Tight Bounds for HTN Planning with Task Insertion

Ron Alford
ASEE-NRL Postdoctoral Fellow
Washington DC, USA

Pascal Bercher
Ulm University
Ulm, Germany

David W. Aha
U.S. Naval Research Laboratory
Washington DC, USA

Abstract

HTN planning is the problem of decomposing an initial task to accomplish into a sequence of executable steps. HTN planning with Task Insertion (TIHTN planning) allows the insertion of operators from outside the method hierarchy, which:

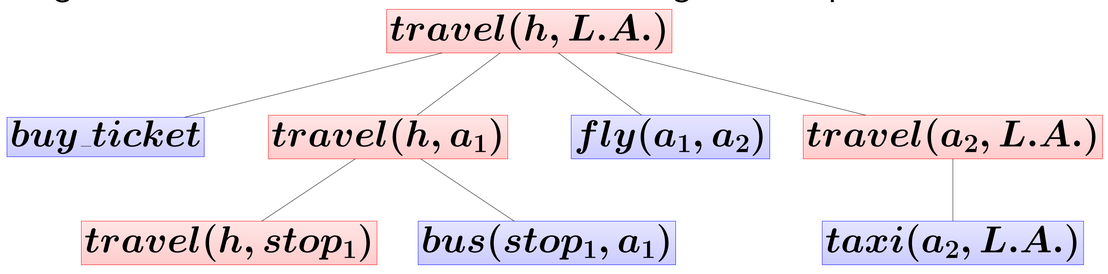
- Hybridizes classical planning with HTN planning
- Allows partial task hierarchies with "missing required tasks" inserted by planner

We provide tight complexity bounds for TIHTN planning along two axis: whether variables are allowed and whether methods must be totally ordered.

HTN Planning (Overview)

The purpose of HTN planning is to complete a *task*. Tasks are either:

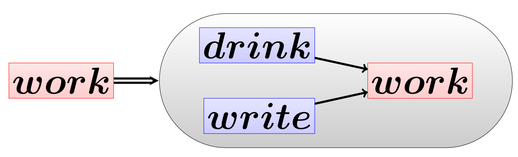
- *Primitive*, which corresponds to some concrete action we know how to perform
  - e.g: $walk(room, hall)$, or $drink(coffee)$
- *Non-primitive*, which is an abstract task. E.g. $travel(home, L.A.)$
- Must recursively decompose non-primitive tasks until we get primitive tasks we know how to execute directly
- We are given a set of *methods*, which are recipes on how to accomplish abstract tasks. E.g., to travel from $home$ to $L.A.$, we might decompose as follows:



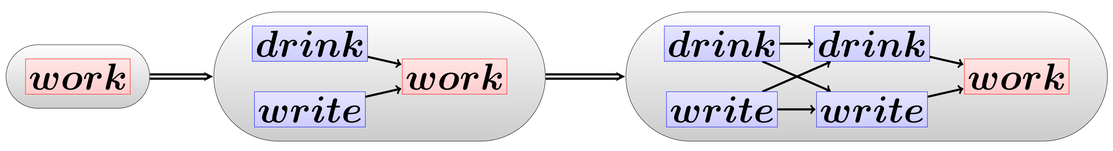
Methods and Decomposition

- A method (t, tn) is a non-primitive task t paired with a network tn

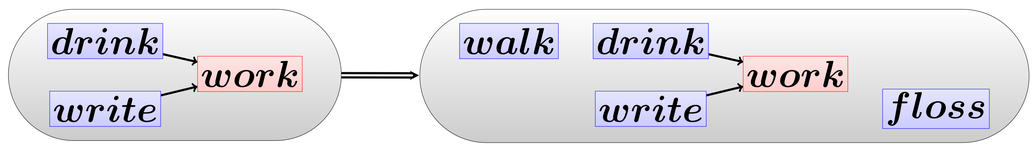
Method:



- We *decompose* a task network by replacing a node in the network with a corresponding method's network.



- An alternate set of semantics, HTN Planning with *Task Insertion* (TIHTN Planning) allows the insertion of tasks without a method.

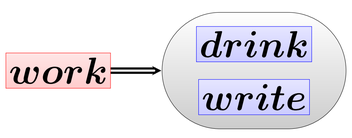


- To a great extent, we can characterize the complexity of HTN and TIHTN planning by the structure of a problem's methods: whether the methods are fully grounded, whether the methods are totally ordered, and where in the method recursion occurs.

