Was Fixing this *Really* That Hard? On the Complexity of Correcting HTN Domains

Songtuan Lin & Pascal Bercher
School of Computing, The Australian National University, Canberra, Australia
{songtuan.lin, pascal.bercher}@anu.edu.au

Introduction

Motivation : The task of modeling planning domains is a major obstacle for deploying planning techniques more broadly.

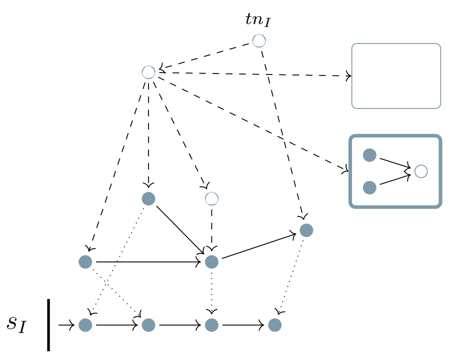
- ✎ Tools for modeling assistance are demanded!
- ✎ Planning. Domains, itSIMPLE, etc.

Objective : We want to study the complexity of deciding whether a set of *white* list plans and of *black* list plans can be respectively turned into *solutions* and *non-solutions* by changing an HTN planning domain.

- ✎ We will identify all hardness sources that make this problem $\mathbb{NP}$-hard.

HTN Planning

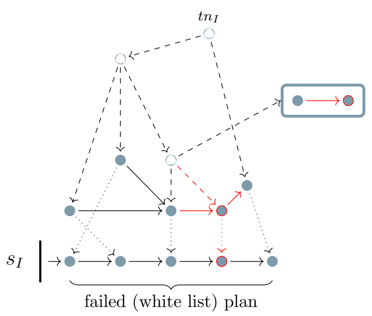
HTN Planning is to keep decomposing so-called *compound* tasks until a primitive action sequence called a plan is obtained.



- ✎ A compound task is decomposed (refined) into a *task network* by a *method*.
- ✎ A task network is a partial order set of compound and primitive tasks.
- ✎ A plan is a solution if it is an executable linearization of a primitive task network obtained by decomposing the initial task network.

Preview of the Results

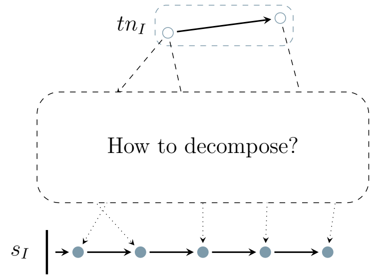
We are concerned with correcting (changing) the domain of an HTN planning problem:



- ✎ **Q1**: How hard is it to turn the failed white list plan into a solution?
  - ✏ It is **NP**-complete, and there are three sources each of which results in **NP**-hardness.
- ✎ **Q2**: What if we consider both white list and black list plans?
  - ✏ It is **NP**-hard, but it is in Σ_2^p.

Hardness Source #1

It is **NP**-complete to find a decomposition hierarchy that results in a given plan.

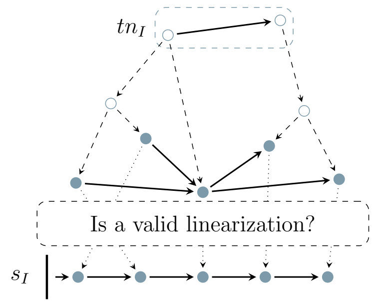


Inputs: An HTN planning problem $\mathcal{P}$.
A set Π of white list plans.

Output: True if each plan in Π can be turned into a solution by changing the domain of $\mathcal{P}$, otherwise false.

Hardness Source #2

It is **NP**-complete to decide whether a plan is a valid linearization of a primitive task network.



Inputs: An HTN planning problem $\mathcal{P}$.
A set Π of white list plans.
A set Θ of decomposition hierarchies.

