Grounded (Lifted) Linearizer at the IPC 2023: Solving Partial Order HTN Problems by Linearizing Them

Ying Xian Wu[1], **Conny Olz**[2], **Songtuan Lin**[1], **Pascal Bercher**[1]

[1]School of Computing, The Australian National University
[2]Institute of Artificial Intelligence, Ulm University
{yingxian.wu, songtuan.lin, pascal.bercher}@anu.edu.au, conny.olz@uni-ulm.de

Abstract

In this paper, we would like to present Grounded (Lifted) Linearizer, a hierarchical task network (HTN) planning system which won the Partial Order (PO) Agile and Satisficing tracks of the International Planning Competition 2023 on Hierarchical Task Network (HTN) Planning. This system consists of two parts. The first part is a *preprocessor* developed in house which transforms a POHTN problem into a *total order* (TO) one and which is the main contribution of this paper. The second part is an *existing* HTN planner. The outstanding performance of our assembled planning system thus serves as an evidence for how our preprocessor can enhance the efficiency of other existing planners.

Introduction

In this paper, we present Grounded and Lifted Linearizer, the systems solving hierarchical task network (HTN) planning problems (Bercher, Alford, and Höller 2019) which participated in the International Planning Competition (IPC) 2023 on HTN Planning and won the Partial Order (PO) Agile and Satisficing tracks. Generally speaking, the systems work as follows: they first transform an input POHTN planning problem into a *total order* (TO) one on the grounded (*resp*. lifted) level while ensuring that a solution to the transformed problem is also a solution to the original one. After that, a third-party HTN planner, which we call the *inner* planner, is invoked to solve the obtained TO problem. If this second step fails, i.e., the inner planner reports that the TO problem is unsolvable, a third-party HTN planner, called the *outer* planner, which might or might not be the same one used as the inner planner, is called to solve the original PO problem. Notably, the novelty of the systems is the linearization technique, and the actual process of solving an HTN problem is still done by other existing HTN planners.

In the remaining section, we will give more details about the process of linearizing and the configurations with respect to the agile track and the satisficing track.

Linearizing PO Problems

There are two variants of our linearizing technique. The first one linearizes *grounded* PO problems, and the other is targeted at *lifted* problems. Both variants share the same idea. That is, for each method (either grounded or lifted), we first infer the precondition and effects of each *compound task* in it, and then, the method is linearized according to those inferred preconditions and effects.

Linearizing Grounded Problems

For the variant which linearizes grounded problems, we employed *two* approaches for inferring compound tasks' preconditions and effects. The first approach is easy-to-compute but less informed while the other (Olz, Biundo, and Bercher 2021; Olz and Bercher 2022) has higher complexity but can compute more precise preconditions and effects. For convenience, we call the former one the *simple* inference approach and the latter *complex* inference approach.

More specifically, for each compound task, the simple inference approach regards the precondition and effects of every action which can be obtained from its decompositions as its own precondition and effects, i.e., it is a simple collection of all actions that can be reached. For more details about this simple inference approach, we refer to our previous work (Wu et al. 2022). Contrastively, the complex inference approach is the one developed by Olz and Bercher (2022) for PO problems which is based on the previous work by Olz, Biundo, and Bercher (2021) for TO problems and which further rule out some impossible propositions in a compound task's inferred precondition and effects, on top of those computed by the simple inference approach.

Having obtained preconditions and effects of compound tasks, we can linearize each method in a PO problem by exploiting them. Note that the linearizing approach only leverages the inferred preconditions and effects and is independent of how they are obtained (i.e., by the simple inference approach or the complex one). Informally, the linearizing approach can be summarized by the following two rules:

1) For any two tasks t_1 and t_2 where t_1 adds a proposition that is required by t_2, we want to place t_1 in front of t_2 so that t_2 will more likely be applicable.
2) If t_1 deletes a proposition that is required by t_2, we would like to place t_2 before t_1 so that t_1 will have less chance to be unapplicable.