Method structures: Acyclic

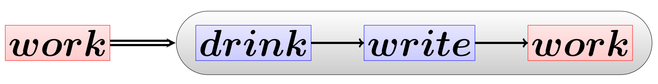
Method (totally-ordered):

Method (partially-ordered):



Method structures: Regular

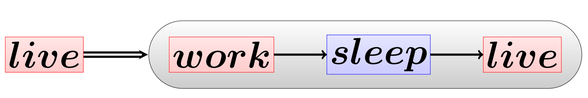
Method (totally-ordered):



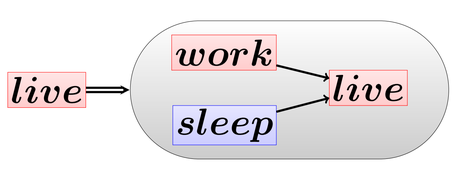
Method (partially-ordered):

Method structures: Tail Recursive

Method (totally-ordered):

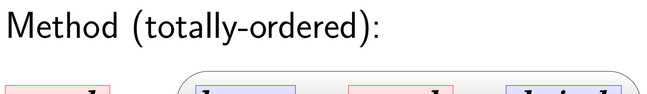


Method (partially-ordered):

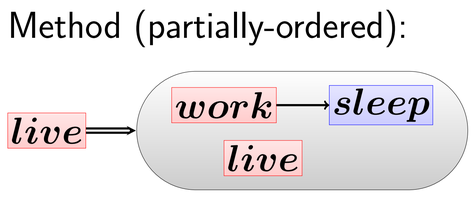


Method structures: Arbitrary

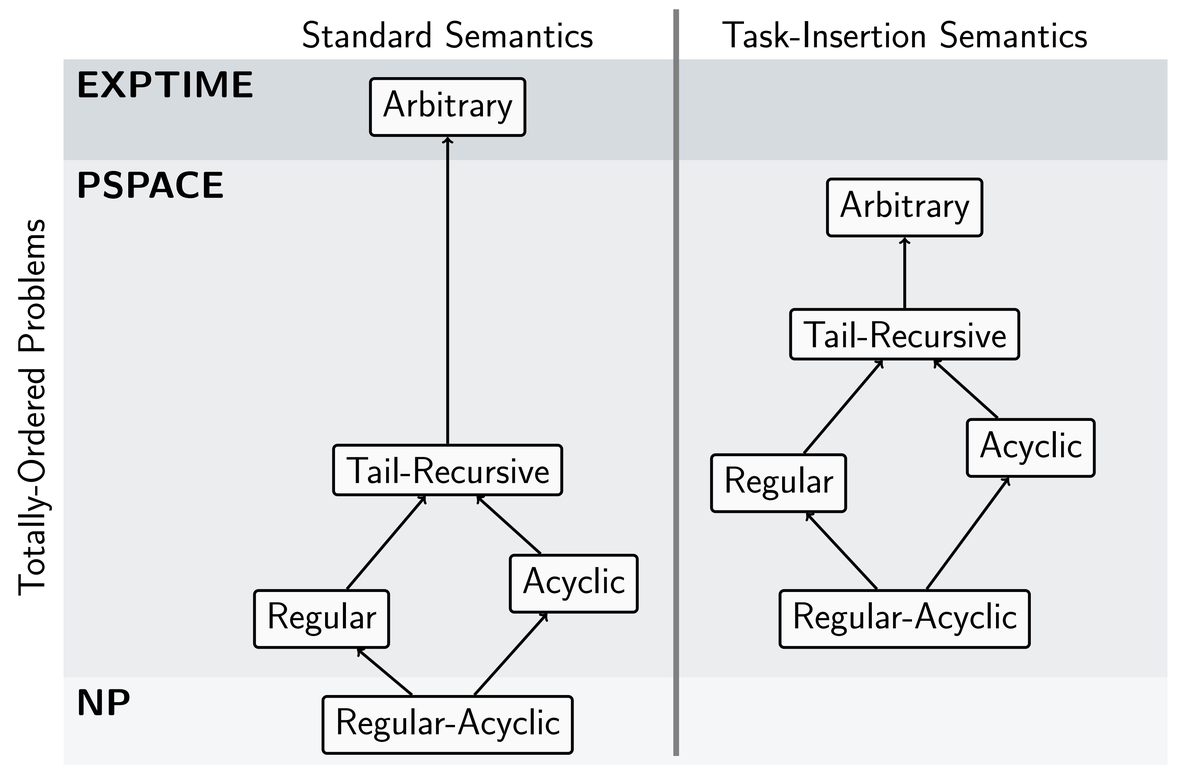
Method (totally-ordered):



