Accelerating SAT-Based HTN Plan Verification by Exploiting Data Structures from HTN Planning

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

[1]School of Computing, The Australian National University
[2]Institute for Logic, Language, and Computation, University of Amsterdam

Oct 2023

Introduction

Plan verification for hierarchical planning is hard!

- It is **NP**-complete in general.

Contribution

We developed a SAT-based HTN plan verifier by adapting data structures for *solving* HTN problems which can verify plans for *partial order* (PO) HTN problems more efficient.

HTN Planning

- c_I

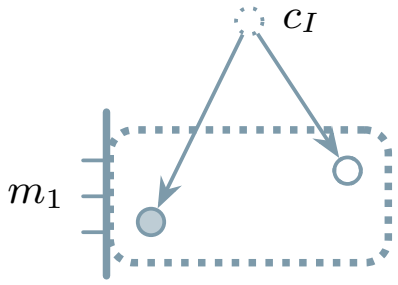
A domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

A problem $\mathcal{P} = (\mathcal{D}, c_I, s_I)$:

- c_I: The initial task.
- s_I: The initial state.

HTN Planning



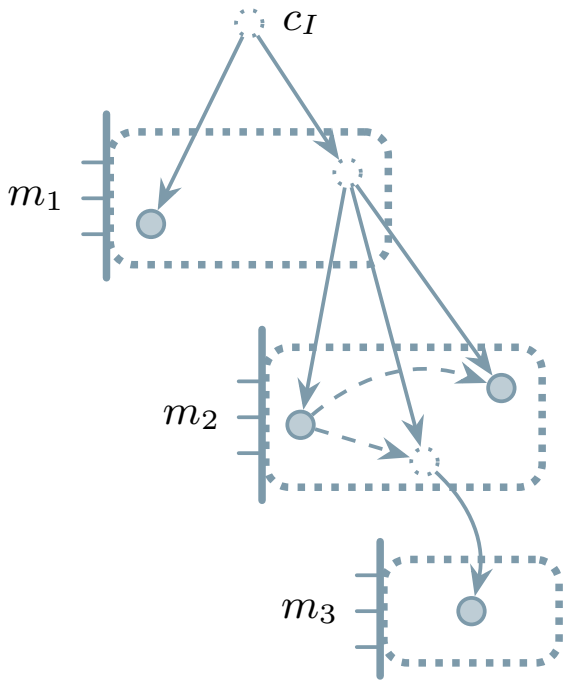
A domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

A problem $\mathcal{P} = (\mathcal{D}, c_I, s_I)$:

- c_I: The initial task.
- s_I: The initial state.

HTN Planning



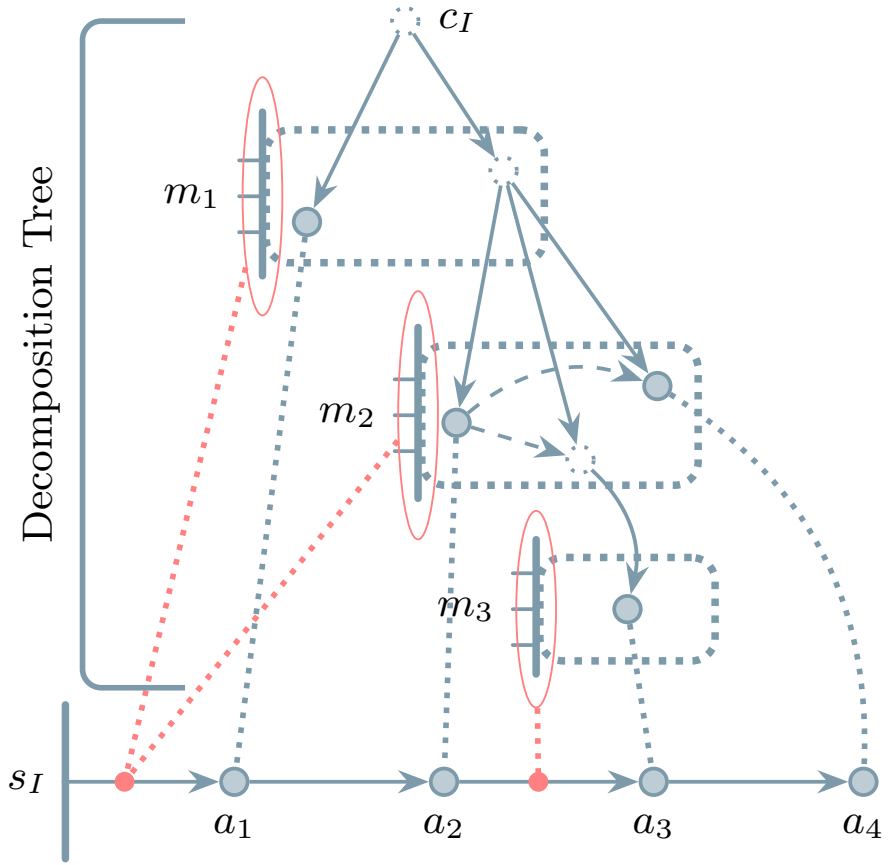
A domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

