33rd International Conference on
Automated Planning and Scheduling

July 8 – 13, 2023, Prague (Czech Republic)

IPC 2023

Proceedings of the

Hierarchical Task Network (HTN) Track of the
11th International Planning Competition:
Planner and Domain Abstracts

Preface

Since its first edition in 1998, the International Planning Competition (IPC) has been an integral event of the planning community. For more than 20 years, it established unified input languages for planners and enabled a thorough comparison between them based on an accessible benchmark set. The first two IPCs featured a track on *hand-tailored planners* in which the planners could be provided with additional information or select their algorithms based on the input domain. Among these planners, some used *hierarchical planning*. Many years later, the International Planning Competition 2020 featured for the first time a track dedicated to hierarchical planning. In contrast to the previous track on hand-tailored planners, the question was how well planners can exploit a *given* hierarchical refinement structure. The HTN (*Hierarchical Task Network*) track of the IPC 2023 directly succeeds IPC 2020 as a forum to assess the state of the art in hierarchical planning—specifically, totally and partially ordered HTN planning.

For this iteration of the HTN track, we were able to benefit from the pioneering work made in IPC 2020 that consolidated input languages and benchmark problems—requiring all planners to parse problems in the HDDL (Hierarchical Domain Description Language) format. Since the formalism has gained traction in the research community, we were pleased to receive a total of eleven distinct planner submissions. We were consequently able to evaluate planners across six different sub-tracks: (total order, partial order) $\times$ (satisficing, agile, optimal). While IPC 2020 only featured the *agile* evaluation scheme, which favors short running times over robustness, the satisficing and optimal sub-tracks encourage to report high-quality and entirely optimal plans, respectively (considering the number of primitive tasks / actions as the hierarchical solution's plan length). Employing these quality-focused evaluation metrics is only possible due to planners adopting a common input language and the recent advances made in optimal HTN planning.

The results of this year's HTN track highlight further interesting advancements. After IPC 2020 being dominated by lifted planners, now ground planners with efficient grounding algorithms and well-informed search heuristics have consistently performed the best on the majority of domains. In the partial order track, an approach that attempts to *linearize* problems to totally ordered problems has shown promise. The planner abstracts assembled in the proceedings at hand shed light on those and many further approaches and techniques. We congratulate the winners of the 2023 HTN track and thank all authors for their participation and cooperation.

Ron, Gregor, and Dominik
Organizers of the IPC 2023 HTN track
July 2024

Table of Contents

Overview

Organizing Committee

Ron Alford	MITRE, McLean, Virginia, USA
Gregor Behnke	University of Amsterdam, Netherlands
Dominik Schreiber	Karlsruhe Institute of Technology, Germany

Timeline of the Competition

Call for domains, participation	October 2022
Domain submission deadline	February 28, 2023
Demo problems provided	February 10, 2023
Initial planner submission	February 28, 2023
Feature stop (final planner submission)	May 31, 2023
Planner abstract submission deadline	May 31, 2023
Contest run	June 2023
Results announced	July 12, 2023
Result analysis deadline	August 2023

Input and Output Format

We re-used the input and output specification of IPC 2020. Specifically, domains and problems are formulated in HDDL [14], and the planner is expected to output a hierarchical plan in a specific format that allows its automated verification.[1] For further information, we refer to the respective sections of the IPC 2020 proceedings [5].

Sub-Tracks

We decided to offer six different sub-tracks:

- Total order agile
- Total order satisficing
- Total order optimal
- Partial order agile
- Partial order satisficing
- Partial order optimal

A domain is *totally ordered* iff the subtasks in all methods and in the initial task network form a sequence, i.e. the declared ordering arranges the tasks in a sequence. A domain is *partially*

[1] https://gki.informatik.uni-freiburg.de/ipc2020/format.pdf

ordered iff it is not totally ordered, i.e. there is at least one method whose subtasks are not totally-ordered or the initial task network is not a sequence. The subdivision into totally and partially ordered HTN planning is well-established in the community—totally ordered HTN planning is decidable and more straight forward to handle for heuristics and exploration. The three evaluation schemes *agile*, *satisficing*, and *optimal* are directly taken from the IPC's non-hierarchical tracks [24]. The respective metrics are given in the *Rules* section below. We did not offer a *non-recursive* track due to lack of interest and lack of domains in the last IPC.

Rules

Authors were allowed to submit an arbitrary number (within reason) of *different planning systems* per track. Two planning systems are considered different if the relevant codebase, i.e., the parts of the code that are actually being executed, differs substantially, as judged by the organizers. (This allowed, e.g., for submitting different planning approaches as separate planning systems even if they are part of a common software.) For each planning system submitted, authors were allowed to submit a maximum of *three different configurations.*

Across all tracks, each planner on each problem is executed on a single CPU core for up to 1800 s and is allowed to use up to 8 GB of main memory. All produced plans were verified. If an invalid plan is returned, all tasks in the domain are counted as unsolved. If that happens in more than one domain, the planner is disqualified.

Agile tracks

In the agile tracks, planners are encouraged to find a plan as quickly as possible. Plan quality is ignored; problems solved close to the time limit receive scores close to zero, as do entirely unsolved problems. Specifically, the score of a planner on a solved task is 1 if the task was solved within 1 second; 0 if the task was not solved; and otherwise, if the task was solved in t seconds ($1 \leq t \leq 1800$), its score is $\min\{1, 1 - \frac{\log(t)}{\log(1800)}\}$. The score of a planner is the sum of its scores for all tasks.

Satisficing tracks

The satisficing tracks put a focus on high-quality plans. Planners may return multiple plans (in an anytime manner), but only the one with the lowest cost is counted. The score of a planner on a solved task is the ratio C^*/C where C is the cost of the cheapest discovered plan and C^* is the cost of a reference plan. The score on an unsolved task is 0. The score of a planner is the sum of its scores for all tasks.

Optimal tracks

The optimal tracks require plans of minimal cost. A suboptimal plan in this track is treated just like an invalid plan in the other subtracks. The score attributed to a planner is simply the number of solved tasks.

Domains

We selected 22 TO planning domains and ten PO domains for evaluating planners. While the majority of planning domains were already a part of IPC 2020 (namely 20 TO domains and eight PO domains) [5], four new domains were introduced, namely Lamps, SharpSAT, Ultralight-Cockpit, and Coloring. We dropped the former domains Childsnack, Entertainment, Elevators-Learned, UM-Translog, and Blocksworld-GTOHP and generated more difficult problems for the Snake domain. All benchmarks of IPC 2023 are available online at `https://github.com/ipc2023-htn/ipc2023-domains`.

Total Order Track

- Assembly [4]
- Barman BDI [25]
- Blocksworld HPDDL [3]
- Depots [19]
- Factories simple [23]
- Freecell Learned [16]
- Hiking [19]
- Lamps [6] (page 30–31)
- Logistics Learned [16]
- Minecraft Player [26]
- Minecraft Regular [26]
- Monroe Fully [15]
- Monroe Partially [15]
- Multiarm Blocksworld [3]
- Robot [1]
- Rover GTOHP [19]
- Satellite GTOHP [19]
- SharpSAT [6] (page 30–31)
- Snake [17]
- Towers [2]
- Transport
- Woodworking [22]

Partial Order Track

- Barman BDI [25]
- Monroe Fully [15]
- Monroe Partially [15]
- PCP [13]
- Rover
- Satellite [21]
- Transport
- Ultralight Cockpit [6] (page 30–31)
- Woodworking [22]
- Colouring [6] (page 30–31)

Participants

We received a total of eleven distinct submissions with a varying number of configurations:
Aries [7] (page 7–9); **LiftedLinear**, **LinearSimple**, and **LinearComplex** [27] (page 24–29); **LTP** [20] (page 21–23); **OptiPlan** [9] (page 10–11); **PANDApro** [11] (page 14–15); **PANDA** λ [10] (page 12–13); **PandaDealer** [18] (page 18–20); **SIADEX** [8]; and **TOAD** [12] (page 16–17).
All planners are described in the proceedings at hand, except for SIADEX, which was submitted to IPC 2020 before and has thus already been presented in the IPC 2020 proceedings.

Awards

Sub-track	*Winner*	*Runner-up*
Total order agile	**PandaDealer**-agile-{1,lama}	**PANDA** λ lm-cut
Total order satisficing	**PandaDealer**-agile-lama	**PANDA** λ ao
Total order optimal	**PandaDealer**-optimal	**PANDApro** {lm-cut,dof}
Partial order agile	**LinearSimple**-agile-2	**PANDA** λ {ao,lm-cut}
Partial Order satisficing	**LinearComplex**-satisficing-1	**PANDA** λ {ao,lm-cut}
Partial Order optimal	**PANDApro** lm-cut	**ARIES**

Resources

The webpage `https://ipc2023-htn.github.io/` features full details of the competition, references to all benchmark domains and planner repositories, and detailed results for all configurations, including scores by domain.

The IPC 2023 journal article [24] features a chapter on the HTN track, where we provide some further information and a brief discussion of results.

