On Total-Order HTN Plan Verification with Method Preconditions – An Extension of the CYK Parsing Algorithm

Songtuan Lin[1], Gregor Behnke[2], Simona Ondrčková[3], Roman Barták[3], Pascal Bercher[1]

[1]School of Computing, The Australian National University
[2]ILLC, University of Amsterdam
[3]Faculty of Mathematics and Physics, Charles University
[1]{songtuan.lin, pascal.bercher}@anu.edu.au, [2]g.behnke@uva.nl,
[3]{ondrckova, bartak}@ktiml.mff.cuni.cz

February 2023

Australian National University

Motivation and Objective

- The TOHTN instances in the IPC 2020 on HTN planning significantly outnumber the partial order ones.
- Verification is important to test the correctness of HTN systems. Three approaches exist so far.
  - One appraoch has special treatment for total order HTNs.
  - It relies on parsing and shows promising results!

Objective

Speed up TO-HTN plan verification.

- Approach should support method preconditions.

(We do so by extending the CYK parsing algorithm.)

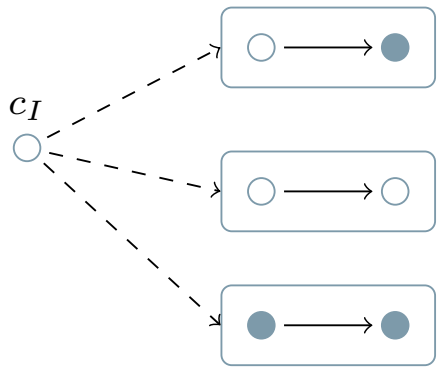
Background

A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:

- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

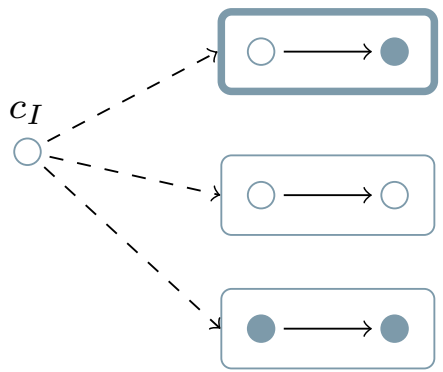
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

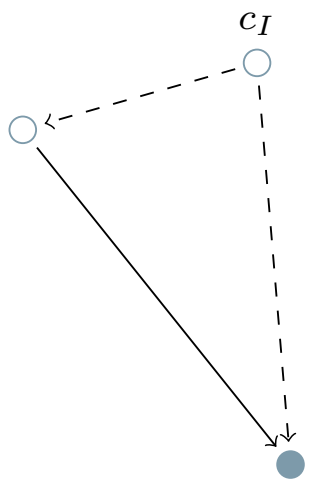
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

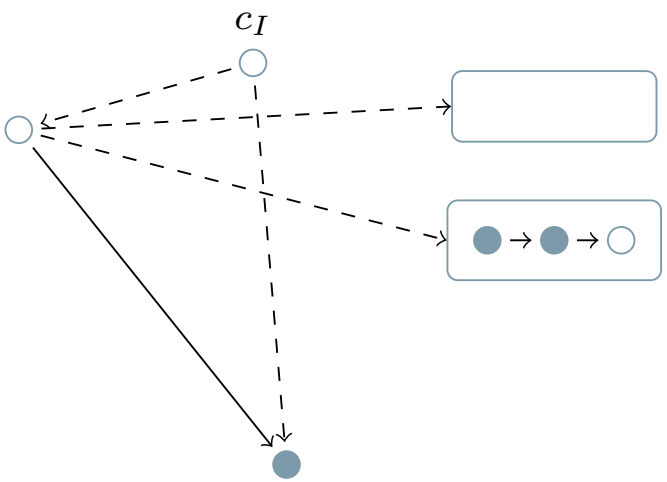
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

