Finding Solution Preserving Linearizations For Partially Ordered Hierarchical Planning Problems

Ying Xian Wu[1], Songtuan Lin[1], Gregor Behnke[2], Pascal Bercher[1]

School of Computing, The Australian National University, Canberra, Australia[1]
ILLC, University of Amsterdam, Amsterdam, The Netherlands[2]

September 19, 2022

Motivation

Why turn PO to TO?

- POHTN planning is semi-decidable
- TOHTN planning is decidable. Specifically 2-EXPTIME-complete with variables (EXPTIME-complete without)
- Converting a POHTN problem to a TOHTN problem allows us to exploit specialised algorithms and heuristics

Introduction to HTN Planning

Problem Definition

A problem $\mathbf{P} = (D, S_I, T_I)$

- has an initial state $S_I \in 2^F$
- has a initial compound task T_I
- is defined over some domain $D = (F, T_P, T_C, \delta, M)$
  - F is the finite set of state variables,
  - T_P is the finite set of all possible primitive task names
  - δ is a mapping from primitive task name to preconditions and effects.
  - T_C is the finite set of all possible compound task names
  - M is the finite set of methods. Each one maps a compound task name to a task network.

Problem Definition (continued)

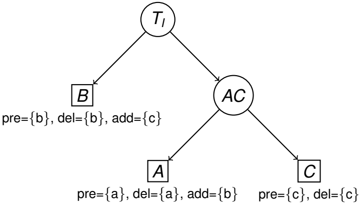
A task network **tn** = $(T, \prec, \alpha)$ consists of

- T, which is a finite set of task identifiers (ids)
- $\prec$, which is a partial order over T;
- α which maps task ids $\in$ T to task names in T_C and T_P.

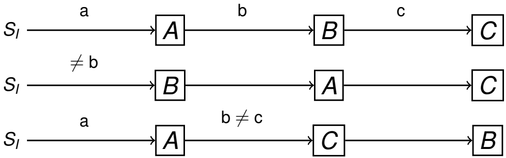
TOHTN problems require $\prec$ to be a total order.

A **solution** to a HTN problem is a task network $tn = (T, \prec, \alpha)$ created via decomposing tn_I. All tasks are primitive, and the sequence must be executable.

$$
\begin{aligned}
F &= \{a, b, c\} \\
S_I &= \{a\} \\
N_p &= \{A, B, C\} \\
N_c &= \{AC, T_I\} \\
M &= \{(T_I, \{AC, B\}), \\
&\quad (AC, \{A, C\})\}
\end{aligned}
$$



(a) The only possible solution A, B, C, requires interleaving



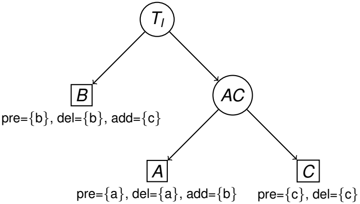
Approach

Linearization Intuition: Linearization Intuition

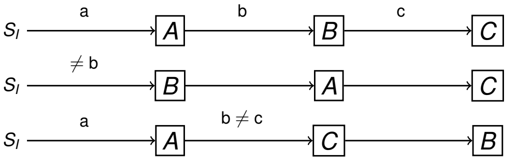
- Transform the problem by linearizing methods
- We want a linearization that will preserve at least one solution.
- A task can't be executed if its preconditions can't be met.
- Therefore:
  - want tasks that add the precondition state variable to execute before-hand
  - don't want tasks that delete its preconditions to directly precede it

Linearization Intuition: Linearization might remove solutions

$F = \{a, b, c\}$
$S_I = \{a\}$
$N_p = \{A, B, C\}$
$N_c = \{AC, T_I\}$
$M = \{(T_I, \{AC, B\}),$
$(AC, \{A, C\})\}$



(a) The only solution *A*, *B*, *C*, requires interleaving. Ordering B before AC, or AC before B, cannot lead to a solution.



Algorithm Example: Infer Preconditions and Effects

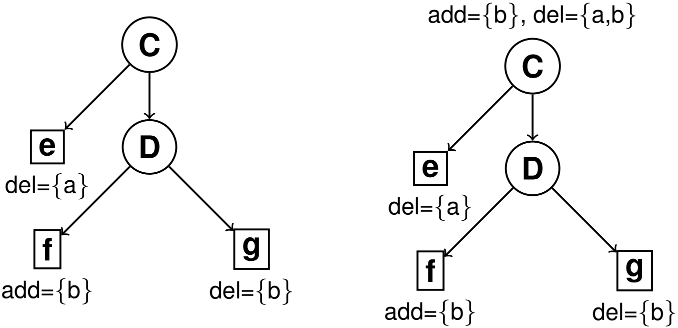
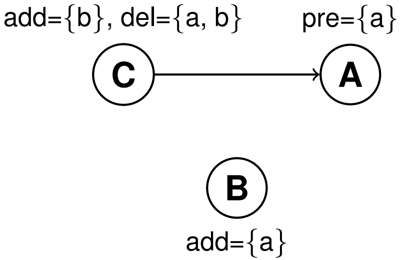