For detailed implementation of the approach, we again refer to our previous work (Wu et al. 2022).

For every method in a PO problem, we only generate *one* linearization. In other words, the number of methods in the linearized problem is identical to that in the original PO problem. Furthermore, a linearized problem is guaranteed to

preserve at least one solution if for *any* two tasks t_1 and t_2 in *any* method in the original PO problem, the order on these two tasks induced by the above two rules is consistent, i.e., t_1 will *not* be put both before and after t_2 (Wu et al. 2022, Thm. 4).

Linearizing Lifted Problems

When linearizing lifted problems, for each lifted method, we instantiate each variable with a hypothetical instance of the appropriate type. For variables of the same type, all are assumed to be different instances from every other. Note that the variables we instantiate are the parameters of the respective method which, as we will show later on, will be inherited by the subtasks of this method (because the parameters of every subtask are also the parameters of the method). This thus ensures that we can reason on the inferred preconditions and effects of the subtasks when linearizing the method.

A method $\mathbf{m}$, whose instantiated counterpart is referred to as $\mathtt{m}$, has some subtasks. For each (abstract or primitive) subtask, we apply the appropriate instantiated variable from those applied to their parent, to produce an instantiated counterpart for each lifted subtask. If the child refers to a variable the parent does not refer to, a unique hypothetical instance is used as instantiation. We perform the following two operations depending on whether a subtask is primitive or not: 1) If the subtask is primitive, we instantiate its precondition and effects with the appropriate hypothetical instances. For variables that do not appear in the primitive task, a unique hypothetical instance is used as instantiation. 2) If the subtask is compound, then for each method $\mathbf{m}'$ that could be applied to it, the method is also instantiated as $\mathtt{m}'$ with the appropriate hypothetical instances in accordance with the instantiation of the task. We repeat this instantiation for methods and compound tasks that the original method could decompose into, until we obtain a collection of instantiated actions that this instantiated method $\mathtt{m}$ could have decomposed to.

For all instantiated actions an instantiated task could eventually decompose into, we consider all instantiated effects and preconditions as the effects and preconditions of the task. For each subtask, we regard instantiated preconditions and effects of every instantiated action which can be obtained from its decompositions as its own precondition and effects (i.e. using the simple inference approach). Having obtained inferred preconditions and effects, we again use the two-step process for linearizing grounded problems. The lifted subtasks are ordered as their instantiated counterparts would be. Anything instantiated is now discarded.

Configurations

Based on the linearizing approaches described above, we developed three planning systems to participate in the IPC 2023 on HTN Planning:

1) Grounded-Simple-Linearizer,
2) Grounded-Complex-Linearizer, and
3) Lifted-Linearizer.

More specifically, all these systems participated the PO agile track and the PO satisficing track. We did *not* participate in the optimal track because a linearized problem might *not* preserve any optimal solution to the original problem.

All systems consist of the following three components:

1) a linearizer that linearizes a PO problem,
2) an inner planner that solves the linearized problem, and
3) an outer planner that solves original PO problems if the linearized problem has no solutions.

Grounded-Simple-Linearizer

This system participated in the (partial order) agile and the satisficing track. For each track, we had three configurations (i.e., 6 configurations in total). All 6 configurations used the *same* linearizer and the same configuration of the *outer* planner, that is, only the settings of the inner planner were different. More concretely, the linearizer of this system used our linearizing technique for *grounded* problems with the *simple* inference approach. Both the inner and outer planners were PANDA$_\pi$ with the progression-based solver. The outer planner used the relaxed composition (RC) heuristic (Höller et al. 2018, 2020) with *Fast Forward* (FF) (Hoffmann and Nebel 2001) as the inner heuristic, written RC(FF), in conjunction with a weighted A* (WA*) search with the weight being two. The heuristic estimated the number of actions and methods needed to reach a solution (i.e., the distance). The g-value for the search is the mixture of action costs and decomposition costs. The different settings of the inner planner for each configuration are as follows.