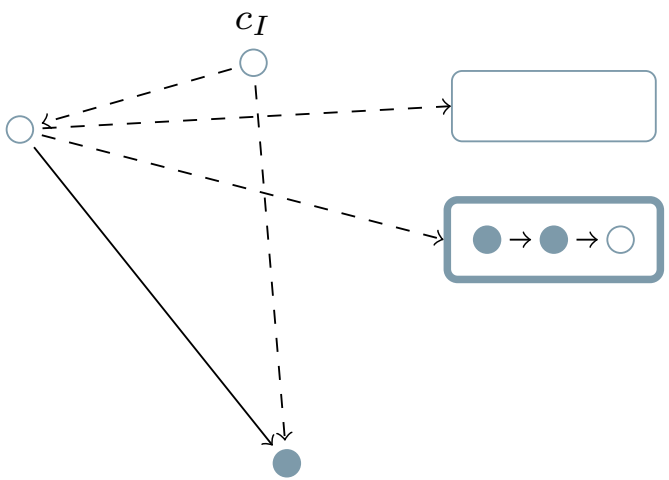
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

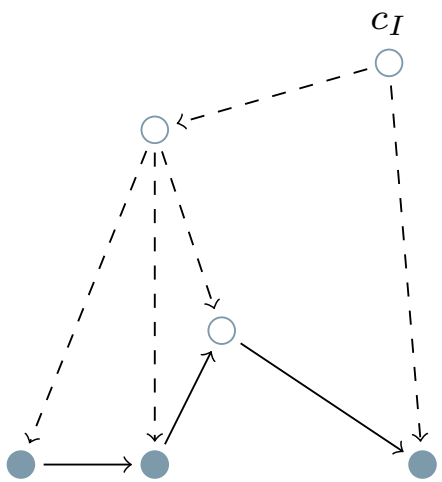
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

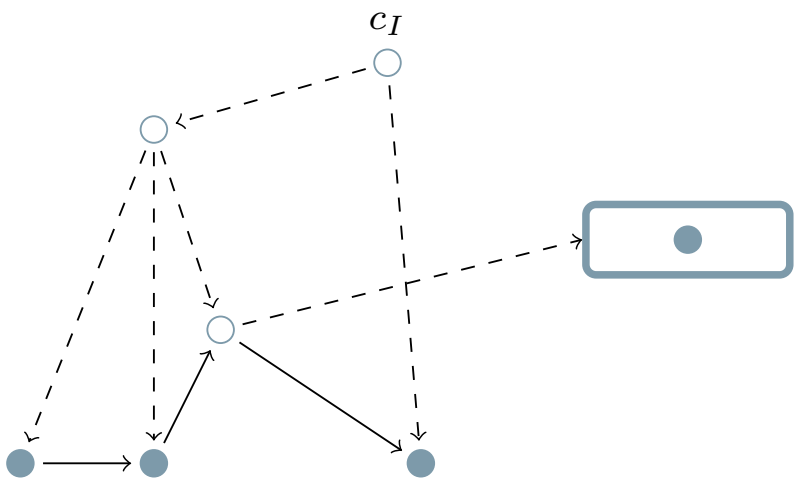
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

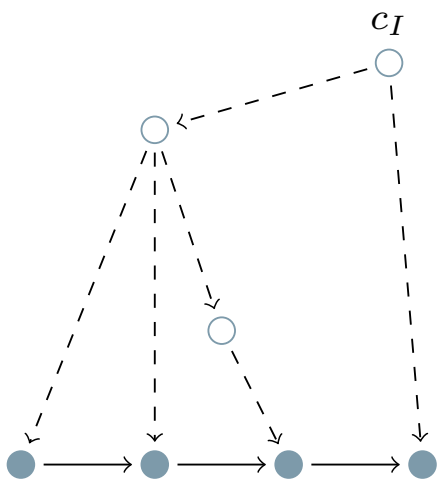
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