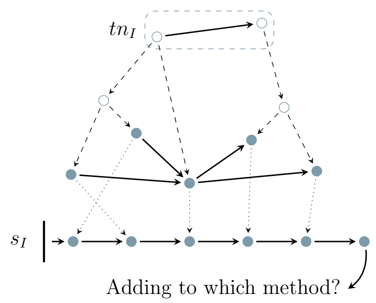
Output: True if we can change the domain of $\mathcal{P}$ to turn each plan in Π into a solution, witnessed by a decomposition hierarchy in Θ, otherwise false.

Why this result is of great importance in practice?

- ✎ It serves as a fundamental hardness source for many problems involving reasoning over partial order.
  - ✏ E.g., POCL planning, the PlanVerification problem, and the PlanCompatibility problem.

Hardness Source #3

It is **NP**-complete to find the method to which an action should be added.

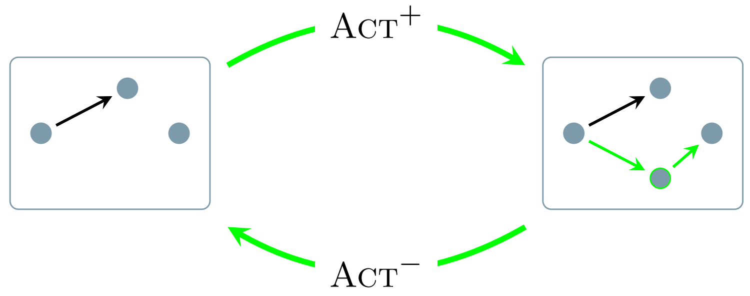


Inputs: A set Π of plans.
An HTN planning problem $\mathcal{P}$.
A set Ω of decomposition hierarchies.
A set Φ of mappings ϱ from the result of a $g \in \Omega$ to the actions in a $\pi \in \Pi$.

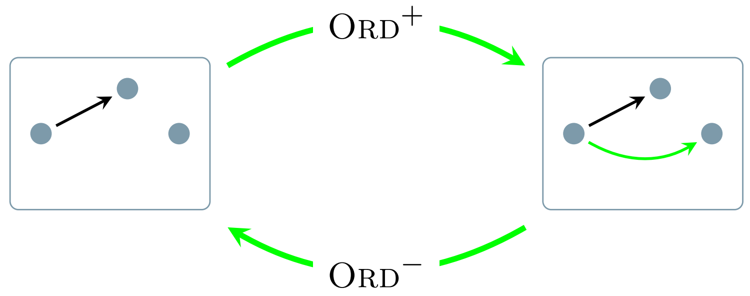
Output:True if we can change the domain of $\mathcal{P}$ to turn each plan $\pi \in \Pi$ into a solution, witnessed by a decomposition hierarchy $g \in \Omega$ and a mapping $\varrho \in \Phi$, otherwise false.

Model Change Operations

- ✎ Act$^+$/Act$^-$: Adding/removing an action to/from a method together with the respective ordering constraints.



- ✎ Ord$^+$/Ord$^-$: Adding/removing an ordering constraint over *actions* to/from a method.



Exploiting the Hardness Sources

We can adapt existing approaches for the hardness sources we identified to develop a method for turning white plans into solutions:

- ✎ Coping with the hardness source #1:
  - ✏ Adapting the existing SAT encoding for searching for a decomposition hierarchy.
- ✎ Coping with the hardness source #3:
  - ✏ Extending the above SAT encoding to incorporate model changes, e.g., we can use a varible x_m^a to indicate whether an action a is added to the method m.
- ✎ Coping with the hardness source #2:
  - ✏ Adapting the SAT encoding for the (sub)graph isomorphism problem.

Given White and Black List Plans

When given both white list and black list plans, the problem is **NP**-hard but in the class Σ_2^p.

- ✎ It generalizes the case where only white list plans are given.
  - ✏ **NP**-hardness follows.
- ✎ We can assume an oracle machine that verifies whether a plan is a solution to an HTN planning problem and develop a non-deterministic poly-time algorithm based on it.
  - ✏ Σ_2^p-membership follows.