Agile track In the first configuration, we adapted the RC heuristic with *Add* (Bonet and Geffner 2001) as the inner classical heuristic, written RC(Add), together with a greedy best first search (GBFS) where a search node with the best heuristic value is expanded. In the second configuration, we used the RC(FF) heuristic with a GBFS. For the last one, we used the heuristic RC(Add) together with a WA* search with the weight being 2 where the g-value is again the mixture of action costs and decomposition costs. The heuristics used in all configurations estimated the distance to a solution.

Satisficing track The settings of the inner planner for the satisficing track are more complicated because the goal of this track is to find a solution whose length is as closed to that of an optimal solution as possible. To this end, we design a three-round search where each round aims at finding a better solution than the previous one. Notably, the g-value used in the search in *all* configurations for the satisficing track is action costs.

Concretely, in the first configuration, we design a three-round search. In the first round, we use the RC(Add) heuristic, estimating the distance to a solution, with a greedy best first search. If a solution is found in the first round, then the inner planner will start the second round of search where we use the RC(FF) heuristic (which again estimated the distance to a solution) in conjunction with a weighted A* search with the weight being two. In particular, in the second round of search, we eliminate search nodes whose f-value is greater than the cost of the solution found in the first round. This ensures that if a solution is found in the second round of search, then its cost is guaranteed to be smaller than that of the solution found in the first round. Similarly, in the third

round of search, we again use the RC(FF) heuristic with a weighted A* search except that the weight is 1.5 this time.

For the second configuration, we used the RC(FF) heuristic (estimating the distance) with a weighted A* search with the weight being 2 in the first round of search. In the second round, we adapted the same setting except for reducing the weight to 1.5. Lastly, in the third round, we used the RC heuristic with *Landmark Cut* (Helmert and Domshlak 2009) as the inner heuristic, written RC(LMC), which now estimated the *cost* of action required to reach a solution and is an *admissible* heuristic, with an A* search.

In the last configuration, we only do a single-round search where we use the RC(LMC) heuristic with an A* search.

Grounded-Complex-Linearizer

This system participated in the agile and satisficing tracks as well. It again has three configurations per track. All of them share the same linearizer, which uses the complex inference approach for grounded problems. The settings of the outer and inner planners for each configuration are identical to the respective one for the agile track except that the inner planner used is PANDADEALER (Olz, Höller, and Bercher 2023), an advanced version of PANDA$_\pi$ equipped with the *dead-end look-ahead* technique (Olz and Bercher 2023) and the landmark technique (Höller and Bercher 2021; Höller 2023a) which won all three total order tracks of the IPC 2023. It uses the same complex inference approach (which thus allows us to reuse the inferred preconditions and effects) and is customized to solve TO problems more efficiently.

Lifted-Linearizer

The last system only participated in the agile track. However, we intended to submit it to both tracks, but we forgot to register it for the satisficing track. This system had three configurations. All of these three configurations use the same linearizer that works on *lifted* problems and the same outer planner that is identical to the previous two. The setting of the inner planner per configuration is as follows. In the first one, we use Lilotane (Schreiber 2021) with Glucose 4 as the SAT solver. In the second configuration, PANDA$_\pi$ with the SAT-based solver (Behnke, Höller, and Biundo 2018, 2019a) is used. For the last one, we also use PANDA$_\pi$ with the SAT-based solver except that it is now configured for finding *optimal* solutions (Behnke, Höller, and Biundo 2019b). Note that all inner planners are based on SAT, and Lilotane works on lifted problems.

Summary

Lastly, we provide a brief summary for our systems.

