Tight Bounds for Lifted HTN Plan Verification and Bounded Plan Existence

Pascal Lauer[1,2], Songtuan Lin[1,2], Pascal Bercher[1]

[1]Australian National University, Australia
[2]Saarland Informatics Campus, Germany

Where this started...

Songtuan Lin, Conny Olz , Malte Helmert , Pascal Bercher:
On the Computational Complexity of Plan Verification, (Bounded) Plan-Optimality Verification, and Bounded Plan Existence (AAAI'24)

		Plan Verification	k-length Plan Existence		Plan Opt. Verification	Bounded Plan Opt. Verification	
			k in binary	k in unary		plan given	plan length given
Classical	ground	In **P**	**PSPACE**	**NP**	**coNP** Prop. 3	**coNP** Prop. 4	**PSPACE** Prop. 5
	lifted	In **P**	**NEXPTIME**	**NP** Thm. 4	**coNP** Prop. 3	**coNP** Prop. 4	**coNEXPTIME** Prop. 5
Hierarchical	ground	**NP** Prop. 2	**NEXPTIME** Thm. 3	**NP** Thm. 5	**coNP** Prop. 3	**coNP** Prop. 4	**coNEXPTIME** Prop. 5
	lifted	**PSPACE**-hard Thm. 1 In **NEXPTIME** Thm. 2	**NEXPTIME**-hard Cor. 1 In 2**NEXPTIME** Cor. 1	**PSPACE**-hard Thm. 6 In **NEXPTIME** Thm. 6	**PSPACE**-hard Prop. 3 In **coNEXPTIME** Prop. 3	**PSPACE**-hard Prop. 4 In **coNEXPTIME** Prop. 4	**coNEXPTIME**-hard Prop. 5 In **co2NEXPTIME** Prop. 5

Understanding the Title

[Complexity of]
Lifted HTN Plan Verification and Bounded Plan Existence

[Complexity of]
Lifted HTN Plan Verification and Bounded Plan Existence

- What is HTN?

$$sol(\text{“}classical\ planning\ problem\text{”}) \cap sol(\text{“}hierarchy\text{”})$$

[Complexity of]
Lifted HTN Plan Verification and Bounded Plan Existence

- What is HTN?

$$sol(\text{“}classical\ planning\ problem\text{”}) \cap sol(\text{“}hierarchy\text{”})$$

- What is lifted?

“User Provided Problem Representation”
(= HDDL Input)

What is the hierarchy?

Problem

The EU forces your logistics company to build a tree for every time a truck is driven.

What is the hierarchy?

Problem

The EU forces your logistics company to build a tree for every time a truck is driven.

Actions: $drive$, $plant\_tree$

Compound Tasks: $EU\_CONFORM$

What is the hierarchy?

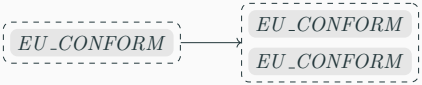
Problem

The EU forces your logistics company to build a tree for every time a truck is driven.

Actions: $drive$, $plant\_tree$

Compound Tasks: $EU\_CONFORM$

Methods:



What is the hierarchy?

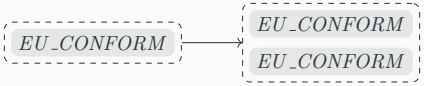
Problem

The EU forces your logistics company to build a tree for every time a truck is driven.

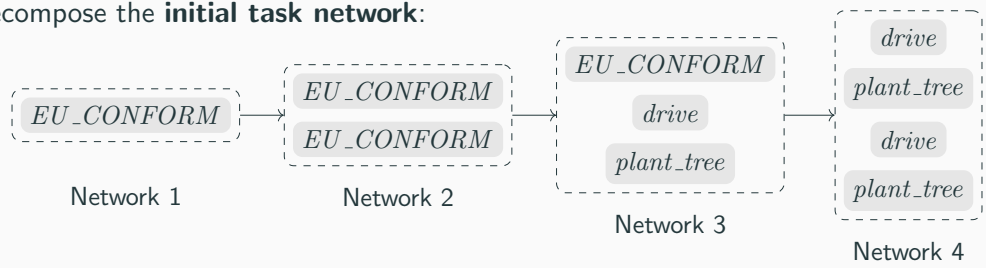
Actions: $drive$, $plant\_tree$

Compound Tasks: $EU\_CONFORM$

Methods:

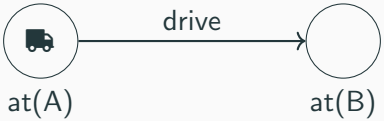


Decompose the **initial task network**:



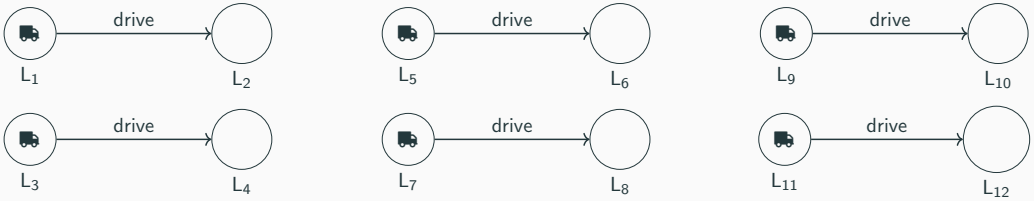
Use any legal ordering, e.g., $plant\_tree, plant\_tree, drive, drive$.

Grounded Actions



$$
\begin{aligned}
&drive(A, B):\\
&\qquad pre : \{at(A)\}\\
&\qquad add : \{at(B)\}\\
&\qquad del : \{at(A)\}
\end{aligned}
$$

Lifted Actions



$$
\begin{aligned}
&drive(?l_1,\ ?l_2):\\
&\qquad pre: \{at(?l_1), adj(?l_1,\ ?l_2)\}\\
&\qquad add: \{at(?l_2)\}\\
&\qquad del: \{at(?l_1)\}
\end{aligned}
$$

$$\mathcal{O} = \{L_1, L_2, ...\}$$

Lifted Methods

$do\_more(?l_1, ?l_2, ?l_3)$:

$head$: $EU\_CONFORM(?l_1, ?l_2)$

$body$: $\{\underbrace{EU\_CONFORM(?l_1, ?l_3)}_{t_1}, \underbrace{EU\_CONFORM(?l_3, ?l_2)}_{t_2}\}$

$order$: $t_1 \prec t_2$

$perform\_task(?l_1, ?l_2, ?t)$:

$head$: $EU\_CONFORM(?l_1, ?l_2)$

$body$: $\{drive(?l_1, ?l_2), plant\_tree(?t)\}$

Decision Problems

Definition

The Plan Verification decision problem is:
Given a task Π, action sequence π decide if π is a plan for Π.

Reason to consider: Do not have to trust the planner. Guess and check procedures.

Definition

The Unary Bounded Plan Existence decision problem is:
Given a task Π, unary bound b decide if there exists a plan for Π of length at most b.

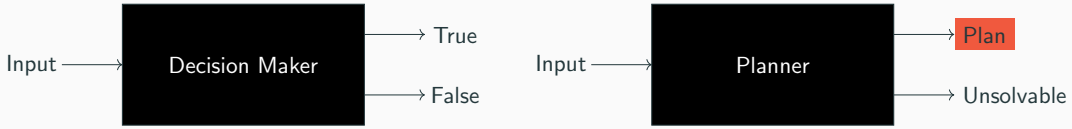
Reason to consider: (Hot take) It's the better plan existence problem.

Definition

The Binary Bounded Plan Existence decision problem is:
Given a task Π, binary bound b decide if there exists a plan for Π of length at most b.

Reason to consider: Weaker version of optimal planning. (& For comparison to unary bound.)

Hot take: Unary Bounded Plan Existence – Tractability in Input and Output



To be tractable: The plan needs to be of polynomial size!

Given a polynomial p:

Has poly-sized plan $\overbrace{\langle \Pi, p \rangle}$ — convert to → Has unary bounded plan $\overbrace{\langle \Pi, 1^{p(|\Pi|)} \rangle}$

Known Results

- Erol, Nau and Subrahmanian (1991) analyzed lifted binary bounded plan existence in classical planning
- Bylander (1994) analyzed grounded binary bounded plan existence in classical planning
- Bäckström and Jonsson (2011) analyzed grounded unary bounded plan existence in classical planning
- Behnke, Holler, and Biundo (2015) analyzed grounded HTN plan verification
- Lin, Olz, Helmert and Bercher (2024) filled the remaining gaps up to lifted HTN planning

$\rightarrow$ We fill the gap for lifted HTN planning.

