Exploiting Solution Order Graphs and Path Decomposition Trees for More Efficient HTN Plan Verification via SAT Solving

Songtuan Lin[1], **Gregor Behnke**[2], **Pascal Bercher**[1]

[1]**School of Computing, The Australian National University, Canberra, Australia**

[2] **ILLC, University of Amsterdam, Amsterdam, The Netherlands**

[1]**{songtuan.lin, pascal.bercher}@anu.edu.au**, [2]**g.behnke@uva.nl**

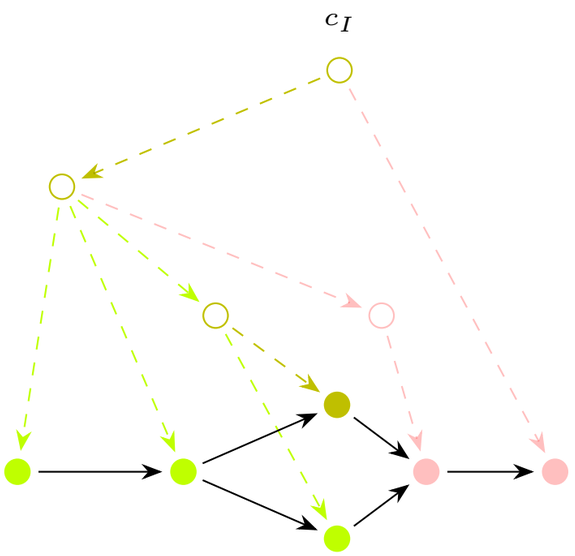
Introduction

The objective of this paper is to:

- develop a SAT-based HTN plan verification approach by exploiting the data structures Path Decomposition Trees (PDTs) and Solution Order Graphs (SOGs) employed in the SOTA SAT-based HTN planner (we call this the SOG-based approach) and
- reimplement the existing SAT-based HTN plan verification approach relying on Decomposition Trees (DTs) in C++ (we call this the DT-based approach).

HTN Planning

An HTN planning problem is $\mathcal{P} = ((\mathcal{F}, \mathcal{C}, \mathcal{A}, \delta, \mathcal{M}), c_I, s_I)$ where • $\mathcal{F}$ is a set of propositions, • $\mathcal{C}$ is a set of compound tasks, • $\mathcal{A}$ is a set of primitive tasks (actions), • $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ is a function mapping an action to its precondition and effects, • $\mathcal{M}$ is a set of methods decomposing compound tasks into partial order sets of tasks called task networks, • c_I is the initial compound task, and • $s_I \in 2^{\mathcal{F}}$ is the initial state.



Decomposition Trees: A decomposition tree is a tree-representation of how compound tasks are decomposed by methods. For instance, the trees on the left colored with lime and pink respectively represent a decomposition tree.

Path Decomposition Trees: A path decomposition tree is a compact representation of **all** possible decomposition trees. For instance, the tree on the left is a path decomposition tree which contains two decomposition trees (Notice that there are overlapping nodes in the DTs stored in the PDT).

Solution Order Graphs: A solution order graph is essentially the leaves of a PDT constructed in an order consistent manner. For instance, the leaves of the tree on the left is a SOG.

A plan (i.e., an action sequence) is a solution if and only if it is executable in the initial state and there exists a decomposition tree which decomposes the initial compound task into a task network such that the plan is a linearization of this task network.

Future Work

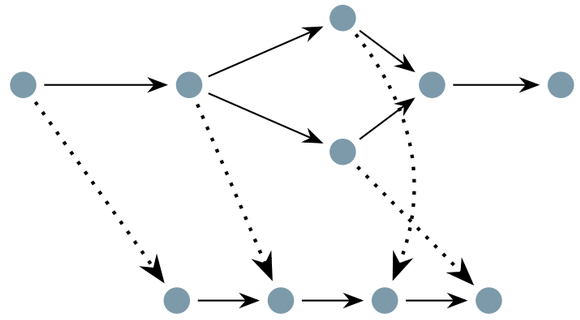
There are several extensions and optimizations that we want to do in the future work:

- We want to extend both two approaches to support method preconditions.
- We want to eliminate all actions in a given HTN planning problem that are **not** in the given plan together with all methods and compound tasks which could lead to them.
- We want to calculate a tight bound for the maximal depth of a decomposition tree which results in the given plan.

SOG-based Approach

Core Idea: The core idea of the SOG-based approach is to use a SAT formula to encode the constraint that there exists a DT in a PDT of which the plan is a linearization.

Our Objective: We only need to construct the SAT clauses encoding that there exists a subset of the vertices of the SOG which has a bijective mapping to the plan because the remaining clauses for encoding, e.g., the construction of the PDT and the constraint that the selected subset of the vertices must be the leaves of a DT, have already been given in the SAT-based HTN planner.



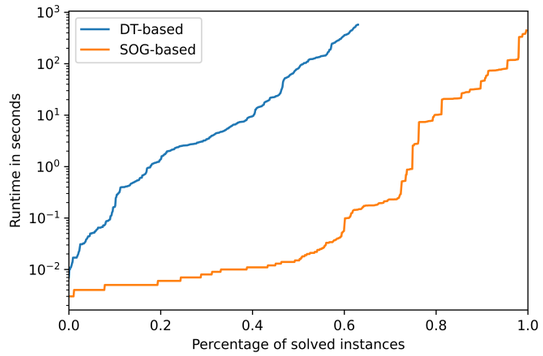
- If a vertex in the SOG is mapped to an action in the plan, then the vertex must be labeled with the same action.
- A vertex in the SOG is activated if and only if it is mapped to an action in the plan.
- For each action in the plan, there exists exact one vertex in the SOG which is mapped to it.
- Every vertex can be mapped to at most one action in the plan.
- The mapping must respect the ordering constraints defined by the edges of the SOG.

Example: Let a be an action in the plan and v a vertex in the SOG. We define the proposition variable m_v^a which is set to *true* if v is mapped to a. We thus have $m_v^a \to v_a$ where the proposition variable v_a is set to *true* if the vertex v is labeled with the action a.

Empirical Evaluation

We compared our SOG-based approach with the reimplemented DT-based approach on the benchmark set from the IPC 2020 on HTN Planning. The following table lists the number of instances from each domain in the benchmark set solved by the SOG-based and the DT-based approach:

	Transport	Woodworking	UM-Translog	Satellite	Monroe-Partially-Observable	PCP	Monroe-Fully-Observable
Total Instances	188	137	52	246	103	26	129
SOG-based	188 (100.00%)	137 (100.00%)	52 (100.00%)	246 (100.00%)	102 (99.03%)	26 (100.00%)	128 (99.22%)
DT-based	138 (73.40%)	95 (69.34%)	52 (100.00%)	246 (100.00%)	0 (0.00%)	25 (96.15%)	0 (0.00%)



The figure on the left depicts the runtimes against the percentages of solved instances for both the SOG-based and the DT-based approach. We can see that the SOG-based approach significantly outperforms the DT-based one. This is because the DT-based approach also relies on finding a decomposition tree which results in a given plan, but it does **not** store decomposition trees in a compact way, which cause a significant overhead.