Agile track

- **Grounded-Simple-Linearizer**
  + Outer planner: PANDA$_\pi$
    * Heuristic: RC(FF) (estimation: distance)
    * Search: WA* ($w = 2$, g-value is the mixture of action costs and decomposition costs)
  + Inner planner: PANDA$_\pi$
    * Configuration 1
      - Heuristic: RC(Add) (estimation: distance)
      - Search: GBFS
    * Configuration 2
      - Heuristic: RC(FF) (estimation: distance)
      - Search: GBFS
    * Configuration 3
      - Heuristic: RC(Add) (estimation: distance)
      - Search: WA* ($w = 2$, g-value is the mixture of action costs and decomposition costs)
- **Grounded-Complex-Linearizer**
  + Outer planner: same as the simple version
  + Inner planner: PANDADEALER (Olz, Höller, and Bercher 2023)
    * All configurations are identical to the simple version
- **Lifted-Linearizer**
  + Outer planner: same as above
  + Configuration 1
    * Inner planner: Lilotane
  + Configuration 2
    * Inner planner: PANDA$_\pi$ with SAT
  + Configuration 3
    * Inner planner: PANDA$_\pi$ with SAT (optimal version)

Satisficing track

- **Grounded-Simple-Linearizer**
  + Outer planner: same as that for the agile track
  + Inner planner: PANDA$_\pi$
    * Configuration 1
      - Round 1
        · Heuristic: RC(Add) (estimation: distance)
        · Search: GBFS
      - Round 2
        · Heuristic: RC(FF) (estimation: distance)
        · Search: WA* ($w = 2$, g-value is action costs)
      - Round 3
        · Heuristic: RC(FF) (estimation: distance)
        · Search: WA* ($w = 1.5$, g-value is action costs)
    * Configuration 2
      - Round 1
        · Heuristic: RC(FF) (estimation: distance)
        · Search: WA* ($w = 2$, g-value is action costs)
      - Round 2
        · Heuristic: RC(FF) (estimation: distance)
        · Search: WA* ($w = 1.5$, g-value is action costs)
      - Round 3
        · Heuristic: RC(LMC) (estimation: action costs)
        · Search: A* (g-value is action costs)
    * Configuration 3
      - Heuristic: RC(LMC) (estimation: action costs)
      - Search: A* (g-value is action costs)
- **Grounded-Complex-Linearizer**
  + Outer planner: identical to the simple version
  + Inner planner: PANDADEALER (Olz, Höller, and Bercher 2023)
    * All configurations are identical to the simple version