References

[1] Ron Alford. "HTN IPC-2020 Domain: Robot". In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, p. 32.

[2] Ron Alford. "HTN IPC-2020 Domain: Towers". In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, p. 33.

[3] Ron Alford. "HTN IPC-2020 Domains: Blocksworld-HPDDL and Multiarm-Blocksworld". In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, p. 31.

[4] Gregor Behnke. “AssemblyHierarchical – Connecting Devices through Cables”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 19–20.

[5] Gregor Behnke, Daniel Höller, and Pascal Bercher, eds. *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021. URL: https://ipc2020.hierarchical-task.net/publications/IPC2020Booklet.pdf.

[6] Gregor Behnke, Jane Jean Kiam, and Dominik Schreiber. “New HTN Domains in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 30–31.

[7] Arthur Bit-Monnot. “Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 7–9.

[8] Juan Fernandez-Olivares, Ignacio Vellido, and Luis Castillo. “Addressing HTN Planning with Blind Depth First Search”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 1–4.

[9] Oleksandr Firsov, Humbert Fiorino, and Damien Pellier. “OptiPlan – a CSP-based partial order HTN planner”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 10–11.

[10] Daniel Höller. “The PANDA λ System for HTN Planning in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 12–13.

[11] Daniel Höller. “The PANDA Progression System for HTN Planning in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 14–15.

[12] Daniel Höller. “The TOAD System for Totally Ordered HTN Planning in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 16–17.

[13] Daniel Höller et al. “From PCP to HTN Planning Through CFGs”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 24–25.

[14] Daniel Höller et al. “HDDL – A Language to Describe Hierarchical Planning Problems”. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*. AAAI Press, 2020, pp. 9883–9891.

[15] Daniel Höller et al. “Plan and Goal Recognition as HTN Planning”. In: *Proceedings of the 30th International Conference on Tools with Artificial Intelligence (ICTAI 2018)*. IEEE Computer Society, 2018, pp. 466–473.

[16] Damir Lotinac, Filippos Kominis, and Anders Jonsson. “Hierarchical Task Networks Generated Using Invariant Graphs for IPC2020”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 26–30.

[17] Maurício Cecílio Magnaguagno. “Snake Domain for HTN IPC 2020”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 37–38.

[18] Conny Olz, Daniel Höller, and Pascal Bercher. “The PandaDealer System for Totally Ordered HTN Planning in the 2023 IPC”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 18–20.

[19] Damien Pellier and Humbert Fiorino. “From Classical to Hierarchical: Benchmarks for the HTN Track of the International Planning Competition”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 21–23.

[20] Gaspard Quenard, Damien Pellier, and Humbert Fiorino. “LTP: Lifted Tree Path”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 21–23.

[21] Bernd Schattenberg. “The Hierarchical Satellite Domain”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 40–42.

[22] Bernd Schattenberg and Pascal Bercher. “The Hierarchical Woodworking Domain”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 43–44.

[23] Malte Sönnichsen and Dominik Schreiber. “The HTN Domain “Factories””. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, pp. 45–46.

[24] Ayal Taitler et al. *The 2023 International Planning Competition*. 2024. DOI: 10.1002/aaai.12169.

[25] Max Waters, Lin Padgham, and Sebastian Sardina. “The Barman-HTN Domain for IPC 2020”. In: *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*. 2021, p. 39.

[26] Julia Wichlacz, Alvaro Torralba, and Jörg Hoffmann. “Construction-Planning Models in Minecraft”. In: *Proceedings of the Second ICAPS Workshop on Hierarchical Planning*. 2019, pp. 1–5.

[27] Ying Xian Wu et al. “Grounded (Lifted) Linearizer at the IPC 2023: Solving Partial Order HTN Problems by Linearizing Them”. In: *Proceedings of the Hierarchical Task Network (HTN) Track of the 11th International Planning Competition (IPC 2023): Planner and Domain Abstracts*. 2024, pp. 24–29.

Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC

Arthur Bit-Monnot
LAAS-CNRS, Université de Toulouse, INSA, CNRS, Toulouse, France
abitmonnot@laas.fr

Abstract

In this paper we give a high level overview of the ARIES planner at the time of its participation in the hierarchical track of the 2023 *International Planning Competition (IPC)*.

ARIES is an experimental solver whose aim is to jointly evolve a combinatorial solver for scheduling-like problems and an encoding of task planning that exploits it.

Overview

Focus The focus of ARIES is on optimization planning problems with rich temporal and concurrency requirements, driven by applications in robotics and logistics. As such, the problems targeted by the 2023 IPC on hierarchical planning are certainly out of its comfort zone. Nevertheless, with ARIES participation in the IPC we hope to highlight the relative weaknesses and hopefully strengths of its approach.

As HDDL is not its primary target, at the time of its submission, very little effort has been dedicated to improving the performance of ARIES on a corpus of HDDL problems. This participation is also the occasion of setting a milestone and reference point to be used as a baseline for future improvements.

Search Space In terms of search space, ARIES pertains to the family of lifted plan-space planners like IxTeT (Ghallab and Laruelle 1994) and FAPE (Bit-Monnot et al. 2020). In practice, it means that the solver does not construct snapshots of the entire state at various epochs but instead reasons on the interactions and dependencies between individual actions (through threats and causal links, Ghallab, Nau, and Traverso 2004). In addition, being lifted implies that the problem is never grounded: instead each action of the plan is parameterized with variables that must obey a set of constraints, notably reflecting the causal requirements of the action. The solver must eventually impose a value for these parameters that ensures that the plan is valid and achieves the intended objectives.

Search Strategy Unlike IxTeT and FAPE, ARIES delegates the responsibility of search to an independent solver. We adopt the approach of generating a CSP of bounded size that is submitted to a combinatorial solver, as common in SAT-based planners such as Lilotane (Schreiber 2021). ARIES uses in own specialized solver: a hybrid CP-SAT combinatorial solver for optional scheduling.

At a very high level, the planner adopts the following procedure:

1. Parse HDDL problem files and translates them into chronicles, where each chronicle consists of a collection of timed conditions, effects and subtasks linked by shared variables and constraints.
2. From the objective tasks of the problem, build the decomposition tree with a maximum depth (initially 1).
3. Encode the decomposition tree into a Constraint Satisfaction Problem (CSP)
4. Solve the CSP with the internal combinatorial solver
5. If the CSP was proven unsatisfiable, repeat from step 2 with an increased maximum depth.

Encoding HDDL

The planning model used internally by ARIES is based on *chronicles* (Ghallab, Nau, and Traverso 2004), a formalism to compactly represent requirements and effects of a process in time. In planning, chronicles find their first usage in IxTeT (Ghallab and Laruelle 1994) and have been then extended for hierarchical planning in FAPE (Bit-Monnot et al. 2020).

Translating HDDL to chronicles is fairly straightforward as they are for the most part strictly more expressive. In practice, it mostly consists in assigning temporal semantics to a non-temporal language. An example of a translation of an HDDL method into a chronicle is given in Figure 1.

Given a problem representation as chronicles, we encode it as a CSP using the formulae of Godet and Bit-Monnot (2022). In spirit, this encoding corresponds to a lifted plan-space representation. For each action/mehtod node of the lifted decomposition tree, it introduces decision variables representing *(i)* the presence of the action/method in the solution plan, *(ii)* its parameters and *(iii)* its start and end times. These decision variables are linked by a set of constraints, notably:

- *Decomposition constraints* that force the refinement of all tasks with exactly one method or action.
- *Support constraints* that require each condition to be either absent or supported by an effect.

```
(:method m8_send_soil_data
  :parameters (
    ?x - rover
    ?from - waypoint
    ?l - lander
    ?w1 - waypoint
    ?w2 - waypoint)
  :task (send_soil_data ?x ?from)
  :precondition (and (at_lander ?l ?w2)
                     (visible ?w1 ?w2))
  :ordered-subtasks (and
    (t1 (do_navigate1 ?x ?w1))
    (t2 (comm_soil_data1 ?x ?l ?from ?w1 ?w2))
      ))
```

(a) HDDL methods fo the *rovers* domain.

m8-send-soil-data

variables:	$x, \mathit{from}, l, w_1, w_2$ (parameters)
	$t_s, t_e, t_s^1, t_e^1, t_s^2, t_e^2$ (timepoints)
task:	$[t_s, t_e]$ *send-soil-data*(x, from)
conditions:	$[t_{start}]$ *at-lander*$(l, w_2) =$ true
	$[t_{start}]$ *visible*$(w_1, w_2) =$ true
subtasks:	$[t_s^1, t_e^1]$ *do-navigate1*(x, w_1)
	$[t_s^2, t_e^2]$ *comm-soil-data1*$(x, l, \mathit{from}, w_1, w_2)$
constraints:	$t_s = \mathit{min}(t_s^1, t_s^2)$
	$t_e = \mathit{max}(t_e^1, t_e^2)$
	$t_e^1 < t_s^2$ (total order)

(b) Equivalent chronicle representation.

