KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-020-00699-y

SYSTEMS DESCRIPTION

The PANDA Framework for Hierarchical Planning

Daniel Höller[1] · Gregor Behnke[1] · Pascal Bercher[1] · Susanne Biundo[1]

Received: 15 February 2020 / Accepted: 11 December 2020

Abstract
During the last years, much progress has been made in hierarchical planning towards domain-independent systems that come with sophisticated techniques to solve planning problems instead of relying on advice in the input model. Several of these novel methods have been integrated into the *PANDA framework*, which is a software system to reason about hierarchical planning tasks. Besides solvers for planning problems based on plan space search, progression search, and translation to propositional logic, it also includes techniques for related problems like plan repair, plan and goal recognition, or plan verification. These various techniques share a common infrastructure, like e.g. a standard input language or components for grounding and reachability analysis. This article gives an overview over the PANDA framework, introduces the basic techniques from a high level perspective, and surveys the literature describing the diverse components in detail.

1 Introduction

Planning is the task of finding a course of action that fulfills the goal(s) of an agent. The task is usually solved based on a model that describes an environment and how to change it. *Hierarchical* planning [14, 23] comes with an additional hierarchy on the tasks that describes how an abstract task can be divided into more concrete ones. During the last years, much progress has been made, not only regarding solving techniques [1, 2, 8–11, 15, 19, 33, 34, 36, 42, 44], but also preprocessing (e.g. grounding and reachability analysis [12, 39]), and related tasks like plan and goal recognition or plan verification [3, 7, 29].

Much of this recent work has been done based on the PANDA framework[1]. PANDA stands for *Planning and Acting in a Network Decomposition Architecture*. In its very beginning, it was a single planning system based on plan space search, combining Partial Order Causal Link techniques with Hierarchical Task Network (HTN) planning to an approach called hybrid planning [20, 40]. Search guidance was based on syntactical comparison of different parts of the search space [40, 41]. Over time, solving techniques have been advanced and additional functionalities have been integrated. Now it combines work on diverse topics related to hierarchical planning and has recently been used to integrate planning capabilities into several other systems [4, 13, 16].

While having such a framework of reasoners sharing the same infrastructure like a common input language simplifies the development and comparison of new techniques, there has also been an issue we want to resolve with this article: by now, the PANDA framework consists (among others) of three completely different planning approaches [8, 18, 33] – yet all of them are referred to as PANDA since they are part of this framework; this led to confusion.

In this article, we give a detailed outline of the planning approaches and other functionalities that are part of the PANDA framework. Due to limited space, we focus on recent work. We give a high-level description of each functionality and the respective solving technique(s) realized in PANDA and an elaborated survey of the corresponding literature, including pointers to discussions of related work and benchmark results.

✉ Daniel Höller
daniel.hoeller@alumni.uni-ulm.de

Gregor Behnke
gregor.behnke@alumni.uni-ulm.de

Pascal Bercher
pascal.bercher@alumni.uni-ulm.de

Susanne Biundo
susanne.biundo@uni-ulm.de

[1] Institute of Artificial Intelligence, Ulm University, 89069 Ulm, Germany

[1] The source code of the PANDA framework and benchmark sets are available at *panda.hierarchical-task.net*.

Published online: 07 January 2021

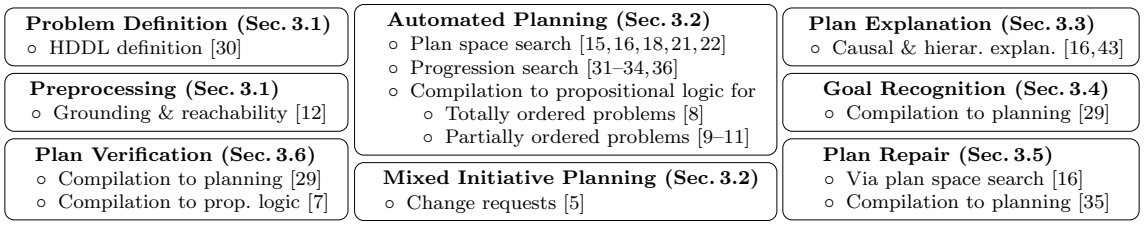

Fig. 1 Overview of the functionalities integrated in the PANDA framework. Each box represents a functionality. For each of them, a list of the different approaches and the corresponding literature is given

2 Hierarchical Planning

We start with an informal introduction of the problem classes supported by PANDA. In hierarchical planning, there are two types of tasks: so-called *primitive* tasks are equivalent to actions in classical planning, they hold preconditions and effects, are directly executable in the environment, and cause propositional state features to change, i.e., they implicitly define a state transition system similar to a deterministic finite automaton. *Abstract* tasks represent a more abstract thing to do, like *"deliver package P at position A"*. They cannot be executed in a single step and need to be *decomposed*.