Planner	Score	Domains									
		Barman	**Monroe (FO)**	**Monroe (PO)**	**PCP**	**Rover**	**Satellite**	**Transport**	**Cockpit**	**Woodworking**	**Colouring**
Linearizer (G, C, Config-1)	**7.60238 (197)**	0.66 (14)	0.94 (24)	0.70 (18)	**0.82 (14)**	**0.86 (19)**	0.98 (25)	0.26 (12)	**1.00 (29)**	**0.70 (21)**	**0.69 (21)**
Linearizer (G, C, Config-2)	7.47407 (189)	**0.73 (15)**	0.94 (24)	0.64 (16)	**0.82 (14)**	0.85 (18)	**1.00 (25)**	0.17 (8)	**1.00 (29)**	0.63 (19)	**0.69 (21)**
Linearizer (G, S, Config-1)	7.47253 (194)	0.60 (13)	**0.96 (24)**	0.66 (17)	**0.82 (14)**	0.81 (19)	**1.00 (25)**	0.26 (12)	**1.00 (29)**	**0.70 (21)**	0.66 (20)
Linearizer (G, S, Config-2)	6.96385 (176)	0.54 (11)	0.84 (21)	0.56 (14)	**0.82 (14)**	0.83 (18)	**1.00 (25)**	0.15 (7)	**1.00 (29)**	0.57 (17)	0.66 (20)
Linearizer (G, C, Config-3)	6.92736 (176)	0.50 (10)	0.90 (23)	0.60 (15)	**0.82 (14)**	0.75 (16)	**1.00 (25)**	0.19 (9)	**1.00 (29)**	0.50 (15)	0.66 (20)
Linearizer (G, S, Config-3)	6.76902 (172)	0.50 (10)	0.84 (21)	0.52 (13)	**0.82 (14)**	0.73 (16)	**1.00 (25)**	0.20 (9)	**1.00 (29)**	0.50 (15)	0.66 (20)
$\text{PANDA}_{\text{pro}}$-$\lambda$ (agl, gas, ao)	6.46739 (171)	0.15 (3)	0.91 (23)	**0.75 (19)**	**0.82 (14)**	0.43 (10)	0.99 (25)	**0.33 (15)**	**1.00 (29)**	0.42 (13)	0.66 (20)
$\text{PANDA}_{\text{pro}}$ (sat, gas, ff)	6.30870 (167)	0.19 (4)	**0.96 (24)**	0.72 (18)	**0.82 (14)**	0.27 (7)	0.97 (25)	0.32 (14)	**1.00 (29)**	0.39 (12)	0.66 (20)
$\text{PANDA}_{\text{pro}}$-$\lambda$ (agl, gas, lmc)	6.28603 (166)	0.15 (3)	0.95 (24)	0.66 (17)	**0.82 (14)**	0.37 (9)	0.99 (25)	0.29 (13)	**1.00 (29)**	0.39 (12)	0.66 (20)
Aries (sat)	4.73026 (118)	0.15 (3)	0.44 (11)	0.52 (13)	0.59 (10)	0.79 (16)	**1.00 (25)**	0.32 (13)	0.66 (19)	0.00 (0)	0.27 (8)
SIADEX	2.17641 (73)	0.62 (20)	0.26 (8)	0.04 (2)	0.00 (0)	0.31 (14)	0.83 (25)	0.03 (1)	0.00 (0)	0.09 (3)	0.00 (0)
Total	10.0000 (261)	1.00 (20)	1.00 (25)	1.00 (25)	1.00 (17)	1.00 (20)	1.00 (25)	1.00 (40)	1.00 (29)	1.00 (30)	1.00 (30)

Table 1: The performance scores of all participating planners in the satisficing track together with the number of problem instances solved by them (written within the parentheses in each cell). The configurations of each planner are written within the parentheses below the name of the respective planner. For our planner (Linearizer), the letter G refers to the version of the linearization technique that works on grounded problems, and the letter S (C) refers to the simple (complex) approach for inferring compound tasks' preconditions and effects. For the PANDA family, "agl" and "sat" refer to the agile and the satisficing track, respectively, "gas" means the greedy A^* search algorithm. "ff" and "lmc" are the heuristics RC(FF) and RC(LMC).

