ordered problems are not always more computationally challenging than totally-ordered problems. Regular problems are those where every method contains at most
Our major contribution is in providing six new completeness results for common classes of TIHTN planning, including better bounds for the general case. This provides both worst-case lower bounds for any sound and complete TIHTN algorithm; and matching worst-case upper bounds, which provides a theoretical target for future algorithms to match. The lower bounds, in particular, should help focus TIHTN research, as they provide firm limits on what soundness results for HTN planning.

Our major contribution is in providing six new completeness results for common classes of TIHTN planning, including better bounds for the general case. This provides both worst-case lower bounds for any sound and complete TIHTN algorithm; and matching worst-case upper bounds, which provides a theoretical target for future algorithms to match. The lower bounds, in particular, should help focus TIHTN research, as they provide firm limits on what soundness results for HTN planning.

Our major contribution is in providing six new completeness results for common classes of TIHTN planning, including better bounds for the general case. This provides both worst-case lower bounds for any sound and complete TIHTN algorithm; and matching worst-case upper bounds, which provides a theoretical target for future algorithms to match. The lower bounds, in particular, should help focus TIHTN research, as they provide firm limits on what soundness results for HTN planning.

Our major contribution is in providing six new completeness results for common classes of TIHTN planning, including better bounds for the general case. This provides both worst-case lower bounds for any sound and complete TIHTN algorithm; and matching worst-case upper bounds, which provides a theoretical target for future algorithms to match. The lower bounds, in particular, should help focus TIHTN research, as they provide firm limits on what soundness results for HTN planning.