Figure 1: HDDL and chronicle representation of an HTN method of the *rovers* domains

- *Coherence constraints* that ensure that each state variable is never given more than on one value at a time.

Compared to SAT-based encodings for HTN, the size of this encoding is mostly independent of the size of the plan and the number of predicates in the state. On the other hand, it grows mostly quadratically with the size of the lifted decomposition tree, because of the support and coherence constraints.

Matching PDDL semantics Despite the overall compatibility of chronicles with $\{$HP$\}$DDL, several peculiarities of PDDL must be explicitly handled.

PDDL introduces the concept of *mutex actions* to forbid interfering actions from been executed at the same time. This is required in PDDL due to the instantaneous nature of actions but introduces an awkwardness for a temporal model as the handling of a condition statement depends on the place where it was asserted. To handle such cases, we extend the constraints of Godet and Bit-Monnot (2022) with *mutex constraints* that forbid a condition's interval to meet the start of an effect from another action chronicle. Note that mutexes induced by pairs of *effects* are already enforced by *coherence constraints*.

For historical reasons, the *delete-before-add* semantics of PDDL stipulates that if a single action assigns both *true* and *false* to a common boolean state-variable, then only the positive assignment should be considered. This violates the principle that a state variable should not be assigned two variables at the same time enforced by the *coherence constraints*. ARIES does not support this particular corner case of the PDDL semantics as handling it properly in a fully lifted representation would notably complexify the encoding and may induce non-negligible runtime penalties. As a result, ARIES will consider inapplicable any action that relies on the *delete-before-add* semantics, making it incomplete in the presence of such problems. This is for instance the case of the *woodworking* domain where ARIES is unable to find a solution.

Finally, support for HDDL syntax remains partial. For instance, the multiple inheritance of types exploited in the *UMTranslog* domain of IPC 2020 is unsupported and ARIES would refuse to parse the problem.

Combinatorial Solver

The encoding of Godet and Bit-Monnot (2022) is generic and could in theory exploit any solver capable of representing disjunctions and reified difference constraints, which is the case of most CP, SMT or MILP solvers. Our experience with existing solvers for this encoding is however lukewarm.

Previous experiences

First let us state that the highly combinatorial structure of the encoding does not appear to play well with the linear relaxations of MILP solvers that were at difficulty even for the simplest instances.

Experiments with the Z3 SMT solver (De Moura and Bjørner 2008), showed interesting results with reasonable performance on non-hierarchical domains (Bit-Monnot 2018). Most impressive in particular was the ability of the activity-based search of SAT/SMT solver to robustly converge to a solution or prove its absence without any planning-specific knowledge. On the other hand, SMT-based solvers remain hard to tune with limited support for optimization. While Z3 excelled at proving unsatisfiability, the runtime to a first solution remained very high even on simple problems.

Our own experiments with CPOptimizer (Laborie et al. 2018), a leading CP solver for scheduling, showed that it had complementary strength: fine-tuning and strong propagation allowed it to quickly converge to a first solution and incrementally improve it. A key feature for strong propagation was a native support of optional activities, that are ubiquitous in task planning. CPOptimizer may however struggle to prove unsatisfiability or escaping dead-ends in its search space in situations that would be easily handled by the conflict-directed clause learning of SAT/SMT solvers.

While our use of existing solvers clearly showed limitations, FAPE provided an alternate model. FAPE uses best-search in the space of lifted partial plans, with each partial plan encoded as a CSP that is iteratively refined and constrained as search progresses. This CSP was handled with a custom propagation engine developed within the solver code base. Our experience is that developing side-by-side a propagation engine and the planner that uses it, allowed us to identify and address fundamental limitations of the propagation engines. For FAPE, this resulted in very strong propagation engine and compact search space (Bit-Monnot et al. 2020).

ARIES **Scheduler**

Following a similar route, ARIES relies on its own combinatorial solver that could be captioned as a *hybrid CP-SAT* solver for *optional scheduling*. Its most distinctive features are the following:

- The core of a SAT solver, (clause learning, unit propagation and activity-based search) integrated with finite-domain constraint solver.
- Dedicated reasoner for difference logic (aka. Disjunctive Temporal Networks), that notably enables conflict detection and explanations for temporal constraints.
- First-hand support of optional activities, enabling eager propagation of the potential timing and parameters of activities that are not yet part of the plan

While the solver is still in its early days, preliminary evaluation shows that it has state-of-the-art performance on disjunctive scheduling problems such as the jobshop and openshop scheduling problems (Bit-Monnot 2023).

Technical Remarks

ARIES is implemented from the ground up in Rust and is freely available under the MIT license at https://github.com/plaans/aries. Most of the code base is dedicated to the implementation of our hybrid CP-SAT solver. Comparatively, planning specific code only occupies a small fraction of the code base, mostly dedicated to straightforward parsing and encoding tasks.

Beside its partial support for HDDL, ARIES also provides an integration as a backend for the *unified-planning* library with richer modeling features. Direct encoding of planning problems as chronicles or as a CSP is another way to exploit ARIES in a more flexible way.

Acknowledgments

This work has been partially supported by AIPlan4EU, a project funded by EU Horizon 2020 research and innovation program under GA n.101016442.

References

Bit-Monnot, A. 2018. A Constraint-Based Encoding for Domain-Independent Temporal Planning. In *International Conference on Principles and Practice of Constraint Programming (CP)*.

Bit-Monnot, A. 2023. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. In *Under review at ECAI*.

Bit-Monnot, A.; Ghallab, M.; Ingrand, F.; and Smith, D. E. 2020. FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning. arXiv:2010.13121.

De Moura, L.; and Bjørner, N. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Ghallab, M.; and Laruelle, H. 1994. Representation and Control in IxTeT, a Temporal Planner. In *International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*.

Godet, R.; and Bit-Monnot, A. 2022. Chronicles for Representing Hierarchical Planning Problems with Time. In *ICAPS Workshop on Hierarchical Planning (HPlan)*.

Laborie, P.; Rogerie, J.; Shaw, P.; and Vilím, P. 2018. IBM ILOG CP Optimizer for Scheduling. *Constraints*.

Schreiber, D. 2021. Lilotane: A Lifted SAT-based Approach to Hierarchical Planning. *Journal of Artificial Intelligence Research (JAIR)*.

OptiPlan - a CSP-based partial order HTN planner

Oleksandr Firsov Humbert Fiorino Damien Pellier
Univ. Grenoble Alpes - LIG
Grenoble, France
oleksandr.firsov humber.fiorino damien.pellier @univ-grenoble-alpes.fr

Introduction

In this paper, we introduce OptiPlan, a planner that participated in partial-order HTN track of IPC 2023. OptiPlan is a hierarchical planner capable of solving partial-order problems by encoding them as CSPs [1] and generating partial order solution plans.

Encoding resolution techniques, particularly SAT encodings, have proved their worth in various IPC competitions. Solving HTN problems via CSP was little studied [2] compared to SAT [3]. These techniques, however, offer a number of advantages over SAT: (1) CSP encoding provides a natural way of expressing numerical constraints, which is not possible with SAT; (2) CSP encoding lets naturally express constraints on method decompositions (logical or numerical) and (3) CSP techniques are much more mature than SAT, and are covered my multiple industrial-level solvers[4].

The most distinct, property of OptiPlan is its capability to produce partial-ordered solution plans. The reasoning behind this feature is twofold. First, in various contexts, oftentimes industrial, it is crucial to make plans as flexible as possible, as it helps anticipate unforeseeable events [5][6]. A natural answer to this are partial-ordered solutions, which allow shifting tasks without any impact on the quality of the plan. Second, unlike deordering of total-order plans [7] which cannot guarantee quality, or optimality, of the produced plans, OptiPlan can guarantee these properties.

Optiplan Principle

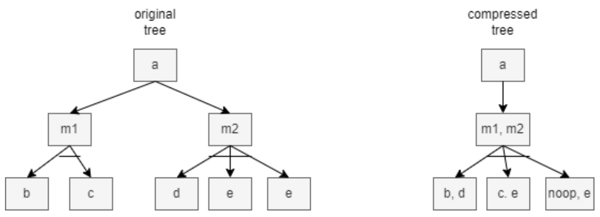
OptiPlan is based upon Task Decomposition Graph (TDG) [8] structure and hybrid planning formalization [9], which combines the concepts of HTN with POCL [10]. The difference being that our search space is a tree, instead of graph, where:

- **OR** nodes represent possible method decompositions
- **AND** nodes represent abstract and primitive tasks

Only tasks can be *leaves*, and only primitive tasks are considered *terminal* leaves.

Constructing a complete search space is infeasible in HTN planning[11], so OptiPlan operates using an iterative deepening search. Initially, our tree consists of an artificial root and its children: abstract tasks of initial HTN, as well as dummy primitive tasks t_0 and t_∞, that correspond to the initial and goal states. Along with the tree, we keep track of ordering constraints introduced by initial task network and method decompositions.