Planner	Score	Domains									
		Barman	**Monroe (FO)**	**Monroe (PO)**	**PCP**	**Rover**	**Satellite**	**Transport**	**Cockpit**	**Woodworking**	**Colouring**
Linearizer (G, S, Config-2)	**7.03533 (218)**	0.81 (18)	**0.57 (24)**	0.44 (21)	**0.82 (14)**	**1.00 (20)**	**1.00 (25)**	0.52 (23)	0.93 (29)	0.63 (24)	0.30 (20)
Linearizer (G, C, Config-2)	7.01996 (222)	0.85 (18)	0.50 (24)	0.43 (21)	**0.82 (14)**	**1.00 (20)**	**1.00 (25)**	0.54 (24)	0.93 (29)	0.64 (26)	**0.31 (21)**
Linearizer (G, C, Config-1)	6.89761 (217)	0.75 (16)	0.49 (24)	0.34 (18)	**0.82 (14)**	**1.00 (20)**	**1.00 (25)**	0.63 (27)	0.93 (29)	0.62 (23)	**0.31 (21)**
Linearizer (G, C, Config-3)	6.89485 (217)	0.75 (16)	0.49 (24)	0.32 (17)	**0.82 (14)**	**1.00 (20)**	**1.00 (25)**	0.66 (28)	0.93 (29)	0.61 (23)	**0.31 (21)**
Linearizer (G, S, Config-1)	6.81972 (210)	0.70 (16)	**0.57 (24)**	0.35 (17)	**0.82 (14)**	0.99 (20)	**1.00 (25)**	0.54 (23)	0.93 (29)	0.61 (22)	0.30 (20)
Linearizer (G, S, Config-3)	6.64191 (207)	0.51 (12)	**0.57 (24)**	0.33 (17)	**0.82 (14)**	**1.00 (20)**	**1.00 (25)**	0.57 (24)	0.93 (29)	0.60 (22)	0.30 (20)
Linearizer (L, Config-1)	6.44606 (193)	0.70 (17)	0.51 (18)	**0.60 (21)**	0.00 (0)	0.97 (20)	**1.00 (25)**	0.78 (34)	0.97 (29)	**0.91 (29)**	0.00 (0)
Linearizer (L, Config-2)	6.24541 (196)	0.75 (18)	0.43 (18)	0.48 (21)	0.00 (0)	0.99 (20)	**1.00 (25)**	**0.88 (39)**	**0.98 (29)**	0.74 (26)	0.00 (0)
Linearizer (L, Config-3)	5.20113 (182)	0.74 (18)	0.38 (18)	0.40 (21)	0.00 (0)	0.58 (19)	**1.00 (25)**	0.64 (31)	0.85 (29)	0.60 (21)	0.00 (0)
$\text{PANDA}_{\text{pro}}$-$\lambda$ (agl, gas, ao)	5.01524 (171)	0.08 (3)	0.50 (23)	0.36 (19)	**0.82 (14)**	0.36 (10)	0.99 (25)	0.34 (15)	0.92 (29)	0.36 (13)	0.29 (20)
$\text{PANDA}_{\text{pro}}$ (agl, gas, ff)	5.00565 (167)	0.10 (4)	0.53 (24)	0.36 (18)	**0.82 (14)**	0.32 (7)	**1.00 (25)**	0.30 (14)	0.91 (29)	0.36 (12)	0.30 (20)
$\text{PANDA}_{\text{pro}}$-$\lambda$ (agl, gas, lmc)	4.95080 (166)	0.08 (3)	0.50 (24)	0.34 (17)	**0.82 (14)**	0.38 (9)	**1.00 (25)**	0.26 (13)	0.90 (29)	0.39 (12)	0.29 (20)
Aries (agile)	3.22657 (119)	0.05 (2)	0.21 (11)	0.26 (13)	0.38 (12)	0.44 (15)	**1.00 (25)**	0.20 (13)	0.58 (19)	0.00 (0)	0.12 (9)
SIADEX	3.03256 (73)	**0.92 (20)**	0.24 (8)	0.05 (2)	0.00 (0)	0.70 (14)	**1.00 (25)**	0.03 (1)	0.00 (0)	0.10 (3)	0.00 (0)
Total	10.0000 (261)	1.00 (20)	1.00 (25)	1.00 (25)	1.00 (17)	1.00 (20)	1.00 (25)	1.00 (40)	1.00 (29)	1.00 (30)	1.00 (30)

Table 2: The performance scores of all participating planners in the agile track. For our planner (Linearizer), the letter "L" within the parentheses refers to the lifted version.

		Simple Inference Approach						Complex Inference Approach					
		Agile			Satisficing			Agile			Satisficing		
	Total	Config-1	Config-2	Config-3	Config-1	Config-2	Config-3	Config-1	Config-2	Config-3	Config-1	Config-2	Config-3
Barman	20	3	1	6	3	5	8	0	0	0	0	0	7
Monroe (FO)	25	8	8	8	8	8	9	6	6	6	6	6	6
Monroe (PO)	25	5	5	5	5	5	12	3	3	3	3	3	5
PCP	17	17	17	17	17	17	17	17	17	17	17	17	17
Rover	20	0	0	0	0	1	4	0	0	0	0	1	4
Transport	40	0	0	0	0	0	20	0	0	0	0	0	18
Cockpit	34	28	28	28	28	28	28	28	28	28	28	28	28
Woodworking	31	7	5	8	8	5	5	1	1	1	1	1	3
Colouring	30	30	30	30	30	30	30	30	30	30	30	30	30

Table 3: Number of linearized problems that are reported to be unsolvable by the inner planner for each configuration. The data for the lifted system is not shown here because it is not recoverable.