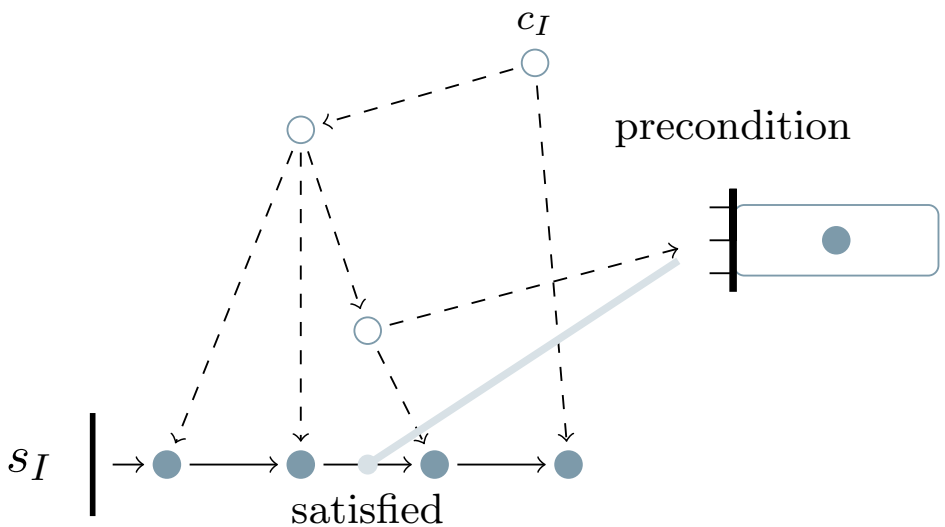
A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Background

A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

TOHTN Planning Problems and CFGs

The basis for using the CYK algorithm in plan verification is the connection between a TOHTN planning problem and a context-free grammar:

- A primitive task is equivalent to a terminal symbol.
- A compound task is equivalent to a non-terminal symbol.
- A method *without preconditions* is equivalent to a production rule.
- The set of solutions to a TOHTN problem is the language of the grammar.

Procedure

Given a plan $\pi = \langle a_1 \cdots a_n \rangle$ (*resp.* a word) and a TOHTN planning problem $\mathcal{P}$ (*resp.* a CFG):

- $\mathcal{P}$ should be in Chomsky Normal Form.
  - Every method decomposes a compound task into either a primitive task or two compound tasks.
- For every $\pi_{i,j} = \langle a_i \cdots a_j \rangle$, we memorize the set $A[i,j]$ of all compound tasks that can be decomposed into $\pi_{i,j}$:

$$A[i,j] = \begin{cases} \{c \mid c \to \langle p_i \rangle\} & \text{if } i = j \\ \left\{ c \;\middle|\; \begin{array}{l} c \to \langle c'_1\, c'_2 \rangle, c'_1 \in A[i,k] \\ c'_2 \in A[k+1,j], i \leq k < j \end{array} \right\} & \text{if } i < j \end{cases}$$

i.e., when $i < j$, we add a task c to the entry $A[i,j]$ if c can be decomposed into $\langle c'_1\, c'_2 \rangle$ such that c'_1 can be decomposed into $\pi_{i,k}$ and c'_2 into $\pi_{k+1,j}$ for some $i \leq k < j$.

TOHTN Plan Verification

Modification #1

Instead of transforming a TOHTN problem into CNF, we transform it into 2-Norm Form (2NF).

- Every decomposition method decomposes a compound task into *at most* two subtasks (and they can be mixed primitive/compound).

TOHTN Plan Verification

Modification #1

Instead of transforming a TOHTN problem into CNF, we transform it into 2-Norm Form (2NF).

- Every decomposition method decomposes a compound task into *at most* two subtasks (and they can be mixed primitive/compound).

Using 2NF can avoid the quadratic explosion of the problem size caused by transforming the problem into CNF.

- E.g., consider a unit production sequence $c_1 \to \cdots \to c_n$, and c_n can be decomposed into any of $\langle a_1\, b_1 \rangle, \cdots, \langle a_m\, b_m \rangle$.
  - For each c_i, we need to create m extra methods which decomposes c_i into $\langle a_j\, b_j \rangle$, $1 \leq j \leq m$.

TOHTN Plan Verification

Modification #1

Instead of transforming a TOHTN problem into CNF, we transform it into 2-Norm Form (2NF).

- Every decomposition method decomposes a compound task into *at most* two subtasks (and they can be mixed primitive/compound).

Using 2NF can avoid the quadratic explosion of the problem size caused by transforming the problem into CNF.