Figure: Inferring preconditions and effects for compound tasks

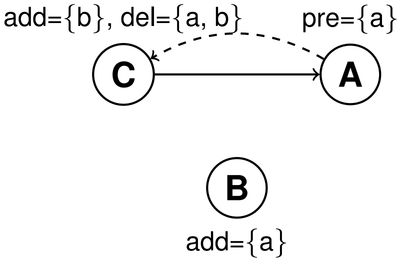
Algorithm Example: Add Orderings



(a) Method with sub-tasks A,B,C, where C is ordered before A

Figure: Adding possible orderings to methods

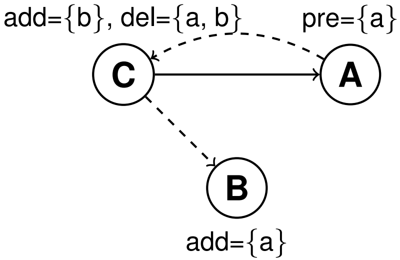
Algorithm Example



(a) C deletes variable *a*, that is in preconditions for A - so A is ordered before C, to prevent making A un-executable

Figure: Adding possible orderings to methods

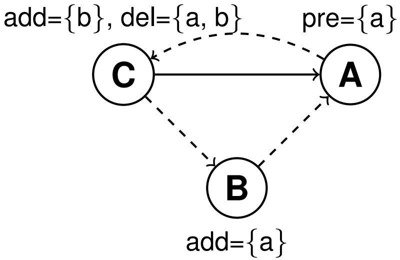
Algorithm Example: Add Orderings (continued)



(a) B adds a variable *a* that C deletes - so C is ordered before B, to preserve *a*

Figure: Adding possible orderings to methods

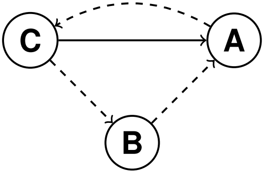
Algorithm Example: Add Orderings (continued)



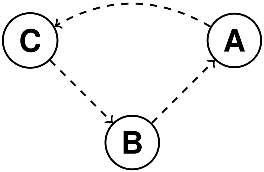
(a) B adds a variable *a* that is in preconditions for A - so B is ordered before A, to help make A executable

Figure: Adding possible orderings to methods

Algorithm Example



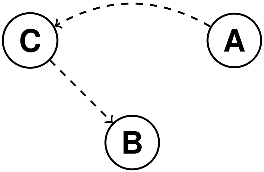
(a) Perform depth-first search on the modified method



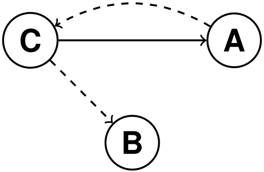
(b) Identify cycle (path along which a node is reachable from one of their ancestors)

Figure: Cycle-breaking (cycle 1)

Algorithm Example: Linearization of orderings



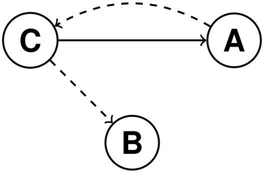
(a) Pick an edge not originally in the method (i.e. a dashed line edge) and delete it.



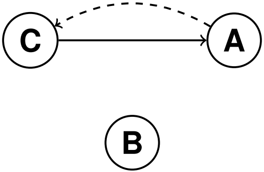
(b) Repeat as necessary until there is no path back to a previously visited node

Figure: Cycle-breaking (cycle 1)

Algorithm Example: Linearization of orderings



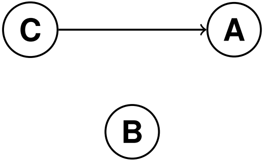
(a) Perform depth-first search on the modified method (again)



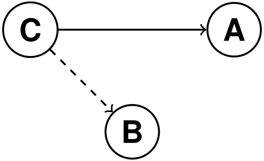
(b) Identify cycle (path along which a node is reachable from one of their ancestors (again))

Figure: Cycle-breaking (cycle 2)

Algorithm Example: Linearization of orderings



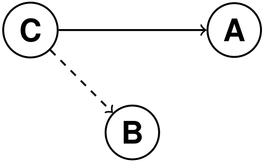
(a) Pick an edge not originally in the method (i.e. a dashed line edge) and delete it (again).