IPC Results

Lastly, we present the performance scores of our planners in the IPC together with that of all the remaining participants, namely, $\text{PANDA}_{\text{pro}}$ (Höller 2023b), $\text{PANDA}_{\text{pro}}$-$\lambda$ (Höller 2023a), Aries (Bit-Monnot 2023), and SIADEX by Expósito, Soler-Padial, Fernández-Olivares, and Castillo which is based on the previous work by Fernández-Olivares et al. (2006). The results for the satisficing track are summarized in Tab. 1, whereas Tab. 2 summarizes the results for the agile track. A planner's performance score for solving *one* single instance for the agile track is computed as $1 - (\log T / \log 1800)$ where T is the time needed by the planner to solve the instance. The total score for a planner is the sum of each score for solving each instance. For the satisficing track, the score for solving one instance is computed as C/C^* where C is the cost of the solution found, and C^* is the cost of an *optimal* solution. The total score is again the sum of each score for solving each instance.

Furthermore, our linearization techniques *cannot* guarantee that a linearized problem is solvable. Hence, for the purpose of better characterizing the performance of those linearization techniques, we also summarize in Tab. 3 the number of *unsolvable* linearized problems reported by the inner planner for each configuration and domain. The columns labelled with 'simple' and 'complex' indicate which inference approach is used, and those labelled with 'agile' and 'satisficing' indicate the configurations for the respective track. We did not collect the number of unsolvable linearized instances produced by the lifted linearizer because we wrapped up all intermediate information output by that system, which caused the result that the number on demand was not recoverable.

Conclusion

In this paper, we presented our *linearization techniques*, two for grounded problems and one for lifted ones, and three systems that incorporate our techniques into other *existing* planners. The systems won the PO agile and satisficing tracks of the IPC 2023 on HTN Planning. We did not participate in the optimal track because our linearization technique does not guarantee to maintain optimal solutions. The results indicate that although POHTN problems are in theory much more expressive than TO ones in terms of complexity (Erol, Hendler, and Nau 1996; Geier and Bercher 2011; Alford, Bercher, and Aha 2015; Bercher, Lin, and Alford 2022) and solution space (Höller et al. 2014, 2016), most PO problems *in practice* do not require such additional expressive power and can be solved more efficiently by eliminating partial order and using specialized TOHTN planners.

Acknowledgments

We would like to thank all authors who allowed us to use their planners as our inner (and outer) planners, specifically, Daniel Höller and Gregor Behnke who developed PANDA_π and Dominik Schreiber who developed Lilotane. We want to particularly thank Daniel Höller, who allowed us to use his planner even though the same system also participated in the same tracks as our systems – thus knowing that our planner will most likely outperform his. We would also like to thank Ron Alford, Dominik Schreiber, and Gregor Behnke for organizing the IPC 2023 HTN tracks.

Lastly, we emphasize again that our main contribution is the development of the linearization techniques and that the actual planning processes are still done by the inner and outer planners. Therefore, although our systems won the respective tracks, we still regard the inner and outer planners we used as the actual winners.

References

Alford, R.; Bercher, P.; and Aha, D. W. 2015. Tight Bounds for HTN Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling, ICAPS 2015*, 7–15. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2018. totSAT - Totally-Ordered Hierarchical Planning Through SAT. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 6110–6118. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2019a. Bringing Order to Chaos - A Compact Representation of Partial Order in SAT-Based HTN Planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, 7520–7529. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2019b. Finding Optimal Solutions in HTN Planning - A SAT-based Approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 5500–5508. IJCAI.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 6267–6275. IJCAI.