The model includes a set of rules on how to do this, called *(decomposition) methods*. A method is a pair consisting of an abstract task specifying which task may be decomposed by the method and a partially ordered (multi-) set of *subtasks* (that may be primitive or abstract). When the method is applied, the abstract task that is decomposed is removed and replaced by the subtasks. When the decomposed task was ordered with respect to other tasks, the inserted subtasks inherit its ordering relations. Decomposition is similar to rule application in formal grammars. However, the "left-hand side" of the rules only contain a single abstract task, and the "right-hand side" is partially ordered. Such a grammar is able to represent a (non-context-free) subset of the context-sensitive languages [27, 28].

The set of tasks to do is maintained in *task networks* that form a partially ordered (multi-)set of tasks. The planning process starts with one or more *initial tasks* that are usually abstract. They are decomposed until all remaining tasks are primitive.

To be a solution, such a task network containing only primitive tasks must be executable, i.e., there must be a linearization of all contained tasks (in line with the ordering constraints imposed by the decompositions) that can be applied in the initial state of the system.

This most basic formalism is called *Hierarchical Task Network* (HTN) planning [23], but there is a large set of slightly different formalisms. A summary has been given by Bercher et al. [14]. It has been shown that HTN planning can describe undecidable problems like the intersection problem of context-free grammars [23].

HTN planning forms the core of the PANDA system. However, parts of PANDA can also deal with problems where abstract tasks hold preconditions and effects that impose a certain structure in methods decomposing it (see Bercher et al. [17] for an overview). Another variant supported is HTN planning with task insertion (TIHTN). Here, the planning system needs to decompose the initial tasks, but is allowed to insert actions apart from the hierarchy. This makes the problem decidable [24], but less expressive [27].

3 The PANDA Framework

While the first versions of PANDA have been developed in JAVA, the recent work has been done in C++. PANDA is available under GPLv3 license.

An overview of its functionality is given in Fig. 1. Each box represents one functionality. For some of them, more than one approach has been realized. Different approaches are given inside the respective box, each with the literature describing it. The following sections describe each functionality and approach.

3.1 A Common Infrastructure

At the left of Fig. 1, two basic functionalities are given that are used by many other components: The standard input language HDDL [30] and our grounding procedure that integrates a reachability analysis [12].

3.1.1 The Problem Definition Language HDDL

A common language for problem definition is very important for a field of research. For system developers it enables the simple comparison of their solvers. It might lead to a common set of benchmark problems to judge the performance and the progress in the field. For users wanting to integrate

planners into their systems, it means that they do not have to commit to a planning system before modeling their specific problem. They can describe it in the standard language and try which planning system fits their needs afterwards.

In classical planning there is such a language, called *Planning Domain Definition Language* (PDDL) [25]. We developed the *Hierarchical Domain Definition Language* (HDDL), an extension to PDDL to support the definition of hierarchical planning problems, and proposed it as common input language [30]. It is the official input language for the International Planning Competition (IPC) 2020. We tried to stay as close as possible to the PDDL standard and created it as syntactical extension to it. We hope that this helps to share techniques and tools not only in the HTN community but also between classical and hierarchical planning.

3.1.2 Grounding and Reachability Analysis

The models used to define planning problems usually support variables to make them more compact, but many planning systems (also most solvers in the PANDA framework) need grounded (i.e. variable-free) models as input. During the grounding process, variables in the definition are replaced by all possible constants. An optimal grounding would include exactly those instantiations that may occur in at least one solution. However, it is infeasible to compute this set. Instead, an over-approximation is calculated. To enable an efficient planning process, the set should be as small as possible.

There has been little work on grounding hierarchical problems [39]. In classical planning there are techniques for state-based reachability analysis that can be reused in the hierarchical setting. Hierarchical reachability can also be used to make the grounding smaller. Only those elements reachable via state *and* hierarchy need to be included into the grounded model.

Besides the final size of the grounded model, a serious problem in practice is its intermediate size during the grounding process. We developed an approach that combines a fast generation with a small size of the final grounding [12]. The paper also contains benchmark results against the only other system from the literature.