- E.g., consider a unit production sequence $c_1 \to \cdots \to c_n$, and c_n can be decomposed into any of $\langle a_1\, b_1 \rangle, \cdots, \langle a_m\, b_m \rangle$.
  - For each c_i, we need to create m extra methods which decomposes c_i into $\langle a_j\, b_j \rangle$, $1 \leq j \leq m$.
- When computing each cell $A[i, j]$, we need to search for all method sequences $\overline{m}$ with $c \to^*_{\overline{m}} \langle c' \rangle$ with $c' \in A[i, j]$.

TOHTN Plan Verification

Modification #2

We have to check whether the precondition $prec(m)$ of a method m is satisfied.

Let $\pi = \langle a_1 \cdots a_n \rangle$ be the given plan.

- We can compute the state sequence $\langle s_0 \cdots s_n \rangle$ with $s_0 = s_I$, which is obtained by applying π in s_I.
- When checking whether a task c can be decomposed into $\pi_{i,j}$ via a method m, we check whether $prec(m) \subseteq s_i$.

Experiment Configuration

We compare our approach against three existing plan verification approaches:

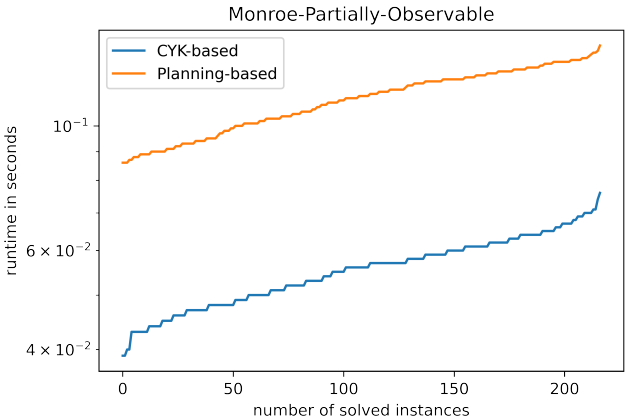
- Another parsing-based approach.
  - Builds on the Rete parser.
  - It does not exploit any normal forms.
  - Supports lifted input.
- A planning-based approach (compiles verification problem into HTN planning problem).
  - Can use existing HTN planning heuristics
  - Usually not a decision procedure
- A SAT-based approach (compiles it into SAT problem).
  - It encodes the plan verification problem as a SAT formula.
  - Does act as decision procedure.
  - Does not support method preconditions.

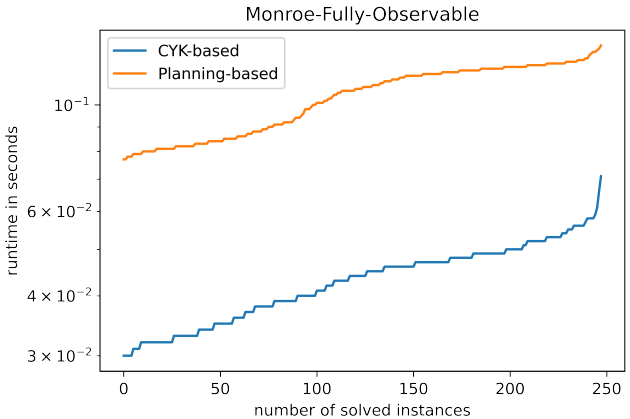
Experiment Results

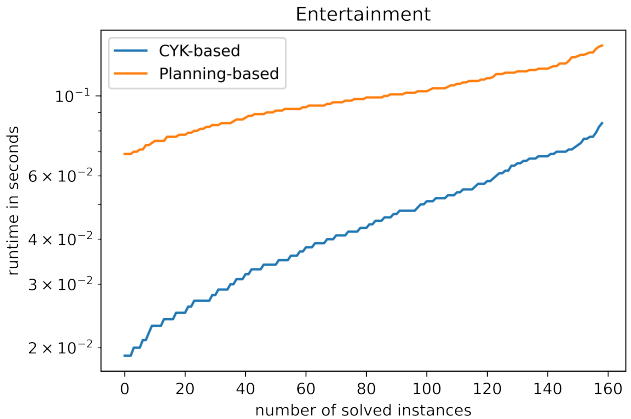
Benchmark	Instances	Parsing-based	Planning-based	SAT-based	CYK-based (Ours)
to-val	10961	9158 (83.55%)	**10925** (**99.67%**)	Not support	10917 (99.60%)
to-inval	1406	1301 (92.53%)	1364 (97.01%)	Not support	**1406** (**100.00%**)
to-val-no-mprec	11304	7889 (69.79%)	9679 (85.62%)	1036 (9.16%)	**9946** (**87.99%**)
to-inval-no-mprec	1063	915 (86.08%)	898 (84.48%)	684 (62.01%)	**981** (**88.94%**)