Figure 1: Example of TDG compression



In this search space, we attempt to find a subtree, such that:

1. It has exclusively terminal leaves (i.e., plan is concrete)
2. Every precondition of the subtree has a causal link supporting it (i.e., no open goals)
3. There are no threats on the causal links
4. There are no conflicting ordering constraints

If a solution can't be found, it means that the search space is not big enough to support it. So we update the search space by expanding the non-terminal leaves of the tree, and try to solve the problem again. This process is repeated until either a solution is found, or failure termination conditions are met (e.g., there are no abstract leaves left), in which case problem is deemed unsolvable.

Implementation

As our planner requires a grounded instance of the problem, we use PDDL4J[12] to parse, pre-process, and instantiate it.

To fully benefit from numeric variables, we perform a compression procedure on the tree, where we attempt to merge mutex nodes from the same depth level into a single node. This can be best explained on a simple example in Fig. 1, where we compress two methods *m1, m2* of an abstract task a.

To find the solution, OptiPlan uses Chuffed[13] as its CSP solver.

References

[1] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557–581, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221798003646

[2] P. Surynek and R. Barták, "Encoding htn planning as a dynamic csp," in *Principles and Practice of Constraint Programming-CP 2005: 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005. Proceedings 11*. Springer, 2005, pp. 868–868.

[3] I. Georgievski and M. Aiello, "Htn planning: Overview, comparison, and beyond," *Artificial Intelligence*, vol. 222, pp. 124–156, 2015.

[4] J.-F. Puget, "Constraint programming next challenge: Simplicity of use," in *Principles and Practice of Constraint Programming–CP 2004: 10th International Conference, CP 2004, Toronto, Canada, September 27-October 1, 2004. Proceedings 10*. Springer, 2004, pp. 5–8.

[5] D. Liu, H. Wang, C. Qi, P. Zhao, and J. Wang, "Hierarchical task network-based emergency task planning with incomplete information, concurrency and uncertain duration," *Knowledge-Based Systems*, vol. 112, pp. 67–79, 2016.

[6] D. Liu, H. Li, J. Wong, and M. Khallaf, "Hierarchical task network approach for time and budget constrained construction project planning," *Technological and Economic Development of Economy*, vol. 25, pp. 1–24, 04 2019.

[7] C. Muise, S. McIlraith, and C. Beck, "Optimally relaxing partial-order plans with maxsat," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 22, 2012, pp. 358–362.

[8] P. Bercher, G. Behnke, D. Höller, and S. Biundo, "An admissible htn planning heuristic." in *IJCAI*, 2017, pp. 480–488.

[9] S. Biundo and B. Schattenberg, "From abstract crisis to concrete relief—a preliminary report on combining state abstraction and htn planning," in *Sixth European Conference on Planning*, 2001.

[10] D. McAllester and D. Rosenblatt, "Systematic nonlinear planning," 1991.

[11] R. Alford, V. Shivashankar, U. Kuter, and D. Nau, "Htn problem spaces: Structure, algorithms, termination," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 3, no. 1, 2012, pp. 2–9.

[12] D. Pellier and H. Fiorino, "Pddl4j: a planning domain description library for java," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 30, no. 1, pp. 143–176, 2018.

[13] G. Chu, P. J. Stuckey, A. Schutt, T. Ehlers, G. Gange, and K. Francis. (2016) Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed.

The PANDA λ System for HTN Planning in the 2023 IPC

Daniel Höller

Saarland University, Saarland Informatics Campus,
Saarbrücken, Germany
hoeller@cs.uni-saarland.de

Abstract

The PANDA λ system is an HTN planning system that can handle both totally ordered and partially ordered HTN models. It performs a progression search, i.e., it only processes tasks without predecessor in the task network and is based on the PANDA framework. PANDA λ uses a graph search and guides search by using a combination of heuristics and landmarks. These are combined by using a multi-fringe system.

Introduction

PANDA λ (La<u>n</u>d<u>m</u>ark-<u>b</u>ased PAN<u>DA</u>) is a planning system from the PANDA framework (Höller et al. 2021), which can handle both totally ordered and partially ordered models.

Search-based systems in HTN planning can be divided into plan space-based systems and progression-based systems (see Bercher, Alford, and Höller, 2019). The latter only process tasks without predecessor in the task ordering of the current task network. PANDA λ is based on the systematic progression search introduced by Höller et al. (2020).

It uses the preprocessing stack of the PANDA framework: HDDL (Höller et al. 2020) as input language and by the grounding procedure introduced by Behnke et al. (2020).

During search, PANDA λ maintains a black-list of already visited nodes and processes every node only a single time, i.e., it uses a graph search. While this is (from a computational perspective) no problem in totally ordered HTN planning, it gets a task as hard as graph isomorphism in partially ordered HTN planning. To do it efficiently, PANDA λ uses the techniques introduced by Höller and Behnke (2021), which apply several techniques for hashing search nodes, and exploit certain special cases present in many models of the commonly used benchmark sets.

PANDA λ guides its search by using a combination of landmarks (Höller and Bercher 2021) and heuristics from the family of Relaxed Composition (RC) heuristics (Höller et al. 2018, 2019, 2020) to estimate the goal distance.

Similar to the LAMA system from classical planning (Richter and Westphal 2010), PANDA λ combines these in a multi-fringe search, where one fringe is sorted by a RC heuristic, and one by the LM-count heuristic computed on the landmarks. The system extracts nodes from the fringes in turn and each successor node is inserted into both fringes with the respective heuristic estimate.

We next describe the RC heuristics and the used landmarks afterwards. Each configuration of our overall system combines one of the two RC heuristics with one of the two landmark sets.

RC Heuristics

The family of RC heuristics (Höller et al. 2018, 2019, 2020) uses classical heuristics to estimate the goal distance during HTN search. To do so, it relaxes the HTN model to a classical model which is only used for heuristic calculation. It is created in a way that the set of solutions increases compared to the HTN model. HTN planning starts with the initial task(s) and decomposes them until only actions are left. This process can be seen as the building process of a tree. The classical RC model maintains which tasks are part of that tree, but in a bottom-up manner, *compositing* tasks. When an action from the original HTN is applied in this model, it is marked as part of the tree. Methods are represented in the RC model by special actions. These are applicable when all subtasks of the method are part of the tree. When they are applied, the decomposed task is marked as part of the tree. The goal of the overall problem is to mark the tasks in the current task network as being part of the tree.

This encoding solves several problems when translating HTN models to classical models. First, we always have a state-based goal (which is not the case in HTN models): adding the current tasks to the tree. Second, the model is also informed about applicability of actions, since actions can only be added when they are applicable. Like in other HTN heuristics, the encoding allows for task insertion (adding further actions apart from the decomposition hierarchy) to make actions applicable that are needed elsewhere. However, what is interesting about our encoding when compared to other heuristics (see e.g. Bercher et al., 2017), is that the costs of these added actions are incorporated into the heuristic value. In our implementation, we further restrict task insertion to those actions still reachable via decomposing the current task network. Third, our heuristic is – to some extend – informed about the decomposition process, because the tree must be created up to the current tasks.

Practically, the model can be updated instead of recomputed. The only things that need to be changed are the initial state and the goal condition of the RC model. The model is linear in the size of the HTN model, and can be combined with any classical heuristic. However, the update of the goal is not possible (efficiently) in every classical heuristic.

In the IPC, we combine it with the Add (Bonet and Geffner 2001) heuristic and with the FF (Hoffmann and Nebel 2001) heuristic.

Landmark Generation

In the IPC, we use two types of landmarks, RC-based and AND/OR landmarks, which are described in this section.

RC-based Landmarks

The first type of landmarks computes the LM-Cut heuristic (Helmert and Domshlak 2009) on the RC model of the initial search node. The generated landmarks are stored and tracked during search.

AND/OR Landmarks

The second type of landmarks is generated using the approach of Höller and Bercher (2021). It extends an approach from classical planning by Keyder, Richter, and Helmert (2010), who represent a delete-free classical planning problem as AND/OR graph, and extract landmarks from this graph afterwards. We extend the AND/OR graph to also represents parts of the decomposition hierarchy, and applies the unchanged extraction algorithm afterwards.

In contrast to the classical case, the HTN representation comes with more relaxations than only delete-relaxation. E.g., no ordering relations from the HTN model are represented in the graph.

We generate the landmarks on the initial search nodes and track them afterwards during search.

References

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9775–9784. AAAI Press.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 6267–6275. IJCAI organization.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 480–488. IJCAI organization.

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

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)*.

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.; and Behnke, G. 2021. Loop Detection in the PANDA Planning System. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *Künstliche Intelligenz*, 30(1): 11–20.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9883–9891. AAAI Press.

Höller, D.; and Bercher, P. 2021. Landmark Generation in HTN Planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 11826–11834. AAAI Press.

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)*, 114–122. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6171–6175. IJCAI organization.

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

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, 335–340. IOS Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39: 127–177.