3.2 Planning with PANDA

While PANDA comes with a limited support to plan lifted, all heuristics and both progression search and the translation to propositional logic only support ground planning. This means that a model will pass the grounding process (Sect. 3.1.2) before starting planning.

For a discussion of related work on search-based HTN planning, we refer to Bercher et al. [15] and Höller et al. [36], for one on SAT-based approaches to Behnke et al. [10]. The given papers also contain benchmark results including systems from related work.

3.2.1 Heuristic Search in Plan Space

In HTN planning, there are two standard search techniques, plan space search and progression search, which is similar to state space search in classical planning. The basic characteristic of plan space search is that the system maintains a partial ordering between the tasks in search nodes. That way such systems benefit from a very compact representation of the search space (because a system that cannot deal with partial ordering has to represent all linearizations instead), but suffers from poor information about the "current state" during search (because the ordering of actions in a plan is not yet fixed). The latter might be a problem when computing (or designing) heuristics. The search algorithm used by the PANDA plan space search is described by Bercher et al. [16, 18].

PANDA comes with several heuristics to guide plan space search. Early ones were based on landmarks [21, 22]—tasks that need to be in *every* plan. Landmarks were calculated on a finite structure representing the space of possible decompositions of the abstract tasks in the current task network, called the Task Decomposition Graph (TDG). Recent heuristics [15] are still based on the TDG, but instead of calculating landmarks, they estimate the costs to reach a plan. The decomposition structure underlying an HTN model can be seen as an AND/OR tree [26, Chapter 11]. Abstract tasks form OR nodes since there are several options (methods) on how to decompose them. Methods form AND nodes, because all subtasks need to be processed. To calculate the heuristics, we assign actions their costs, methods the sum of the costs of their subtasks, and abstract tasks the minimum of all applicable methods. Costs need to be updated until they converged (which can be done efficiently). The basic process is optimized by integrating state-based reachability information [15].

Two heuristics have been introduced based on this approach [15]. The TDG_m heuristic estimates goal distance in search space and is thus well-suited to guide the search. The TDG_c heuristic estimates costs of actions that need to be added to the plan to transform it to a solution. It is admissible and can thus be used to find plans with minimal action costs.

3.2.2 Heuristic Progression Search

In progression search, the planner builds the plan in a forward manner only processing tasks that have no predecessor in the ordering relations of the current task network. It thereby commits to a totally ordered prefix of the plan. That way, the current state is fully specified during search and

may be used to calculate heuristics. However, systems from the literature rely on advice in the model instead.

We introduced a *heuristic* progression search system based on a method to use arbitrary heuristics from classical planning to guide the search [33, 34, 36]. To be able to use classical heuristics, we model the process of a bottom-up creation of the AND/OR tree (described in the last section) into a classical planning problem. This model is called the *Relaxed Composition* model (because methods in the encoding are applied bottom-up, *composing* tasks) and the family of resulting heuristics *RC heuristics*. A classical heuristic applied to the RC model estimates the number of action and method applications necessary to transform the current search node into a plan. Our approach results in informed heuristics for HTN planning. It can be used to create heuristics with interesting theoretical properties, e.g., admissible heuristics [11].

We recently described two new heuristic approaches. In the family of heuristics presented in Höller et al. [32], we first relax the delete effects in all actions and the ordering relations in all methods and then solve the resulting problem using integer programming.

A novel method for landmark (LM) extraction in HTN planning is presented in Höller and Bercher [31]. We use the resulting LMs in a LM count heuristic.

3.2.3 Translation to Propositional Logic

A major challenge when creating heuristics in HTN planning is that they need to be informed about the hierarchy *and* the state transition system induced by the set of actions. Translation-based approaches overcome this problem. When a problem is e.g. translated to propositional logic, the solver has a single logic model representing both. We integrated a translation encoding HTN planning problems into propositional logic. To translate the (undecidable) HTN problem into a decidable formalism, we bound the depth of decomposition in possible solutions. When the generated propositional formula is satisfiable, a plan has been found and can be returned. If not, the bound is increased. Due to undecidability of HTN planning, it is in general not possible to stop this process without getting an incomplete overall procedure.

We have introduced several translations. The first one has been specialized to problems where the subtasks of every method and the initial task network are totally ordered [8]. Other encodings can deal with arbitrary problems [9, 10]. Recently we introduced an encoding to find plans with a minimal number of actions [11].

