Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing

Ron Alford  
MITRE; McLean, VA

Vikas Shivashankar  
Knexus Research Corporation; National Harbor, MD

Mark Roberts  
NRC-NRL Postdoctoral Fellow; Washington, DC

Jeremy Frank  
NASA Ames Research Center; Moffett Field, CA

David W. Aha  
Naval Research Laboratory; Washington, DC
Hierarchical planning is the problem of decomposing an initial task or goal into a sequence of executable steps.

- Encoding control knowledge for planning
  - Agent planning in robotics and simulation: ARTUE (2010), Johnny (2012)
  - Planning and mission generation for games: KILLZONE 2, Elder Scrolls, Armed Assault
- Representing and planning with processes and hierarchies
  - Composing web services (Sirin, 2004; Tang 2013; others)
  - Program configuration (Soltani, 2012)
A Proliferation of Hierarchical Planners (and Formalisms)

- **FAPE**
  - task decomposition
  - task sharing
  - task insertion

- **SHOP2**
  - task decomposition

- **HOpGDP**
  - goal decomposition

- **PANDA**
  - task decomposition
  - task insertion

- **GoDel**
  - goal decomposition
  - task insertion
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, IJCAI)`

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 `IJCAI`, we might decompose as follows:
A method \((t, tn)\) is a non-primitive task \(t\) paired with a network \(tn\).

- Method:
  - \(\text{travel}(x, y)\)
  - \(\text{ticket}(a1, a2)\)
  - \(\text{fly}(a1, a2)\)
  - \(\text{travel}(a2, y)\)

- Method:
  - \(\text{travel}(x, y)\)
  - \(\text{taxi}(x, y)\)

- Method:
  - \(\text{travel}(x, y)\)
  - \(\text{walk}(x, y)\)

We decompose a task network by replacing a node in the network with a corresponding method’s network.

- \(\text{travel(home, IJCAI)}\)
Methods and Decomposition

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

Method:

- \(\text{fly}(a_1, a_2)\)
- \(\text{ticket}(a_1, a_2)\)
- \(\text{travel}(x, a_1)\)
- \(\text{travel}(a_2, y)\)

Method:

- \(\text{taxi}(x, y)\)
- \(\text{travel}(x, y)\)

Method:

- \(\text{walk}(x, y)\)

We decompose a task network by replacing a node in the network with a corresponding method’s network.
A method \((t, tn)\) is a non-primitive task \(t\) paired with a network \(tn\).

We decompose a task network by replacing a node in the network with a corresponding method’s network.
Methods and Decomposition

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

Method:

\[
\begin{align*}
\text{travel}(?x, ?y) & \rightarrow \text{ticket}(?a1, ?a2) \\
\text{travel}(?x, ?a1) & \rightarrow \text{fly}(?a1, ?a2) \\
\text{fly}(?a1, ?a2) & \rightarrow \text{travel}(?a2, ?y)
\end{align*}
\]

Method:

\[
\begin{align*}
\text{travel}(?x, ?y) & \rightarrow \text{taxi}(?x, ?y) \\
\text{travel}(?x, ?y) & \rightarrow \text{walk}(?x, ?y)
\end{align*}
\]

- We decompose a task network by replacing a node in the network with a corresponding method’s network.
Goal Decomposition

- A goal method \((g, tn)\) is goal paired with a network \(tn\)

Method:

We decompose a goal by prepending a node in the network with a relevant method’s network.
A goal method \((g, tn)\) is goal paired with a network \(tn\)

Method:

- \(at(\?a1)\)
- \(ticketed(\?a1, \?a2)\)
- \(at(\?a2)\)
- \(\neg ticketed(\?a1, \?a2)\)

We decompose a goal by prepending a node in the network with a relevant method’s network.
An alternate set of semantics, HTN Planning with Task Insertion (TIHTN Planning) allows the insertion of tasks without a method.

Developed by Kambhampati (1998) and Biundo (2002)

Used as a model for replanning and search with partial HTN knowledge
Task sharing allows unconstrained identical tasks to be merged

The ANML language implicitly defines task sharing

FAPE and other ANML implementations support sharing.
Reductions

- Preserves solvability
- Planners for $B$ can technically be used to solve for problems in $A$. 

**Formalism A**

**Problem P**

**Solutions for P**

**Formalism B**

**Problem P’**

**Solutions for P’**

Formalism A transforms Problem P to Problem P’ which can be solved by Formalism B. The solutions for Problem P are mapped onto the solutions for Problem P’.
Relating Hierarchical Planning Formalisms

Reductions
- Preserves solvability
- Planners for $B$ can technically be used to solve for problems in $A$.

Relaxation
- $P$ solvable implies $P'$ solvable (not vice versa)
- B’s heuristics can be used for $A$
Relating Hierarchical Planning Formalisms

Reductions
- Preserves solvability
- Planners for $B$ can technically be used to solve for problems in $A$.

Relaxation
- $P$ solvable implies $P'$ solvable (not vice versa)
- $B$’s heuristics can be used for $A$

Plan-Preserving Transformations
- Same solutions for $P$ and $P'$
- Preserves optimality
Plan-Preserving Transformations

Tasks

Goals + Tasks

Tasks + Sharing

Goals

Goals + Tasks + Insertion

Tasks + Insertion

Goals + Tasks + Insertion + Sharing

Tasks + Insertion + Sharing

Goals + Insertion

semidecidable

NEXPTIME

???

PSPACE
Using goal-decomposition planners for task planning:
Going against the grain

Using goal-decomposition planners for task planning:

- First, write a task planner

Task-decomposition planner
Using goal-decomposition planners for task planning:

- First, write a task planner
- Compile planner into Turing machine (TM)
Using goal-decomposition planners for task planning:
- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)
Using goal-decomposition planners for task planning:

- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)
- Translate PCP into intersection of CFGs
Going against the grain

Using goal-decomposition planners for task planning:
- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)
- Translate PCP into intersection of CFGs
- Encode CFGs as goal methods

Diagram:
- Task-decomposition planner
  - Turing machine
  - Post correspondence problem
  - Intersection of CFGs
  - Goal-decomposition problem
Conclusions

Contributions:

- Combined formalism for goal and task decomposition
- Formal semantics for task sharing
- Formal relationships between sets of semantics

Conclusion: The semantics of HTN planning are both simple and general. Why use anything else?

- Classical heuristics can be easily adapted for goal-decomposition
- Task insertion allows the use of more efficient (terminating) algorithms
- Task sharing may allow even more efficient algorithms when combined with task insertion (future work)

Future work: Practice still has to catch up to theory (more admissible heuristics, optimal and suboptimal planning implementations, etc)