The PANDA Progression System for HTN Planning in the 2023 IPC

Daniel Höller
Saarland University, Saarland Informatics Campus,
Saarbrücken, Germany
hoeller@cs.uni-saarland.de

Abstract

The PANDA Progression System is an HTN planning system that can handle both totally ordered and partially ordered HTN models. It performs a progression search, i.e., it only processes tasks without predecessor in the task network. PANDA uses a graph search and guides search by using heuristics. The configurations for the IPC use the families of Relaxed Composition (RC) heuristics and Delete- and Ordering-Relaxation (DOR) heuristics. RC heuristics relax the HTN model to a classical model and apply heuristics from classical planning to compute heuristic values. This way, also admissible heuristics for optimal planning can be created. The family of DOR heuristics originally capture delete- and ordering-free HTN planning as IP. This basic encoding can be extended by other IP constraints, e.g. encoding landmarks.

Introduction

The PANDA progression (PANDApro) system is a planner from the PANDA framework (Höller et al. 2021), which can handle both totally ordered and partially ordered models.

Search-based systems in HTN planning can be divided into plan space-based systems and progression-based systems (see Bercher, Alford, and Höller, 2019). The latter only process tasks without predecessor in the task ordering of the current task network. PANDApro uses the systematic progression search introduced by Höller et al. (2020).

It uses the common preprocessing stack of the PANDA framework: HDDL (Höller et al. 2020) as standard input language, followed by the grounding procedure introduced by Behnke et al. (2020).

During search, PANDApro maintains a black-list of already visited nodes and processes every node only a single time, i.e., it uses a graph search. While this is (from a computational perspective) no problem in totally ordered HTN planning, it gets a task as hard as graph isomorphism in partially ordered HTN planning. To do it efficiently, PANDApro uses the techniques introduced by Höller and Behnke (2021), which apply several techniques for hashing search nodes, and exploit certain special cases present in many models of the commonly used benchmark sets.

PANDApro guides its search by using heuristics estimating the goal distance (or the remaining costs in case of optimal planning). For the IPC, it uses two families of heuristics, which are described in the following.

RC Heuristics

The family of relaxed composition (RC) heuristics (Höller et al. 2018, 2019, 2020) uses classical heuristics to estimate the goal distance during HTN search. To do so, it relaxes the HTN model to a classical model which is only used for heuristic calculation. It is created in a way that the set of solutions increases compared to the HTN model. HTN planning starts with the initial task(s) and decomposes them until only actions are left. This process can be seen as the building process of a tree. The classical RC model maintains which tasks are part of that tree, but in a bottom-up manner, *compositing* tasks. When an action from the original HTN is applied in this model, it is marked as part of the tree. Methods are represented in the RC model by special actions. These are applicable when all subtasks of the method are part of the tree. When they are applied, the decomposed task is marked as part of the tree. The goal of the overall problem is to mark the tasks in the current task network as being part of the tree.

This encoding solves several problems when translating HTN models to classical models. First, we always have a state-based goal (which is not the case in HTN models): adding the current tasks to the tree. Second, the model is also informed about applicability of actions, since actions can only be added when they are applicable. Like in other HTN heuristics, the encoding allows for task insertion (adding further actions apart from the decomposition hierarchy) to make actions applicable that are needed elsewhere. However, what is interesting about our encoding when compared to other heuristics (see e.g. Bercher et al., 2017), is that the costs of these added actions are incorporated into the heuristic value. In our implementation, we further restrict task insertion to those actions still reachable via decomposing the current task network. Third, our heuristic is – to some extend – informed about the decomposition process, because the tree must be created up to the current tasks.

Practically, the model can be updated instead of recomputed. The only things that need to be changed are the initial state and the goal condition of the RC model. The model is linear in the size of the HTN model, and can be combined with any classical heuristic. However, the update of the goal is not possible (efficiently) in every classical heuristic.

In the IPC, we combine it with the Add (Bonet and Geffner 2001), the FF (Hoffmann and Nebel 2001), and the LM-Cut (Helmert and Domshlak 2009) heuristic. We have shown that the combination of the RC model with an admissible heuristic from classical planning results in an admissible HTN heuristic, so we use the latter (RC with LM-Cut) for optimal planning.

DOR Heuristics

In HTN planning, finding a delete-relaxed solution as done by many classical heuristics is still NP-hard (Alford et al. 2014). To make heuristic computation feasible, a common additional relaxation made by HTN heuristics is *task insertion*. As already discussed for RC heuristics, this means that the planner (or heuristic) is allowed to add actions apart from the hierarchy.

In our work on Delete- and Ordering-Free HTN planning (Höller, Bercher, and Behnke 2020), we introduce the class of HTN models that do not include delete-effects nor ordering constraints between tasks in the methods and in the initial task network. We show that the resulting problem is still NP-hard to solve. Then we show how to (exactly) encode this problem into an integer linear program (IP), combining constraints describing the decomposition process and constraints describing a relaxed planning graph (Imai and Fukunaga 2015).

The heuristic used here builds on this line of work. We combine (a further relaxed version of) the constraints describing HTN decomposition with constraints encoding HTN landmarks, and operator counting constraints (Pommerening et al. 2014). We use two configurations, one using LM-Cut landmarks generated on the RC model, and one using the AND/OR HTN landmarks introduced by Höller and Bercher (2021).

Instead of solving the IP, we further use the relaxation to a linear model to make computation polynomial.

References

Alford, R.; Shivashankar, V.; Kuter, U.; and Nau, D. S. 2014. On the Feasibility of Planning Graph Style Heuristics for HTN Planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS*. AAAI press.

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9775–9784. AAAI Press.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 6267–6275. IJCAI organization.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 480–488. IJCAI organization.

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

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)*.

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.; and Behnke, G. 2021. Loop Detection in the PANDA Planning System. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *Künstliche Intelligenz*, 30(1): 11–20.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9883–9891. AAAI Press.

Höller, D.; and Bercher, P. 2021. Landmark Generation in HTN Planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 11826–11834. AAAI Press.

Höller, D.; Bercher, P.; and Behnke, G. 2020. Delete- and Ordering-Relaxation Heuristics for HTN Planning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 4076–4083. IJCAI organization.

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)*, 114–122. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6171–6175. IJCAI organization.

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

Imai, T.; and Fukunaga, A. 2015. On a Practical, Integer-Linear Programming Model for Delete-Free Tasks and its Use as a Heuristic for Cost-Optimal Planning. *Journal of Artificial Intelligence Research*, 54: 631–677.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-Based Heuristics for Cost-Optimal Planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI press.

The TOAD System for Totally Ordered HTN Planning in the 2023 IPC

Daniel Höller
Saarland University, Saarland Informatics Campus,
Saarbrücken, Germany
hoeller@cs.uni-saarland.de

Abstract

The TOAD system is a translation-based planning system for totally ordered HTN planning. It translates a given HTN planning problem into a classical planning problem. To overcome the differences in expressiveness, it does not bound the problem like other translation-based systems, but approximates the problem instead by modifying the decomposition hierarchy such that the set of solutions increases. Then we encode it as classical planning problem and solve it using classical planners. To ensure that only solutions for the original HTN problem are returned, we apply HTN plan verification.

Introduction

The TOAD (Totally Ordered HTN Approximation using DFA) system (Höller 2021) is a translation-based planning system for totally ordered HTN planning. It translates a given HTN planning problem into a classical problem and uses classical planners to solve it. While translation-based systems from the literature *bound* the HTN problem to overcome the differences in expressiveness (Alford, Kuter, and Nau 2009; Alford et al. 2016; Behnke et al. 2022), TOAD over-approximates the set of solutions. I.e., all solutions to the HTN planning problem are also solutions for the classical problem, but the latter might have more.

The approach is inspired by our results on the expressiveness of planning formalisms (Höller et al. 2014, 2016). The set of solutions to a totally ordered HTN planning problem can be seen as the intersection of two languages: a context-free language describing which action sequences can result from the decomposition process, and a regular language describing which action sequences are applicable and lead to a goal state in the transition system induced by the non-hierarchical part of the HTN problem (actions/state). TOAD uses techniques from the field of formal languages (Nederhof 2000a,b) to create a finite automaton (FA) accepting the words of the context-free language (which might require approximation), which is then combined with the non-hierarchical part of HTN problem.

The TOAD System

We use the preprocessing of the PANDA framework for hierarchical planning (Höller et al. 2021), i.e., HDDL as input language (Höller et al. 2020) and the PANDA grounder to ground the model (Behnke et al. 2020).

Figure 1 illustrates the overall TOAD system, which is described in the following.

Analysis. First, TOAD analyzes whether the HTN problem can be translated exactly or approximation is needed. This is done based on a criterion from formal languages called *self-embedding* (Chomsky 1959), which is checked on the decomposition rules (i.e., the methods). We first construct the decomposition graph, i.e., a graph with the tasks of the problem as nodes in which two nodes c_a and c_b are connected by a directed edge (c_a, c_b) when there is a method decomposing c_a into a task sequence including c_b. We compute the strongly connected components (sccs) of this graph. A problem is self-embedding if there is a scc N_i such that

- there is a method $(c_a, \alpha c_b \beta)$, $c_a, c_b \in N_i$ and $\alpha \neq \varepsilon$ and
- there is a method $(c_a, \alpha c_b \beta)$, $c_a, c_b \in N_i$ and $\beta \neq \varepsilon$.