3.2.4 Change Requests

The plans generated by PANDA are frequently used as the basis for providing assistance to a human user [4, 13, 16]. To this end, the plan is presented step-by-step and the user is instructed to perform the actions. Despite the plan being correct and goal-leading, a user might not be content with the presented instructions and may want to request changes to it. As such, PANDA's SAT-based planner contains a component that is able to handle these requests. We assume that the requests are transformed into formulae in Linear Temporal Logic (LTL [37]), which can be done by using pattern-based techniques. We then transform the formulae into clauses that can be added to any of PANDA's propositional encodings of HTN planning [5].

3.3 Plan Explanation

When integrating planning into other systems, there are several tasks to solve that are related to plan generation. Several of them can be solved with PANDA.

When autonomous agents have contact to humans or even work cooperatively with them, they need to be able to explain their behavior (i.e., the plans). There are different kinds of explanations. The PANDA framework includes techniques to explain why a certain action has been included into the plan [16, 43]. Explanations are thereby either based on the causal structure of the plan, i.e., on the relationship of the preconditions and effects of actions, or on the decomposition structure, i.e., on the abstract task the action in question is part of.

3.4 Plan and Goal Recognition

Plan and Goal Recognition (PGR) is the task of inferring the goal(s) and plan of an observed agent. The task is solved based on a model describing the agent's behavior and a sequence of actions executed by it. There are several approaches using classical planning models to represent the agent's behavior and compile the PGR problem into a series of classical planning problems [38].

PANDA includes a translation-based approach that supports full HTN planning to describe the behavior that shall be recognized [29]. This allows to recognize much more complex behavior than using classical planning [27, 28]. Like in classical planning, our approach compiles a PGR problem into a standard planning problem (an HTN problem in our case). This is done on the lifted model. The resulting problem can be solved by any standard HTN planning system. A discussion of related work is given by Höller et al. [29, Sec. 2].

3.5 Plan Repair

When the execution of a plan fails, there are two options: the planning process may be started again from the new system state (re-planning), or the old plan may be modified to deal with the failure (plan repair).

In HTN planning, re-planning is not an option because the hierarchy induces constraints on plans not represented in the state space of the system. Think about an agent that needs to go to some target position and has to take the same path back to its initial position. When the road system contains loops, this cannot easily be modeled using state because state is (usually) finite. It is, however, no problem in hierarchical planning. When the execution of such a plan is just aborted and it is re-planned when the agent fails to do something at the target position, the information about the way back is lost. Therefore it is necessary to develop plan repair mechanisms for hierarchical planning.

We developed an approach [35] based on our PGR encoding that compiles the original model together with the broken plan into a new planning problem. This is done on the lifted model and the generated problem is a common planning problem. It can thus be solved using arbitrary planning systems. An exhaustive discussion of related work can be found in Höller et al. [35].

3.6 Plan Verification

For the IPC 2020 we developed a plan verifier that checks whether a solution is correct based on the action sequence and all decomposition steps leading to it. Having the decomposition steps makes the task computationally feasible and allows to return a meaningful explanation of errors. Thus it is a useful feature to check whether planning systems work correctly.

However, planners usually do not return decomposition steps. Their reconstruction is NP-complete [6]. There are situations where this is necessary, e.g. when any form of post-processing is applied, like post-optimization or a very simple form of change requests. We developed two solvers for the task: One based on the PGR encoding [29] enforcing the whole plan as prefix and one based on a translation to propositional logic [7].

4 Conclusion

The PANDA framework combines a large set of functionalities related to hierarchical planning. For users, this enables solving various problems using the same software system; developers can re-use internal components. However, it makes the system hard to overlook. In this article we give an overview of the framework, introduce each functionality and approach realized for it, and give pointers to the corresponding literature.