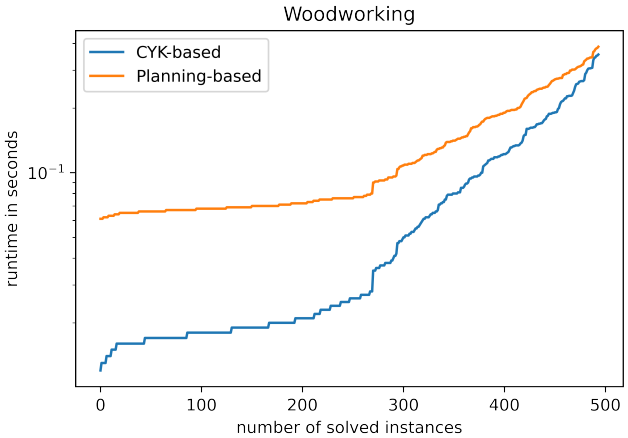
- Our approach outperforms the parsing-based one and the SAT-based one.
- Our approach slightly underperforms the planning-based one in verifying valid plans with method preconditions but is better in all the remaining cases.
  - This might because the planning-based approach exploits some heuristic which performs well in verifying long plans.
  - Such a heuristic is less useful when the transformed planning problem is unsolvable.
  - We believe that our approach acts as a better decision procedure compared with the planning-based one.

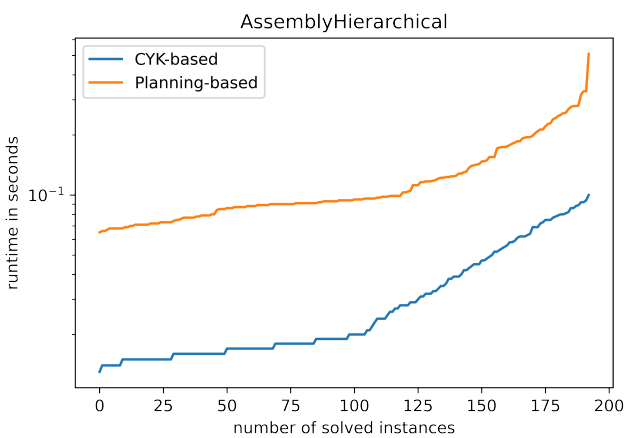
Runtime

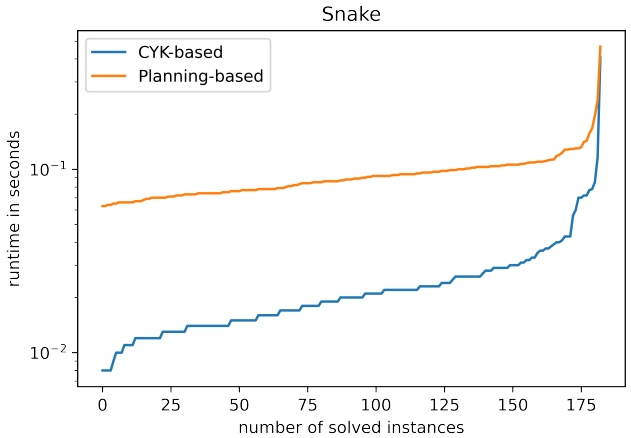












Runtime against solved instances (with method preconditions), planning-based approach vs. ours:
Runtime is significanly lower for our apporach.

Conclusion

We developed a CYK-based TOHTN plan verification method.

- Our approach outperforms the existing SAT-based and parsing-based approaches.
- Although our approach slightly underperforms the planning-based one in verifying valid plans with method preconditions (8 instances of ≈11.000), we believe it still acts as a better decision procedure.