A problem $\mathcal{P} = (\mathcal{D}, c_I, s_I)$:

- c_I: The initial task.
- s_I: The initial state.

HTN Planning



A domain $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \alpha)$:

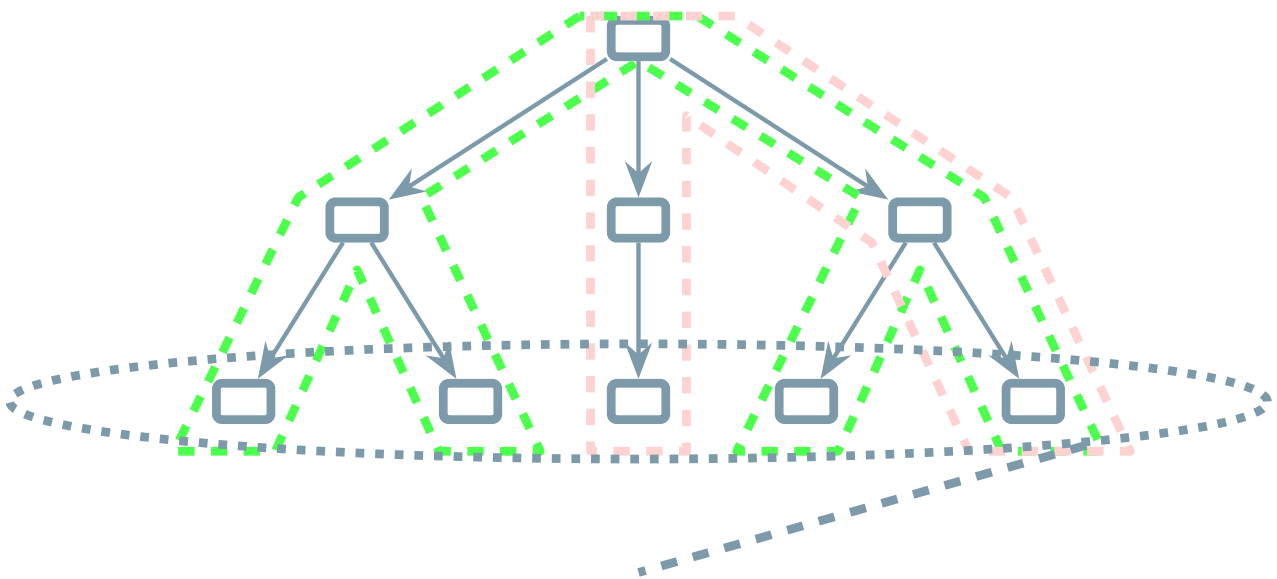
- $\mathcal{F}$: A set of propositions.
- $\mathcal{A}$: A set of primitive tasks.
- $\mathcal{C}$: A set of compound tasks.
- $\mathcal{M}$: A set of methods.
- $\alpha : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

A problem $\mathcal{P} = (\mathcal{D}, c_I, s_I)$:

- c_I: The initial task.
- s_I: The initial state.