Acknowledgements Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projekt-Nr. 232722074 – SFB 1102/ Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Alford R, Behnke G, Höller D, Bercher P, Biundo S, 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), pp 20–28. AAAI Press
2. Alford R, Kuter U, Nau DS (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), pp 1629–1634
3. Barták R, Maillard A, Cardoso RC (2018) Validation of hierarchical plans via parsing of attribute grammars. In: Proceedings of the 28th international conference on automated planning and scheduling (ICAPS). AAAI Press, pp 11–19
4. Behnke G, Bercher P, Kraus M, Schiller M, Mickeleit K, Häge T, Dorna M, Dambier M, Minker W, Glimm B, Biundo S (2020) New developments for Robert – Assisting novice users even better in DIY projects. In: Proceedings of the 30th international conference on automated planning and scheduling (ICAPS). AAAI Press
5. Behnke G, Biundo S (2018) X and more parallelism. Integrating LTL-next into SAT-based planning with trajectory constraints while allowing for even more parallelism. Intel Artif Rev Iberoam de Intel Artif 21(62):75–90
6. Behnke G, Höller D, Biundo S (2015) On the complexity of HTN plan verification and its implications for plan recognition. In: Proceedings of the 25th international conference on automated planning and scheduling (ICAPS), pp 25–33. AAAI Press
7. Behnke G, Höller D, Biundo S (2017) This is a solution! (...but is it though?) – Verifying solutions of hierarchical planning problems. In: Proceedings of the 27th international conference on automated planning and scheduling (ICAPS), pp 20–28. AAAI Press
8. Behnke G, Höller D, Biundo S (2018) totSAT – totally-ordered hierarchical planning through SAT. In: Proceedings of the 32nd AAAI conference on artificial intelligence (AAAI), pp 6110–6118. AAAI Press
9. Behnke G, Höller D, Biundo S (2018) Tracking branches in trees – A propositional encoding for solving partially-ordered HTN planning problems. In: Proceedings of the 30th IEEE international conference on tools with artificial intelligence (ICTAI), pp 73–80. IEEE Computer Society