When a problem is *not* self-embedding, this is a sufficient criterion that it describes a regular language, which for us means that approximation is not needed.

Approximation. When approximation is needed, the grammar rules (methods) are modified such that the set of solutions increases. We use an approximation introduced by Nederhof (2000a; 2000b).

Consider a grammar $G = (\mathcal{C}, \mathcal{A}, M, c_I)$ with the non-terminal and terminal symbols $\mathcal{C} = \{A\}$ and $\mathcal{A} = \{a, b\}$, the production rules $M = \{(A, b), (A, aAa)\}$, and the start symbol $c_I = A$. It describes the context-free language $\{a^n \; b \; a^n \mid n \geq 0\}$. The approximation disconnects the part left and right of the b, resulting in a grammar generating the language $\{a^n \; b \; a^m \mid n, m \geq 0\}$, which is regular.

Based on the method by Nederhof (2000a; 2000b) we construct a finite automaton (FA) accepting action sequences derivable via the (maybe modified) hierarchy.

Classical Encoding. Based on the FA and the non-hierarchical part of the HTN planning problem (actions/state) we build a classical planning problem.

Solving. We use the Fast Downward (FD) planning system (Helmert 2006) to solve the resulting problems. We use a multi-fringe configuration similar to the classical LAMA system (Richter and Westphal 2010) with two fringes. One

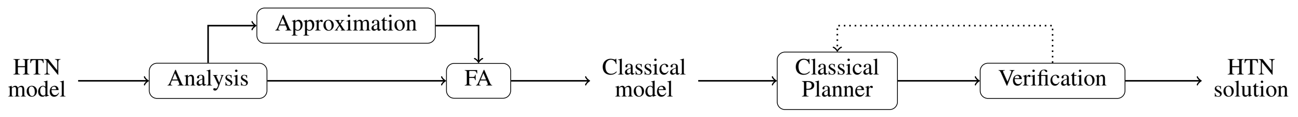


Figure 1: Schema of the TOAD system (based on Figure 1 by Höller, 2021).

uses the FF heuristic (Hoffmann and Nebel 2001) and also its helpful actions, and one uses a precomputed heuristic returning the distance from the current state in the FA to the nearest goal state in the FA as heuristic value.

Verification. To return only valid solutions for the original HTN problem, HTN plan verification is applied as last step. We use the verification system introduced by Höller et al. (2022) in combination with the PANDA progression planner (Höller et al. 2018, 2020).

We modified FD to verify a solution before returning it. When it is not valid, search is continued.

Discussion

While the basic approach is sound and complete, the combination with a graph search like used by FD leads to an incomplete overall system. This is caused by the fact that such a system does not (eventually) return *every* solution to the underlying classical problem. Whenever a particular state in the search space needs to be visited twice before a solution for the HTN problem is found, TOAD will fail. However, at least on the benchmark set of the last (i.e., 2020) IPC, this seems not to be an issue. To the contrary, in this set most problems can be translated without using the approximation.

References

Alford, R.; Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; and Aha, D. 2016. Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 20–28. AAAI Press.

Alford, R.; Kuter, U.; and Nau, D. S. 2009. Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1629–1634.

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9775–9784. AAAI Press.

Behnke, G.; Pollitt, F.; Höller, D.; Bercher, P.; and Alford, R. 2022. Making Translations to Classical Planning Competitive with Other HTN Planners. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, 9687–9697. AAAI Press.

Chomsky, N. 1959. On Certain Formal Properties of Grammars. *Information and Control*, 2(2): 137–167.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

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. 2021. Translating Totally Ordered HTN Planning Problems to Classical Planning Problems Using Regular Approximation of Context-Free Languages. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, 159–167. AAAI Press.

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)*, 447–452. IOS Press.

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)*, 158–165. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *Künstliche Intelligenz*, 30(1): 11–20.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9883–9891. AAAI Press.

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)*, 114–122. AAAI Press.

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

Höller, D.; Wichlacz, J.; Bercher, P.; and Behnke, G. 2022. Compiling HTN Plan Verification Problems into HTN Planning Problems. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS)*, 145–150. AAAI Press.

Nederhof, M.-J. 2000a. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1): 17–44.

Nederhof, M.-J. 2000b. Regular approximation of CFLs: A grammatical view. In *Advances in Probabilistic and other Parsing Technologies*, chapter 12, 221–241. Kluwer Academic Publishers.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39: 127–177.

The PANDADealer System for Totally Ordered HTN Planning in the 2023 IPC

Conny Olz[1], **Daniel Höller**[2], **Pascal Bercher**[3]
[1] Ulm University, Ulm, Germany
[2] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[3] The Australian National University, Canberra, Australia
conny.olz@uni-ulm.de, hoeller@cs.uni-saarland.de, pascal.bercher@anu.edu.au

Abstract

The PANDADealer system is an HTN planning system for solving totally ordered HTN planning problems. It builds on the heuristic progression search of the PANDApro system, and extends it with a look-ahead technique to detect dead-ends and inevitable refinement choices. The technique is based on inferred preconditions and effects of tasks, or more precisely, their decomposition methods.

Introduction

The PANDADealer (Dead-End Analysis with Look-Aheads and Early Refinements) system is a progression search-based planner that has been enhanced with a look-ahead technique based on inferred preconditions and effects of decomposition methods. It is specifically designed to solve totally ordered HTN planning problems. The system is build upon the PANDApro system and uses its pure heuristic search-based configurations (Höller 2023b) and also those using a combined heuristic- and landmark-based search guidance (Höller 2023a).

Search-based systems in HTN planning can be divided into plan space-based systems and progression-based systems (see Bercher, Alford, and Höller, 2019). The latter only process the first task in the task ordering of the current task network. PANDADealer is builds on the systematic progression search introduced by Höller et al. (2020) and uses the graph search described by Höller and Behnke (2021), i.e., it maintains a black-list of already visited search nodes to process every node only a single time.

The system uses the common preprocessing stack of the PANDA framework: HDDL (Höller et al. 2020) as standard input language, followed by the grounding procedure introduced by Behnke et al. (2020).

The search is guided by using heuristics estimating the goal distance (or the remaining costs in case of optimal planning), some configurations additionally exploit landmarks for search guidance. Next we briefly describe the look-ahead technique, followed by the used heuristics and landmarks.

Look-Ahead Technique

The look-ahead technique employed in PANDADealer is based on inferred preconditions and effects of decomposition methods (Olz, Biundo, and Bercher 2021). These preconditions and effects are derived from the primitive tasks within the refinements of a method. Preconditions specify the facts that must hold in the state before executing the refinements, while effects indicate the changes in the state (additions or deletions) that occur after execution. Calculating the exact sets of preconditions and effects is computationally expensive; therefore, we only calculate a relaxed version in a preprocessing step, which disregards the executability of the refinements.

During the actual search, we treat the task network for each search node as a sequence of primitive tasks, where the compound tasks are enriched with their inferred preconditions and effects. Starting from the first task, we check the preconditions of its methods in relation to the current state. For the "applicable" methods, we add all possible positive effects and remove the guaranteed negative effects, resulting in a new state. The new state is then used to evaluate the preconditions of the methods associated with the second task, propagating their effects in a similar manner. This process continues until the end of the task network. If the preconditions of a primitive task are not satisfied or no method of a compound task is applicable in its respective state, the search node is pruned as it represents a dead-end. If this is not the case but if a compound task has only one applicable method, we immediately decompose that task to eliminate future branching points. Further be aware that this "early application" of methods might help getting better heuristic estimates, because heuristics might not be able to detect that there is only a single applicable method.

For a comprehensive and detailed explanation of the look-ahead technique we refer to the respective paper by Olz and Bercher (2023).

RC Heuristics

The family of relaxed composition (RC) heuristics (Höller et al. 2018, 2019, 2020) uses classical heuristics to estimate the goal distance during HTN search. This is done based on a relaxation of the HTN model to a classical model. This model is only used for heuristic calculation. It is created in a way that the set of solutions increases compared to the HTN model. HTN planning starts with the initial task(s) and decomposes them until only actions are left. This process can be seen as the building process of a tree. The RC model captures (a relaxation of) the building process of that tree in the state of the classical model, but in a bottom-up manner,

track	config	landmarks	search	heuristic
agile	agile-1	none	GBFS	rc(add)
	agile-lama	LM-Cut	GBFS	rc(add)
satisf.	agile-lama	LM-Cut	GBFS	rc(add)
optimal	optimal	none	A*	rc(lmc)

Table 1: Overview over the winning configurations.

compositing tasks.

The RC model is computed once in a preprocessing step and updated during search. It is linear in the size of the HTN model and can be combined with arbitrary classical planning heuristics. In the IPC, we combine it with the Add (Bonet and Geffner 2001), the FF (Hoffmann and Nebel 2001), and the LM-Cut (Helmert and Domshlak 2009) heuristic. Höller et al. (2018) have shown that the combination of the RC model with an admissible heuristic from classical planning results in an admissible HTN heuristic, so we use the latter (RC with LM-Cut) for optimal planning.