PDTs and SOGs

- Path Decomposition Trees (PDTs)
  - A PDT of certain height K stores compactly *all* valid decomposition trees of height up to K.
- Solution Order Graphs (SOGs)
  - A SOG stores the *yields* of all decomposition trees in a PDT of certain height.



Solution Order Graph

Procedure

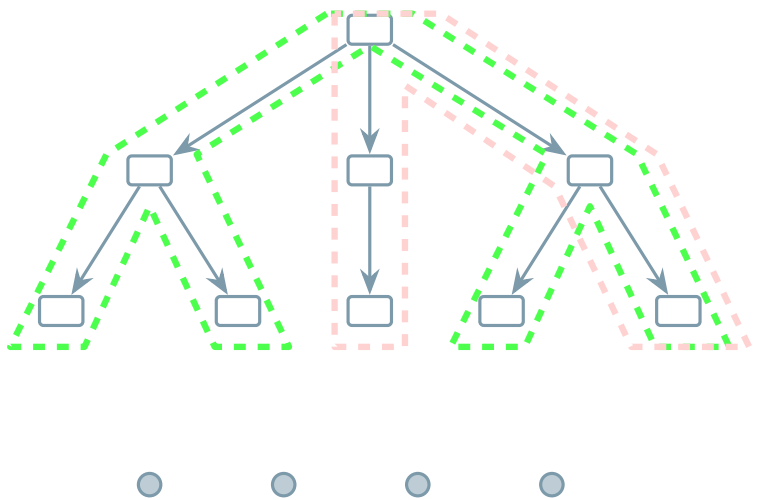
Main Procedure

- Starting with $K = 1$, we build a PDT of height K.
- We check whether there exists a decomposition tree in the constructed PDT which can lead to the plan and all method preconditions are satisfied (by SAT).
- If that is not the case, we increase K by 1 and repeat the previous steps.

- A method precondition can be compiled as a primitive task that is ordered before all the remaining tasks in the method.

Encoding

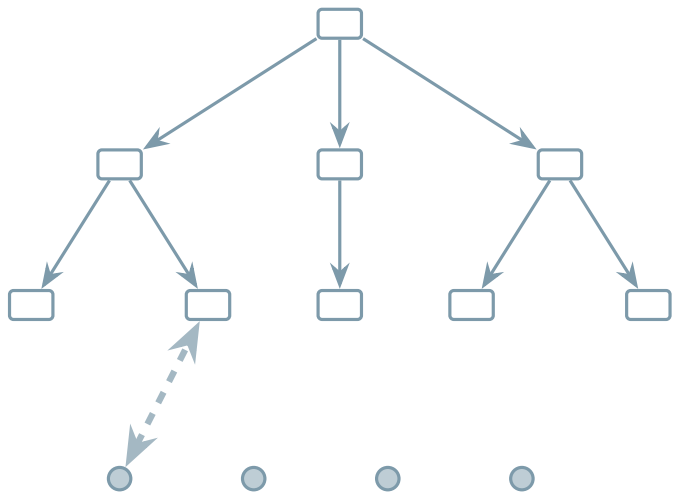
Encoding for constructing a PDT already exists.



Encoding

Encoding for constructing a PDT already exists.

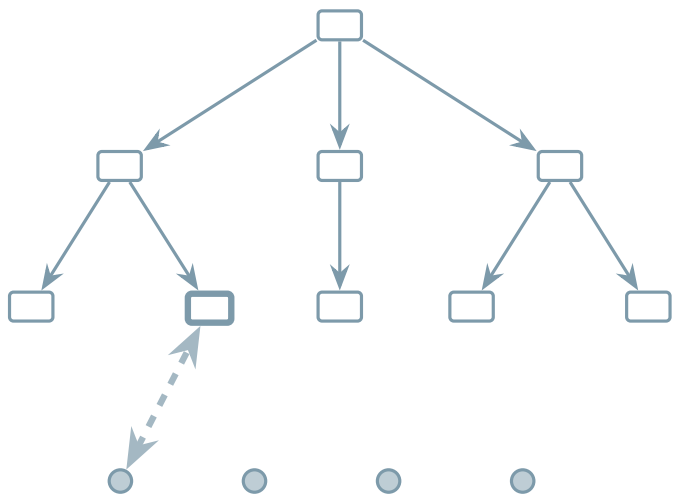
- Use a variable to represent the mapping between an action in the plan and a vertex in the SOG.