10. Behnke G, Höller D, Biundo S (2019) 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), pp 7520–7529. AAAI Press
11. Behnke G, Höller D, Biundo S (2019) Finding optimal solutions in HTN planning – A SAT-based approach. In: Proceedings of the 28th international joint conference on artificial intelligence (IJCAI), pp 5500–5508. IJCAI
12. Behnke G, Höller D, Schmid A, Bercher P, Biundo S (2020) On succinct groundings of HTN planning problems. In: Proceedings of the 34th AAAI conference on artificial intelligence (AAAI). AAAI Press, pp 9775–9784
13. Behnke G, Schiller M, Kraus M, Bercher P, Schmautz M, Dorna M, Dambier M, Minker W, Glimm B, Biundo S (2019) Alice in DIY-wonderland or: Instructing novice users on how to use tools in DIY projects. AI Commun 32(1):31–57
14. Bercher P, Alford R, Höller D (2019) A survey on hierarchical planning – One abstract idea, many concrete realizations. In: Proceedings of the 28th internatioanl joint conference on artificial intelligence (IJCAI), pp 6267–6275. IJCAI
15. Bercher P, Behnke G, Höller D, Biundo S (2017) An admissible HTN planning heuristic. In: Proceedings of the 26th international joint conference on artificial intelligence (IJCAI), pp 480–488. IJCAI
16. Bercher P, Biundo S, Geier T, Hoernle T, Nothdurft F, Richter F, Schattenberg B (2014) Plan, repair, execute, explain – How planning helps to assemble your home theater. In: Proceedings of the 24th international conference on automated planning and scheduling (ICAPS), pp 386–394. AAAI
17. Bercher P, Höller D, Behnke G, Biundo S (2016) More than a name? On implications of preconditions and effects of compound HTN planning tasks. In: Proceedings of the 22nd european conference on artificial intelligence (ECAI). IOS Press, pp 225–233
18. Bercher P, Keen S, Biundo S (2014) Hybrid planning heuristics based on task decomposition graphs. In: Proceedings of the 7th annual symposium on combinatorial search (SOCS), pp 35–43. AAAI Press
19. Bit-Monnot A, Smith DE, Do M (2016) Delete-free reachability analysis for temporal and hierarchical planning. In: Proceedings of the 22nd european conference on artificial intelligence (ECAI), pp 1698–1699. IOS Press
20. Biundo S, Schattenberg B (2001) From abstract crisis to concrete relief: A preliminary report on combining state abstraction and HTN planning. In: Proceedings of the 6th european conference on planning (ECP), pp 157–168. AAAI Press
21. Elkawkagy M, Bercher, P, Schattenberg B, Biundo S (2012) Improving hierarchical planning performance by the use of landmarks. In: Proceedings of the 26th AAAI conference on artificial intelligence (AAAI), pp 1763–1769. AAAI Press
22. Elkawkagy M, Schattenberg B, Biundo S (2010) Landmarks in hierarchical planning. In: Proceedings of the 19th european conference on artificial intelligence (ECAI), pp 229–234. IOS Press
23. Erol K, Hendler J, Nau DS (1994) HTN planning: Complexity and expressivity. In: Proceedings of the 12th national conference on artificial intelligence (AAAI), vol 94. AAAI Press, pp 1123–1128
24. Geier T, Bercher P (2011) On the decidability of HTN planning with task insertion. In: Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI), pp 1955–1961. IJCAI/AAAI
25. Ghallab M, Howe A, Knoblock C, McDermott D, Ram A, Veloso M, Weld D, Wilkins D (1998) PDDL – The planning domain definition language (version 1.2). Technical report CVC TR-98-003/DCS TR-1165, Yale center for computational vision and control
26. Ghallab M, Nau DS, Traverso P (2004) Automated planning - theory and practice. Elsevier, Amsterdam
27. Höller D, Behnke G, Bercher P, Biundo S (2014) Language classification of hierarchical planning problems. In: Proceedings of the 21st european conference on artificial intelligence (ECAI), pp 447–452. IOS Press
28. Höller D, Behnke G, Bercher P, 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), pp 158–165. AAAI Press
29. Höller D, Behnke G, Bercher P, Biundo S (2018) Plan and goal recognition as HTN planning. In: Proceedings of the 30th IEEE international conference on tools with artificial intelligence (ICTAI), pp 466–473. IEEE Computer Society
30. Höller D, Behnke G, Bercher P, Biundo S, Fiorino H, Pellier D, Alford R (2020) HDDL: An extension to PDDL for expressing hierarchical planning problems. In: Proceedings of the 34th AAAI conference on artificial intelligence (AAAI), pp 9883–9891. AAAI Press
31. Höller D, Bercher P (2020) Landmark extraction in HTN planning. In: Proceedings of the 3rd ICAPS workshop on hierarchical planning (HPlan)
32. Höller D, Bercher P, Behnke G (2020) Delete- and ordering-relaxation heuristics for HTN planning. In: Proceedings of the 29th international joint conference on artificial intelligence (IJCAI), pp 4076–4083. IJCAI
33. Höller D, Bercher P, Behnke G, Biundo S (2018) A generic method to guide HTN progression search with classical heuristics. In: Proceedings of the 28th internatioanl conference on automated planning and scheduling (ICAPS), pp 114–122. AAAI Press
34. Höller D, Bercher P, Behnke G, 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), pp 6171–6175. IJCAI
35. Höller D, Bercher P, Behnke G, Biundo S (2020) HTN plan repair via model transformation. In: Proceedings of the 43rd german conference on AI (KI), pp 88–101. Springer
36. Höller D, Bercher P, Behnke G, Biundo S (2020) HTN planning as heuristic progression search. J Artif Intel Res 67:835–880
37. Pnueli A (1977) The temporal logic of programs. In: Proceedings of the 18th annual symposium on foundations of computer science (SFCS), pp 46–57. IEEE
38. Ramírez M, Geffner H (2009) Plan recognition as planning. In: Proceedings of the 21st international joint conference on artificial intelligence (IJCAI), pp 1778–1783. AAAI Press
39. Ramoul A, Pellier D, Fiorino H, Pesty S (2017) Grounding of HTN planning domain. Int J Artif Intel Tools 26(5):1–24
40. Schattenberg B (2009) Hybrid planning & scheduling. Ph.D. thesis, Ulm University, Germany
41. Schattenberg B, Bidot J, Biundo S (2007) On the construction and evaluation of flexible plan-refinement strategies. In: Proceedings of the 30th annual german conference on AI (KI), pp 367–381. Springer
42. Schreiber D, Pellier D, Fiorino H, Balyo T (2019) Tree-REX: SAT-based tree exploration for efficient and high-quality HTN planning. In: Proceedings of the 29th international conference on automated planning and scheduling (ICAPS), pp 382–390. AAAI Press
43. Seegebarth B, Müller F, Schattenberg B, Biundo, S (2012) Making hybrid plans more clear to human users – A formal approach for generating sound explanations. In: Proceedings of the 22nd international conference on automated planning and scheduling (ICAPS), pp 225–233. AAAI
44. Shivashankar V, Alford R, Aha DW (2017) Incorporating domain-independent planning heuristics in hierarchical planning. In: Proceedings of the 31st AAAI conference on artificial intelligence (AAAI), pp 3658–3664. AAAI Press