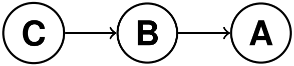
(b) No more cycles, so perform a topological sort to produce a total ordering

Figure: Cycle-breaking (cycle 2)

Algorithm Example



(a) Perform a topological sort on this



(b) Resulting Linearization

Contributions

New class of decideable problems

Theorem

You can preserve at least one solution if you linearize all methods without having to cycle-break.

Proof outline

- Suppose we want to execute task t, with precondition f.
- Then f is in the initial state, or there's a task that adds f.
- Tasks that delete f are ordered after t, by algorithm definition.
- Tasks that add f are ordered before t, by algorithm definition.
- So f is present before t executes, and not deleted until t has executed.

New class of decideable problems

When certain criteria are met, it guarantees that at least one solution will be preserved. This means we obtain a new class of decidable problems, namely those that satisfy the above mentioned criteria.

Empirical Evaluation

- 7.3 percent of problems were unsolvable after linearization.
- 11 percent increase in number of solvable problems
- 20 percent increase in number of solvable problems if using re-run policy

Empirical Evaluation

Table: IPC score, with and without pre-processing, for all planners. If any problems in that domain were proven unsolvable by TO, a number in brackets beside domain name shows how many.

		RC^{add}		RC^{Filter}		RC^{FF}		$RC^{LM\text{-}Cut}$		Lilotane
	max	PO	TO	PO	TO	PO	TO	PO	TO	
Barman-BDI	1	0.08	**0.4**	0.07	**0.34**	0.07	**0.36**	0.05	**0.22**	**0.66**
Monroe Fully Observ. (2)	1	**0.56**	0.45	**0.31**	0.3	**0.46**	0.41	**0.22**	0.18	0.07
Monroe Part. Observ. (2)	1	**0.31**	0.25	**0.13**	0.11	**0.31**	0.26	**0.17**	0.14	0.0
PCP (17)	1	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**	0.0
Rover	1	0.29	**0.95**	0.14	**0.52**	0.2	**0.78**	0.16	**0.48**	**0.98**
Satellite	1	0.91	**1.0**	0.76	**1.0**	0.99	**1.0**	0.89	**0.99**	**1.0**
SmartPhone (1)	1	**0.71**	**0.71**	0.69	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**
Transport	1	0.24	**0.61**	0.04	**0.05**	0.27	**0.32**	0.12	**0.2**	**0.71**
UM-Translog (1)	1	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.95
Woodworking (2)	1	0.38	**0.58**	0.2	**0.41**	0.36	**0.57**	0.27	**0.39**	0.47
Monroe	1	**0.77**	0.69	**0.5**	0.47	**0.75**	0.71	**0.53**	**0.53**	0.46
SmartPhone (1)	1	**0.71**	**0.71**	0.69	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**
Zenotravel	1	**1.0**	**1.0**	0.63	**1.0**	**1.0**	**1.0**	0.83	**1.0**	**1.0**
Total IPC score	13	7.8	**9.2**	6.0	**7.5**	7.7	**8.7**	6.5	**7.4**	7.7

Empirical Evaluation

Table: Coverage, with and without pre-processing, for all planners. If any problems in that domain were proven unsolvable by TO, a number in brackets beside domain name shows how many.

		RC^{add}		RC^{Filter}		RC^{FF}		$RC^{LM\text{-}Cut}$		Lilotane
	max	PO	TO	PO	TO	PO	TO	PO	TO	
Barman-BDI	20	3	**10**	3	**10**	3	**10**	2	**9**	**16**
Monroe Fully Observ. (2)	25	**25**	**25**	18	**25**	22	**25**	15	**16**	6
Monroe Part. Observ. (2)	24	**14**	**14**	**7**	**7**	14	**15**	**10**	**10**	0
PCP (17)	17	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	0
Rover	20	6	**20**	4	**14**	4	**19**	4	**14**	**20**
Satellite	25	24	**25**	22	**25**	**25**	**25**	24	**25**	**25**
SmartPhone (1)	7	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
Transport	40	12	**28**	**2**	**2**	13	**14**	7	**12**	**31**
UM-Translog (1)	22	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	21
Woodworking (2)	30	13	**19**	7	**15**	12	**20**	9	**15**	15
Monroe	100	96	**100**	79	**88**	92	**100**	81	**90**	83
SmartPhone (1)	7	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
Zenotravel	5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
Coverage	342	244	**292**	193	**237**	236	**279**	203	**242**	232
Norm. coverage	13	8.94	**10.67**	7.57	**9.17**	8.71	**10.34**	7.81	**9.16**	8.53

Summary

Summary

1. Almost all problems retain solutions after linearization
2. Problems are generally solved more quickly when using linearization algorithm, for a variety of planners/heuristics.
3. Critera for new class of decidable problems.