Bercher, P.; Lin, S.; and Alford, R. 2022. Tight Bounds for Hybrid Planning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022*, 4597–4605. IJCAI.

Bit-Monnot, A. 2023. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. In *Proceedings of the 11th International Planning Competition: Planner Abstracts – Hierarchical Task Network (HTN) Planning Track, IPC 2023*.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1-2): 5–33.

Erol, K.; Hendler, J.; and Nau, D. S. 1996. Complexity Results for HTN Planning. *Annals of Mathematics and Artificial Intelligence*, 18(1): 69–93.

Fernández-Olivares, J.; Castillo, L. A.; García-Pérez, Ó.; and Palao, F. 2006. Bringing Users and Planning Technology Together. Experiences in SIADEX. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling, ICAPS 2006*, 11–20. AAAI.

Geier, T.; and Bercher, P. 2011. On the Decidability of HTN Planning with Task Insertion. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, 1955–1961. AAAI.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009*. AAAI.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Höller, D. 2023a. The PANDA λ System for HTN Planning in the 2023 IPC. In *Proceedings of the 11th International Planning Competition: Planner Abstracts – Hierarchical Task Network (HTN) Planning Track, IPC 2023*.

Höller, D. 2023b. The PANDA Progression System for HTN Planning in the 2023 IPC. In *Proceedings of the 11th International Planning Competition: Planner Abstracts – Hierarchical Task Network (HTN) Planning Track, IPC 2023*.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language Classification of Hierarchical Planning Problems. In *Proceedings of the 21st European Conference on Artificial Intelligence, ECAI 2014*, 447–452. IOS.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the Expressivity of Planning Formalisms through the Comparison to Formal Languages. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS 2016*, 158–165. AAAI.

Höller, D.; and Bercher, P. 2021. Landmark Generation in HTN Planning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 11826–11834. AAAI.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2018. A Generic Method to Guide HTN Progression Search with Classical Heuristics. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling, ICAPS 2018*, 114–122. AAAI.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN Planning as Heuristic Progression Search. *Journal of Artificial Intelligence Research*, 67: 835–880.

Olz, C.; and Bercher, P. 2022. On the Efficient Inference of Preconditions and Effects of Compound Tasks in Partially Ordered HTN Planning Domains. In *Proceedings of the 5th ICAPS Workshop on Hierarchical Planning, HPlan 2022*, 47–51.

Olz, C.; and Bercher, P. 2023. A Look-Ahead Technique for Search-Based HTN Planning: Reducing the Branching Factor by Identifying Inevitable Task Refinements. In *Proceedings of the 16th International Symposium on Combinatorial Search, SoCS 2023*, 65–73. AAAI.

Olz, C.; Biundo, S.; and Bercher, P. 2021. Revealing Hidden Preconditions and Effects of Compound HTN Planning Tasks - A Complexity Analysis. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 11903–11912. AAAI.

Olz, C.; Höller, D.; and Bercher, P. 2023. The PANDADealer System for Totally Ordered HTN Planning in the 2023 IPC. In *Proceedings of the 11th International Planning Competition: Planner Abstracts – Hierarchical Task Network (HTN) Planning Track, IPC 2023*.

Schreiber, D. 2021. Lilotane: A Lifted SAT-based Approach to Hierarchical Planning. *Journal of Artificial Intelligence Research*, 70: 1117–1181.

Wu, Y. X.; Lin, S.; Behnke, G.; and Bercher, P. 2022. Finding Solution Preserving Linearizations For Partially Ordered Hierarchical Planning Problems. In *33rd PuK Workshop "Planen, Scheduling und Konfigurieren, Entwerfen", PuK 2022*.