Filling The Gap

Problem	Classical Planning	HTN Planning
Plan Verification	P	EXPTIME
Unary Bounded Plan Existence	NP	EXPTIME
Binary Bounded Plan Existence	NEXPTIME	NEXPTIME

Table 1: **New results** in HTN planning compared to classical planning. (Assumes lifted formulation.)

Filling The Gap

Problem	Classical Planning	HTN Planning
Plan Verification	P	EXPTIME
Unary Bounded Plan Existence	NP	EXPTIME
Binary Bounded Plan Existence	NEXPTIME	NEXPTIME

Table 1: **New results** in HTN planning compared to classical planning. (Assumes lifted formulation.)

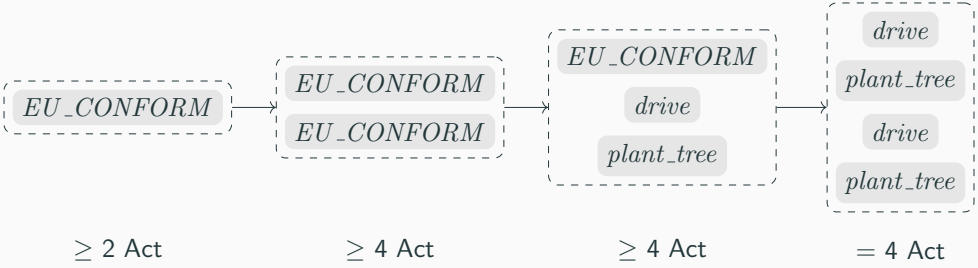
What makes this so complicated?

Why Not Bound The Hierarchy?

- **In general:** Hierarchy is not required as output. (Only the plan.)
- The planner could simplify a complicated hierarchy without performing all decomposition steps:

$$\#plant\_tree = \#drive$$

Relation Plan Length and Decomposition Depth



$\Rightarrow$ There is some relation between number of tasks we decompose to and plan length.

Analysis Extended to Method Size

Definition

The size of a method is the number of elements in it's body.

Problem	Method Size	Complexity
Plan Verification	≥ 0	EXPTIME
	≥ 1	PSPACE
	$= 1$	PSPACE
Unary Bounded Plan Existence	≥ 0	EXPTIME
	≥ 1	PSPACE
	$= 1$	PSPACE
Binary Bounded Plan Existence	≥ 0	NEXPTIME
	≥ 1	NEXPTIME
	$= 1$	PSPACE

Table 2: **Complexity Drop** when restricting method size.

Analysis Extended to Method Size

Definition

The size of a method is the number of elements in it's body.

Problem	Method Size	Complexity
Plan Verification	≥ 0	EXPTIME
	≥ 1	PSPACE
	$= 1$	PSPACE
Unary Bounded Plan Existence	≥ 0	EXPTIME
	≥ 1	PSPACE
	$= 1$	PSPACE
Binary Bounded Plan Existence	≥ 0	NEXPTIME
	≥ 1	NEXPTIME
	$= 1$	PSPACE

Table 2: **Complexity Drop** when restricting method size.

First Things Learned:

- We should ask ourselves whether deleting compound tasks really makes sense.
- There must be a factor that makes repeatedly decomposing even a simple sequence of one task into another very difficult.

Task Insertion Hierarchical Task Network Planning (TIHTN)

- What is HTN?

$$sol(\text{“}classical\ planning\ problem\text{”}) \cap sol(\text{“}hierarchy\text{”})$$

- What is TIHTN?

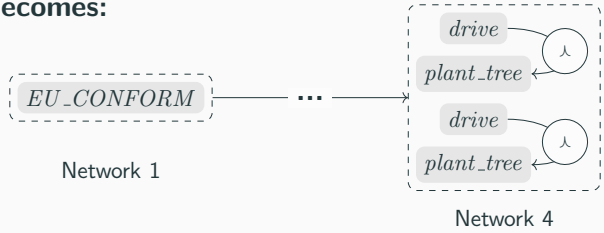
$sol(\text{“}classical\ planning\ problem\text{”}) \cap \{\pi \mid \pi' \in sol(\text{“}hierarchy\text{”})$ is subsequence of $\pi\}$

Makes (unbounded) plan existence decidable. (Geier, and Bercher, 2011)

Task Orderings

Adjust method:

Decomposition becomes:



Use any legal ordering, e.g., ~~*plant_tree*, *plant_tree*, *drive*, *drive*~~.
drive, *drive*, *plant_tree*, *plant_tree*

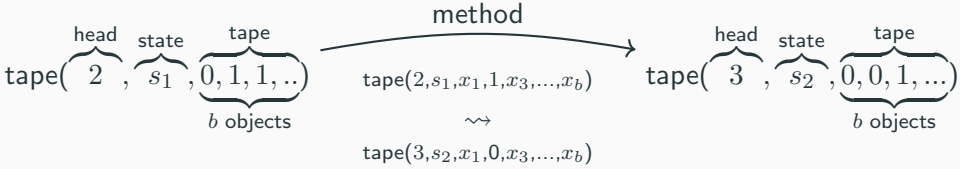
Grounded+Unbounded plan existence drops from undecidable to:

- Total-Order: EXPTime (Erol, Hendler, and Nau, 1996)
- Unordered: PSPACE (Lauer, Zhang, Haslum, and Bercher, 2025)

All Results Summarized

Problem	Method Size	Complexity	Extends to TO/UO	Extends to TIHTN
Plan Verification	≥ 0	EXPTIME	✓	✓
	≥ 1	PSPACE		
	$= 1$	PSPACE		
Unary Bounded Plan Existence	≥ 0	EXPTIME	✓	✓
	≥ 1	PSPACE		
	$= 1$	PSPACE		
Binary Bounded Plan Existence	≥ 0	EXPTIME	✓	✓
	≥ 1	EXPTIME		
	$= 1$	PSPACE		

Simulating (Space Bounded) Turing Machines



Summary

- Identified complexity for Lifted HTN Plan Verification and Bounded Plan Existence (under ordering and method size constraints).
- Found fragments with reduced complexity. (Though still harder than classical planning.)
- Results motivate further study of:
  - How method structure relates to plan length, e.g., only methods that add at least one primitive task.
  - Bounding the number of parameters in compound tasks.

Thank you :)

Focus for remainder of the talk

Theorem

Plan Verification and Unary Bounded Plan Existence for lifted HTN planning tasks is EXPTIME-complete.

Theorem

Plan Verification and Unary Bounded Plan Existence for lifted HTN planning tasks is PSPACE-complete, when method size is restricted to ≥ 1.

We prove:

$$\overbrace{\text{Plan Verification}}^{\text{Hardness}} \leq_{\log} \overbrace{\text{Unary Bounded Plan Existence}}^{\text{Membership}}$$

Plan Verification $\leq_{\log}$ Unary Bounded Plan Existence

Given: Planning task Π, Plan to verify $a_1(\vec{o}_1), ..., a_n(\vec{o}_n)$

Modify actions a in Π:

$$a = (pre(a), add(a), del(a))$$

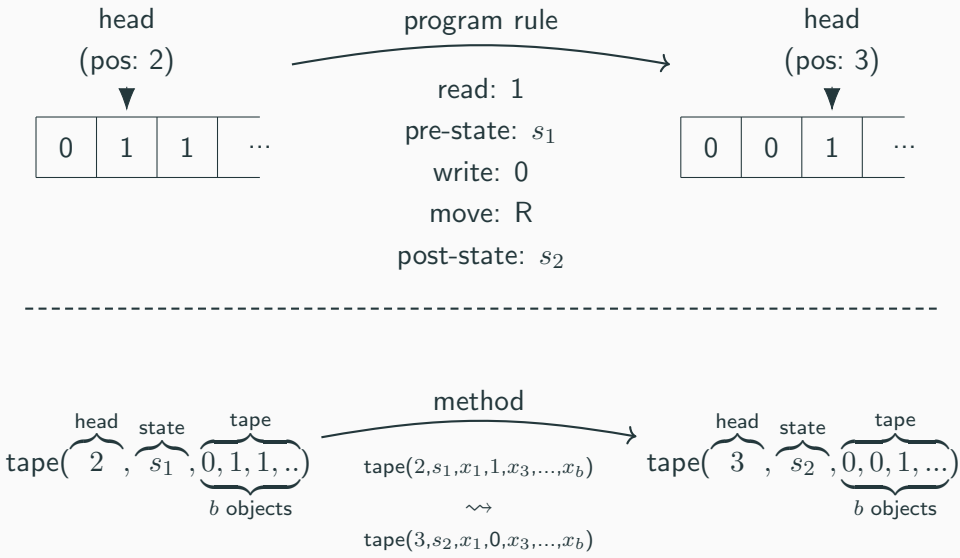
$$\downarrow$$

$$a' := \begin{array}{l} (pre(a) \cup \{count(x_{step}), next(x_{step}, x_{step'})\}, \\ add(a) \cup \{used_a(x_{step}, \vec{x}), count(x_{step'})\}, \\ del(a) \cup \{count(x_{step})\}) \end{array}$$