Encoding

Encoding for constructing a PDT already exists.

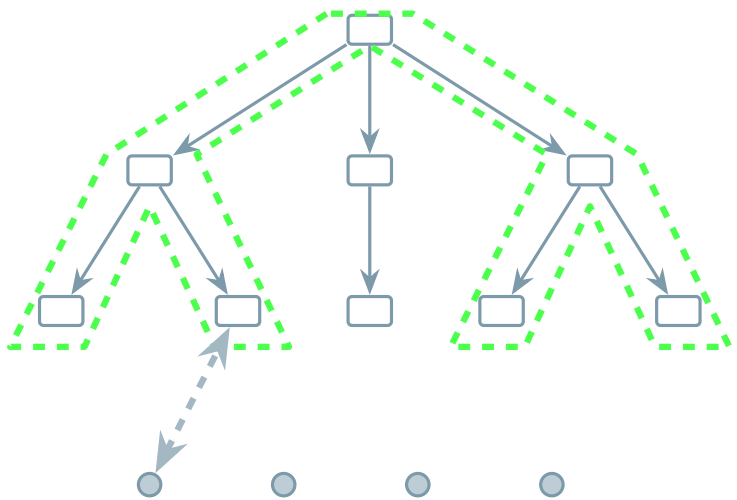
- Use a variable to represent the mapping between an action in the plan and a vertex in the SOG.
- A vertex is activated if and only if it is mapped to an action.



Encoding

Encoding for constructing a PDT already exists.

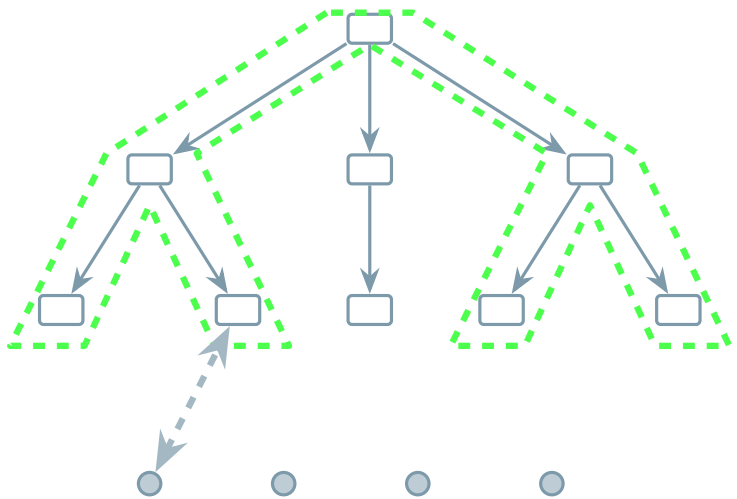
- Use a variable to represent the mapping between an action in the plan and a vertex in the SOG.
- A vertex is activated if and only if it is mapped to an action.
- If a vertex is activated, the decomposition tree to which the vertex belongs must also be activated.



Encoding

Encoding for constructing a PDT already exists.

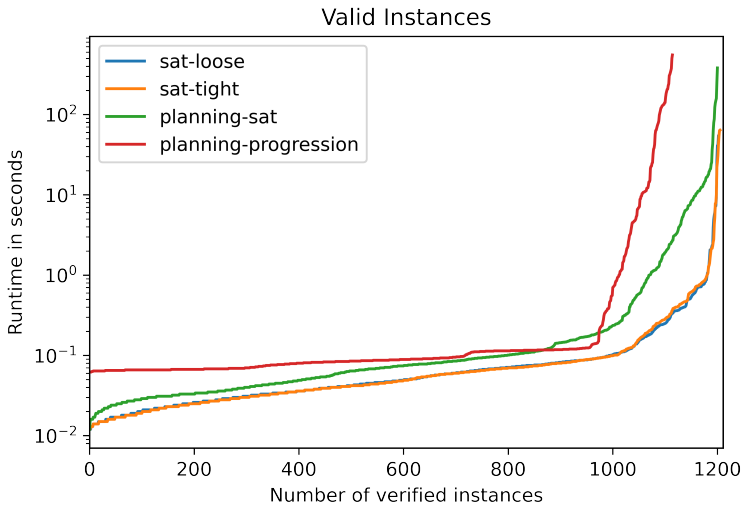
- Use a variable to represent the mapping between an action in the plan and a vertex in the SOG.
- A vertex is activated if and only if it is mapped to an action.
- If a vertex is activated, the decomposition tree to which the vertex belongs must also be activated.
- Every activated vertex should be mapped to an action (and vice versa).