Landmarks

Similar to the LAMA system from classical planning (Richter and Westphal 2010), our configurations using landmarks combine heuristic-based and landmark-based guidance in a multi-fringe search, where one fringe is sorted by a heuristic, and one by an LM-count heuristic computed on the landmarks. The system extracts nodes from the fringes in turn and each successor node is inserted into both fringes with the respective heuristic estimate. We combine it with two approaches for landmark generation.

The first one computes LM-Cut heuristic on the RC model of the initial search node. The generated landmarks are stored and tracked during search.

The second one generates the landmarks using the approach of Höller and Bercher (2021). It extends the work from classical planning by Keyder, Richter, and Helmert (2010), who represent a delete-free classical planning problem as AND/OR graph, and extract landmarks from this graph afterwards. We extend the AND/OR graph to also represents parts of the decomposition hierarchy, and applies the unchanged extraction algorithm afterwards. We again generate the landmarks on the initial search node and track them afterwards during search.

Configurations

PANDADealer won all of the total-order HTN tracks of the IPC 2023. In Table 1 we give an overview over the details of the configurations.

References

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On Succinct Groundings of HTN Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9775–9784. AAAI Press.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 6267–6275. IJCAI organization.

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

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)*. AAAI Press.

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 and Domain 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 and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC)*.

Höller, D.; and Behnke, G. 2021. Loop Detection in the PANDA Planning System. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, 168–173. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9883–9891. AAAI Press.

Höller, D.; and Bercher, P. 2021. Landmark Generation in HTN Planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 11826–11834. AAAI Press.

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)*, 114–122. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6171–6175. IJCAI organization.

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

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, 335–340. IOS Press.

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)*, 65–73. AAAI Press.

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)*, 11903–11912. AAAI Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39: 127–177.

LTP: Lifted Tree Path

Gaspard Quenard, Damien Pellier, Humbert Fiorino
Univ. Grenoble Alpes - LIG
F-38000 Grenoble, France
gaspard.quenard humber.fiorino damien.pellier @univ-grenoble-alpes.fr

Abstract

In this paper, we present our planner named LTP which stand for Lifted Tree Path aimed at solving Totally Ordered Hierarchical Task Network (TOHTN) problems. Our planner is based on the Satisfiability (SAT) planning paradigm and builds upon the concepts of the Lilotane planner (Schreiber [2021]), which has scored 2nd in the last IPC in the HTN Total Order track.

Introduction

Satisfiability (SAT) planning is a widely-used planning paradigm that employs Boolean satisfiability solvers to find solutions for planning problems (Kautz et al. [1992, 2006]). SAT solvers are efficient tools for solving propositional logic problems. The main challenge in SAT planning lies in identifying and formulating the appropriate set of rules and constraints that effectively encode a given planning problem into SAT clauses. Once the planning problem is encoded into SAT clauses, the solution process relies on the underlying SAT solver to efficiently search for satisfying assignments.

Several SAT planners have been developed to encode TOHTN problems (Behnke et al. [2018], Schreiber et al. [2019], Schreiber [2021]). These planners utilize a structure referred to as a hierarchical tree to represent the problem hierarchy up to a certain depth. This hierarchical tree is subsequently used to encode the set of relevant SAT clauses.

The difference between previous approaches and LTP (Lifted Task Planning) is that the latter does not directly encode the entire hierarchy of the problem into propositional logic. Instead, it selectively extracts only the primitives from the hierarchical tree that may appear in valid plans and encodes them into propositional logic. It focuses solely on the actions of the plan rather than the full hierarchy. Therefore, LTP does not utilize boolean variables to encode tasks or methods during the SAT clause encoding process.

Hierarchical Tree

LTP utilizes the same hierarchical tree structure as the lilotane and TreeRex planners.

The hierarchical tree can be described as a sequence of hierarchical layers, where each layer is an array of positions, each containing a set of elements. These elements can be facts, reductions, or actions. The layers are computed incrementally, starting with an initial layer (L0) that includes the initial reduction. Subsequently, each layer is defined by including all operations that match a subtask of some operation from the previous layer.

Figure 1 illustrates an example of a hierarchical tree containing three layers for a problem in the Transport domain, as defined in the Lilotane paper. In this example, Lilotane encodes the entire decomposition tree into SAT clauses. However, LTP differs by keeping only the last layer of the decomposition tree. From this layer, it encodes only the actions that may be part of a solution plan as illustrated in the figure 2. The ordered constraints between these actions can be inferred from the hierarchical tree, and the method's preconditions can be encoded to the relevant actions in their first subtask.

Instantiation

The general planning procedure of LTP is similar to the other SAT planners for TOHTN problems:

1. Initialize the first layer (l0) of the hierarchial tree following the problem description.
2. Construct the next layer (l+1) of the hierarchical tree on the basis of the layer l.
3. Use the current hierarchical tree to encode the SAT clauses.
4. Launch the solver. If no solution is found, goto 2

References

Dominik Schreiber. Lilotane: A lifted sat-based approach to hierarchical planning. *Journal of artificial intelligence research*, 70:1117–1181, 2021.

Henry A Kautz, Bart Selman, et al. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363. Citeseer, 1992.

Henry Kautz, Bart Selman, and Joerg Hoffmann. Satplan: Planning as satisfiability. In *5th international planning competition*, volume 20, page 156, 2006.

Gregor Behnke, Daniel Höller, and Susanne Biundo. totsat-totally-ordered hierarchical planning through sat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

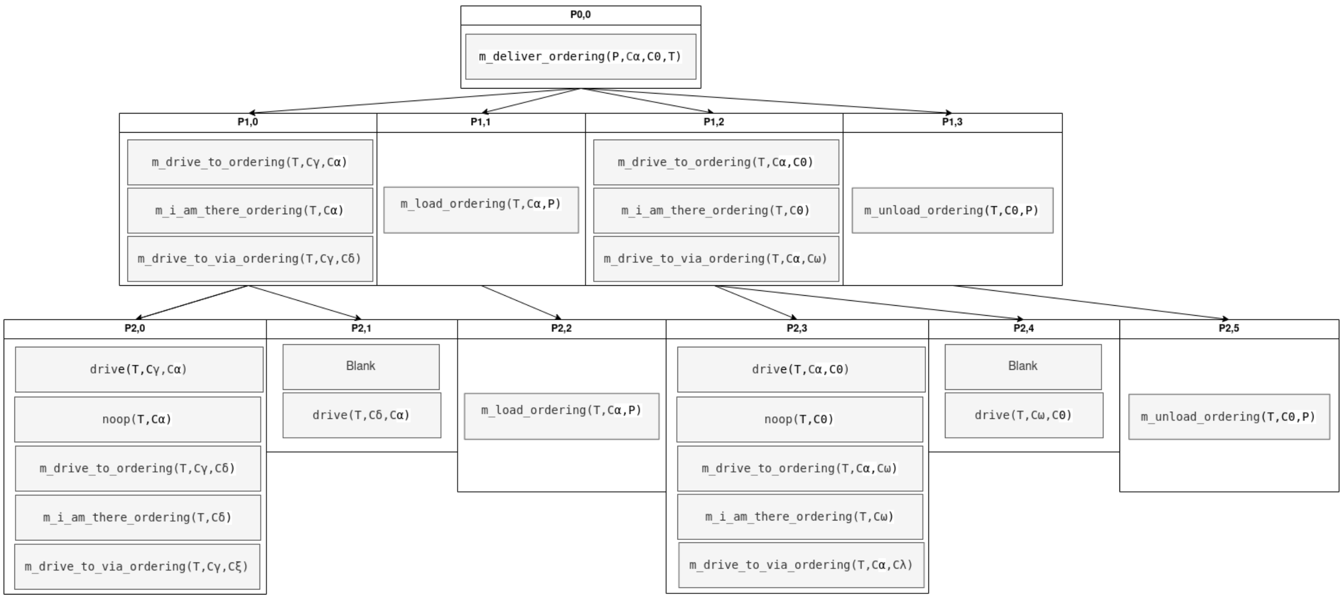

P0,0
m_deliver_ordering(P,Cα,C0,T)
P1,0
m_drive_to_ordering(T,Cγ,Cα)
m_i_am_there_ordering(T,Cα)
m_drive_to_via_ordering(T,Cγ,Cδ)
P1,1
m_load_ordering(T,Cα,P)
P1,2
m_drive_to_ordering(T,Cα,C0)
m_i_am_there_ordering(T,C0)
m_drive_to_via_ordering(T,Cα,Cω)
P1,3
m_unload_ordering(T,C0,P)
P2,0
drive(T,Cγ,Cα)
noop(T,Cα)
m_drive_to_ordering(T,Cγ,Cδ)
m_i_am_there_ordering(T,Cδ)
m_drive_to_via_ordering(T,Cγ,Cξ)
P2,1
Blank
drive(T,Cδ,Cα)
P2,2
m_load_ordering(T,Cα,P)
P2,3
drive(T,Cα,C0)
noop(T,C0)
m_drive_to_ordering(T,Cα,Cω)
m_i_am_there_ordering(T,Cω)
m_drive_to_via_ordering(T,Cα,Cλ)
P2,4
Blank
drive(T,Cω,C0)
P2,5
m_unload_ordering(T,C0,P)