Modfy goal $\mathcal{G}$ of Π:

$$\mathcal{G}' := \mathcal{G} \cup \{used_{a_1}(1, \vec{o}_1), ..., used_{a_n}(n, \vec{o}_n)\}$$

PSPACE hardness (I): Reduction from space bounded Turing Machines



PSPACE hardness (II): Turing Encoding

Tape construction:

$$t_{w_1,w_2,w_3,i,q} = \begin{cases} tape(1, q, w_2, w_3, x_3, ..., x_b) & \text{, if } i = 1 \\ tape(b, q, x_1, ..., x_{b-2}, w_1, w_2) & \text{, if } i = b \\ tape(i, q, x_{x_1}, ..., x_{i-2}, w_1, w_2, w_3, x_{i+2}, ..., x_b) & \text{, o/w} \end{cases}$$

Transition construction: For

- each transition rule $r = \{(q, w_2) \mapsto \{(q', w_2', lr)\}\} \in \triangleright$
- each tape position $i \in \{1, ..., b\}$ (except for $(lr, i) \in \{(L, 1), (R, b)\}$)

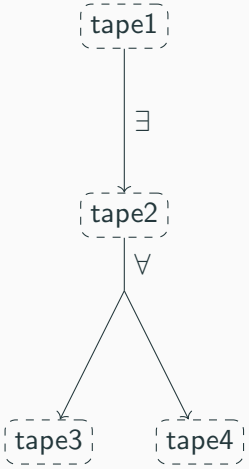
We encode a method that decompose $t_{x_{i-1},w_2,x_{i+1},i,q_1}$ into:

$$t = \begin{cases} a & \text{, if } \mathfrak{l}(q') = \text{accept} \\ t_{w_2,x_{i+1},x_{i+2},i+1,q_2} & \text{, if } lr = R \\ t_{x_{i-2},x_{i-1},w_2,i-1,q_2} & \text{, if } lr = L \end{cases}$$

Verify that plan is a. – Holds for method size ≥ 1.

EXPTime hardness (I): Space Bounded Alternating Turing Machines

Turing Transitions:



Decomposition:

EXPTime hardness (II): Plans to verify

Adjust reduction to delete task on acceptance:

$$t = \begin{cases} \varepsilon & \text{, if } \mathfrak{l}(q') = \text{accept} \\ t_{w_2, x_{i+1}, x_{i+2}, i+1, q_2} & \text{, if } lr = R \\ t_{x_{i-2}, x_{i-1}, w_2, i-1, q_2} & \text{, if } lr = L \end{cases}$$

Verify the empty plan is correct.

Now method size is ≥ 0.

PSPACE completeness for method size ≥ 1

Input: Unary bound b, an HTN problem $\Pi = (\mathcal{P}, C, X, \mathcal{O}, \mathcal{A}, \mathcal{M}, s_{\mathcal{I}}, tn_{\mathcal{I}}, \mathcal{G})$ with method size ≥ 1

$r_b \leftarrow |C^{\mathcal{O}}| + 1, \quad r_c \leftarrow 0$
$tn \leftarrow tn_{\mathcal{I}}$

while tn is not primitive **do**
  guess compound task t from tn, $m \in \mathcal{M}^{\mathcal{O}}$
  if m can not decompose t **then**
    reject
  end if
  $r_c \leftarrow r_c + 1$ if method size of m is 1 else 0
  if $r_c > r_b$ **then**
    reject
  end if
  $tn \leftarrow decompose(tn, t, m)$
  if tn has more than b tasks **then**
    reject
  end if
end while
guess linearization $a_1, \ldots, a_n$ of tn
accept if $a_1, \ldots, a_n$ is a plan, otherwise reject

EXPTime Completeness

- A task can be reduced to an EXPTIME problem, allowing us to verify whether it can be decomposed into the empty task in ground polynomial time (Behnke, Höller, Biundo, 2015).
- The previous algorithm is extended to also include task guessing.