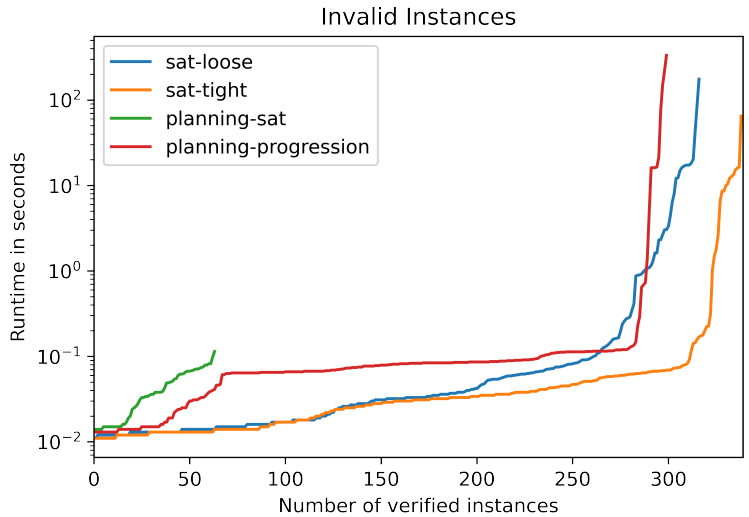


Configurations

- The benchmarks are plans generated by planners that participated in the IPC 2020 on HTN Planning.
  - The set contains 1211 valid plans and 138 invalid plans.
- We manually added additional invalid plans to the benchmark set.
  - Most of the original invalid plans fail because of some simple reasons.
- We compared our new SAT-based verifier against the current SOTA one based on planning.
  - It has been shown that the planning-based verifier outperforms all other existing ones.

Experimental Results





	Instances	SAT (Ours)		Planning	
		Loose	Tight	SAT	Progression
Satellite	269	**269**	**269**	**269**	**269**
Transport	219	**219**	**219**	**219**	141
Rover	171	**171**	**171**	**171**	163
Woodworking	162	**162**	**162**	**162**	**162**
Monroe (FO)	130	**130**	**130**	**130**	**130**
Monroe (PO)	103	**103**	**103**	**103**	**103**
Barman-BDI	68	**63**	**63**	58	58
UM-Translog	57	**57**	**57**	**57**	**57**
PCP	31	**31**	**31**	**31**	**31**
Zenotravel	1	**1**	**1**	**1**	**1**
	1211	**1206**	**1206**	1201	1115

	Instances	SAT (Ours)		Planning	
		Loose	Tight	SAT	Progression
Satellite	66	**66**	**66**	23	**66**
Transport	64	54	**64**	18	33
UM-Translog	59	**59**	**59**	14	**59**
Rover	53	41	**53**	6	47
Monroe (FO)	24	**24**	**24**	1	**24**
Woodworking	21	**21**	**21**	0	**21**
Barman-BDI	18	**18**	**18**	0	16
Monroe (PO)	18	**18**	**18**	2	**18**
PCP	12	**12**	**12**	0	**12**
Zenotravel	4	**4**	**4**	0	**4**
	339	317	**339**	64	300

Conclusion

In this paper, we developed a new SAT-based HTN plan verifier.

- It exploits PDTs and SOGs from the SAT-based HTN problem solver.
- It outperforms the SOTA planning-based verifier in verifying plans for POHTN problems.