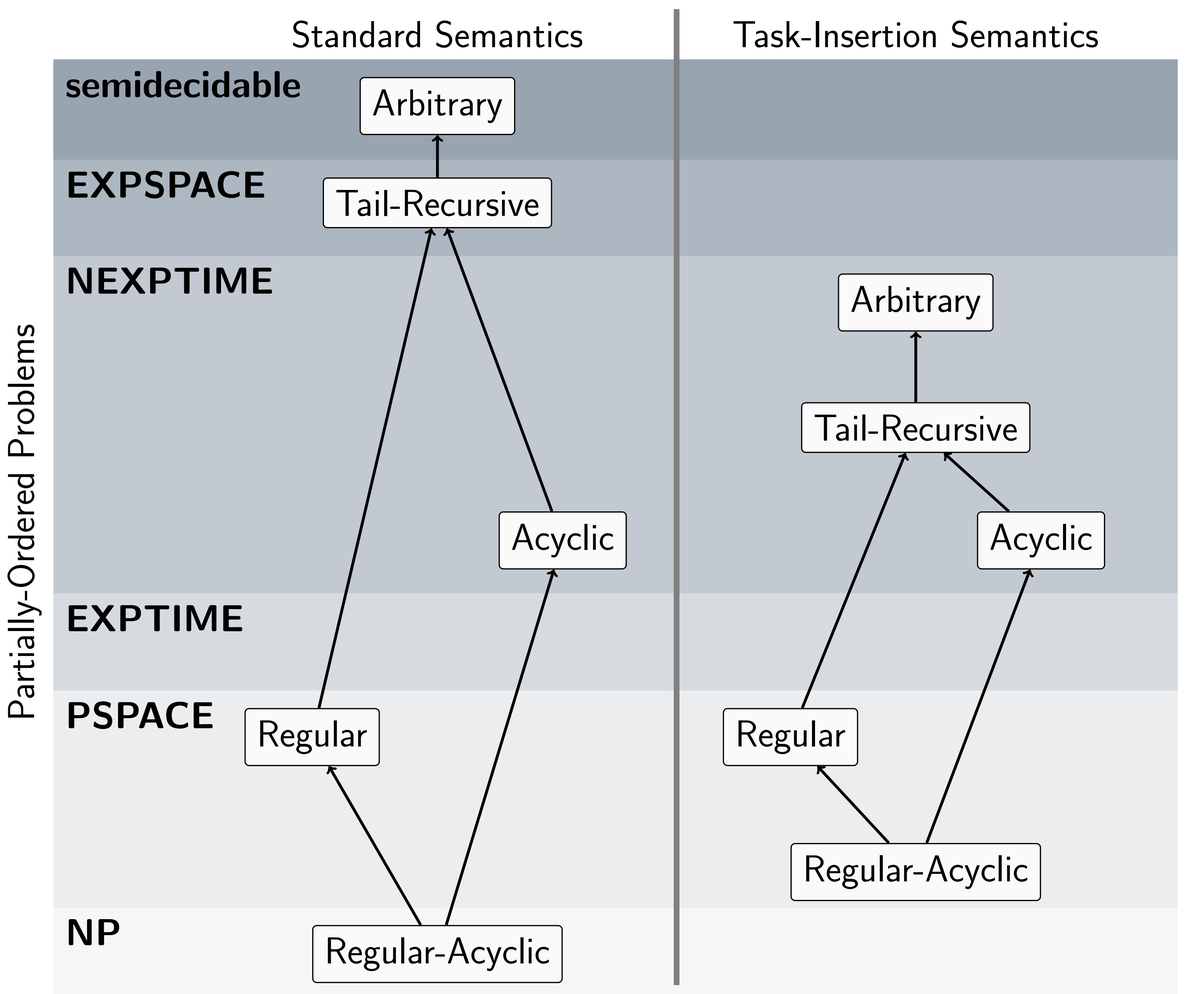
Method (partially-ordered):



Totally-Ordered Propositional HTN Complexity



Partially-Ordered Propositional HTN Complexity



Complete Results

Comparison of the complexity classes for HTN planning (completeness results) for HTN planning, with and without variables and task insertion (TI).

Vars.	Ordering	TI	Recursion	Complexity
no	total	no	acyclic	**PSPACE**
no	total	no	regular	**PSPACE**
no	total	no	tail	**PSPACE**
no	total	no	arbitrary	**EXPTIME**
no	total	yes	–	**PSPACE**
no	partial	no	acyclic	**NEXPTIME**
no	partial	no	regular	**PSPACE**
no	partial	no	tail	**EXPSPACE**
no	partial	no	arbitrary	**undecidable**
no	partial	yes	regular	**PSPACE**
no	partial	yes	–	**NEXPTIME**
yes	total	no	acyclic	**EXPSPACE**
yes	total	no	regular	**EXPSPACE**
yes	total	no	tail	**EXPSPACE**
yes	total	no	arbitrary	2-**EXPTIME**
yes	total	yes	–	**EXPSPACE**
yes	partial	no	acyclic	2-**NEXPTIME**
yes	partial	no	regular	**EXPSPACE**
yes	partial	no	tail	2-**EXPSPACE**
yes	partial	no	arbitrary	**undecidable**
yes	partial	yes	regular	**EXPSPACE**
yes	partial	yes	–	2-**NEXPTIME**

Conclusions

- Totally-ordered TIHTN planning has the same worst-case complexity as classical planning.
- Partially-ordered TIHTN planning has the same worst-case complexity as partially-ordered acyclic HTN planning (**NEXPTIME**), and is sometimes simpler

Future Work: In the paper, we provide a new planning technique for TIHTN planning, called acyclic progression, that let us define worst-case efficient TIHTN planning algorithms. Theoretical efficiency is not implementation efficiency, and so we hope to implement and evaluate acyclic progression.

ronald.alford.ctr@nrl.navy.mil pascal.bercher@uni-ulm.de david.aha@nrl.navy.mil