Figure 1: Example a hierarchial tree containing 3 hierarchical layers for a problem of the domain Transport as defined in the Lilotane paper. The first subtask of the method m_deliver_ordering can be accomplish by the three methods reported in the position P1,0)

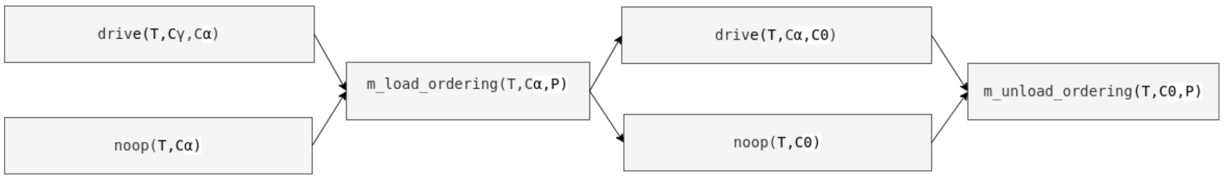


Figure 2: Space of reseach encoded by LTP into SAT clauses

Dominik Schreiber, Damien Pellier, Humbert Fiorino, et al. Tree-rex: Sat-based tree exploration for efficient and high-quality htn planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 382–390, 2019.

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

		Domains									
Planner	**Score**	**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)
PANDA$_{\text{pro}}$-λ (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)
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)
PANDA$_{\text{pro}}$-λ (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).

		Domains									
Planner	**Score**	**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)
PANDA$_{\text{pro}}$-λ (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)
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)
PANDA$_{\text{pro}}$-λ (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, PANDA$_{\text{pro}}$ (Höller 2023b), PANDA$_{\text{pro}}$-λ (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$_\pi$ 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*.

New HTN Domains in the 2023 IPC

Gregor Behnke[1], Jane Jean Kiam[2], Dominik Schreiber[3]
[1]University of Amsterdam, Netherlands, g.behnke@uva.nl
[2]Universität der Bundeswehr, Munich, Germany, jane.kiam@unibw.de
[3]Karlsruhe Institute of Technology, Germany, dominik.schreiber@kit.edu

Abstract

We present four new hierarchical planning domains named Colouring, Lamps, SharpSAT, and Ultralight-Cockpit.

Colouring

Coloring is a Partially Ordered domain authored by G. Behnke. It encodes a version of the tiling problem (van Emde Boas 2019), which is frequently used for complexity reductions. Given a set of available tiles, each having a color at one of its edges, the task is to fill an $n \times n$ square with these tiles, s.t., touching edges have the same color. The outer edge has no required color. This problem is NP-complete for unary encoded n. The encoding uses the idea of proof encoding double-exponentially time-bounded Turing Machine (Alford, Bercher, and Aha 2015). The colouring is determined by a sequence of actions generated by a totally-ordered decomposition, which only checks the touching colours in the left and right directions. The HTN's partial order is used to simulate a memory of arbitrary size that keeps track of the up-facing colours of each line to check whether the up/down touching tiles are the same colour.

Lamps

The Totally Ordered domain *Lamps*, authored by G. Behnke, models a variant of the game "Lights Out," which is about an $n \times m$ field with lamps that can either be on or off. Switching a lamp forces all horizontally and vertically connected lamps of the same status (on or off) to also toggle. This reachability-based procedure can be easily modeled with an HTN, but is hard to express using classical planning.

SharpSAT

The Totally Ordered domain *SharpSAT*, authored by D. Schreiber, models the problem #SAT, or *(exact) model counting*. This problem is to count the number of different variable assignments ("models") which satisfy a given propositional formula (Gomes, Sabharwal, and Selman 2021). The complexity class of #SAT, named #P, is (handwavingly) somewhere between NP and PSPACE and therefore not considered as hard as HTN planning. Nevertheless, a hierarchical planning model for #SAT is appealing due to the problem's natural hierarchical structure, its rather simple formulation, and a number of interesting search properties (see below).

We express instances of #SAT as TOHTN planning instances. Our hierarchical model is based on the straight forward recursive CDP algorithm (Birnbaum and Lozinskii 1999), which is a modified DPLL search procedure (Davis, Logemann, and Loveland 1962). When finding a model at decision depth d, the procedure does not terminate (as DPLL) but instead adds $2^{|V|-d}$ to its global model count and backtracks. The only decisions a planner can make is choosing a literal to branch on (or, if unit clauses are present, which one to satisfy first). The domain has no dead-ends (DPLL backtracking is performed with explicit subtasks); however, the branching choices a planner makes can result in substantial differences with respect to the effective search space size. As such, informative heuristics and/ or restarts with different decision-making have the potential to make a big difference. The hierarchy's depth is limited to $\mathcal{O}(|V| + |C|)$ levels until all variables have been assigned.

Arbitrarily difficult benchmarks can be generated from DIMACS CNF instances, e.g., benchmarks from the International SAT Competitions[1] or randomly generated difficult 3-SAT instances. A found plan can be transformed to an actual model count in linear time using an associated decoder script. This linear-time procedure just looks for specific actions `A_OUTPUT_EXPONENTIAL_COUNT d` and adds $2^{|V|-d}$ to a model counter for each such action.

Ultralight-Cockpit

Ultralight-Cockpit is a Partially Ordered domain authored by J. J. Kiam and P. Jamakatel (Kiam and Jamakatel 2023), and is motivated by its application on a Pilot Assistance System (PAS) for single-pilot ultralight aircraft. It models various tasks to be performed by a private pilot, while focusing on tasks necessary for handling emergency situations. Modeling the tasks in HTN is natural, as instructions documented in pilot operating handbooks (e.g. SHARK (2017); Pooley (2003)), are sequences of abstract or primitive tasks, without reference to reachable states.

The HTN model of the Ultralight-Cockpit domain is mainly used for two purposes: for generating instructions in

[1]https://satcompetition.github.io/2022/downloads.html

form of a task plan to be displayed on the cockpit as guidance for the private pilot in distress (Jamakatel and Kiam 2024), as well as for automated pilot observation to recognize the pilot's goal task (Jamakatel et al. 2023) using the Plan and Goal Recognition (PGR) technique for HTN planning problems developed by Höller et al. (2018). With an automated goal recognition, the PAS can intervene or guide the pilot without asking continuously for the pilot's intent during an emergency. By doing so, the PAS avoids increasing the pilot's mental workload, as private pilots in general do not undergo intensive training or strict health screening to ensure their capabilities for coping with emergencies.

Unexpected precautionary landing may be necessary when anomalies arise. In this use case, to land an ultralight aircraft safely, the emergency landing site is still required to fulfill certain conditions such as being free of obstacles and within reach of the aircraft. Besides, the surface of the site as well as its slope gradient must be reasonable for landing. For the sake of benchmarking HTN-planners, instances of arbitrary numbers of landing sites with randomly set conditions (i.e. distance to aircraft, presence of obstacles, condition of the surface and the slope) can be generated.

References

Alford, R.; Bercher, P.; and Aha, D. 2015. Tight bounds for HTN planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 25, 7–15.

Birnbaum, E.; and Lozinskii, E. L. 1999. The good old Davis-Putnam procedure helps counting models. *Journal of Artificial Intelligence Research*, 10: 457–477.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem-proving. *Communications of the ACM*, 5(7): 394–397.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2021. Model counting. In *Handbook of satisfiability*, 993–1014. IOS press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2018. Plan and Goal Recognition as HTN Planning. In *Proceedings of the 30th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 466–473. Volos, Greece: IEEE Computer Society.

Jamakatel, P.; Bercher, P.; Schulte, A.; and Kiam, J. J. 2023. Towards Intelligent Companion Systems in General Aviation using Hierarchical Plan and Goal Recognition. In *Proceedings of the 11th International Conference on Human-Agent Interaction (HAI 2023)*. Association for Computing Machinery.

Jamakatel, P.; and Kiam, J. J. 2024. Generation and Communication of Strategic Plans at Different Levels of Abstraction for Intelligent Assistance Systems in General Aviation. In *34th Congress of the International Council of the Aeronautical Sciences (ICAS)*.

Kiam, J. J.; and Jamakatel, P. 2023. Can HTN Planning Make Flying Alone Safer? In *Proceedings of the 6th ICAPS Workshop on Hierarchical Planning (HPlan 2023)*, 44–48.

Pooley, D. 2003. *POOLEYS Private Pilots Manual: JAR Flying Training, Volume 1*. Cranfield, UK: POOLEYS.

SHARK. 2017. *Flight Manual: UL airplane*. SHARK.AERO CZ s.r.o.

van Emde Boas, P. 2019. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, 331–363. CRC Press.