Heuristics in HTN Planning

Daniela Kuinchtner, Felipe Meneguzzi
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
daniela.kuinchtner@edu.pucrs.br, felipe.meneguzzi@pucrs.br

Abstract

Within the field of Automated Planning, there are several approaches to solve planning tasks, two of the most popular are *classical planning* and *hierarchical planning*. Hierarchies are common structures used to understand and conceptualize the world. Hierarchical Task Network (HTN) planning is the branch that represents and handles hierarchies. Heuristic search can be the basis for the existing solvers in classical and hierarchical planning. However, heuristics in classical planning, such as landmarks, delete-relaxation, and linear programming-based ones, are more sophisticated than hierarchical ones due to the classical planning research so far. In this paper, we measure the efficiency of classical heuristics applied to hierarchical planning using PANDA, a progression-search-based solver.

Introduction

Planning is the task of finding a course of action that achieves the goal(s) of an agent when we execute it. In its simplest form, the model underlying the task contains a description of the relevant parts of the environment the agent is planning for and a set of actions that describe how the agent can change it. Usually, a (finite) set of propositional variables describes the environment; the definition of an action contains a description of preconditions that need to be fulfilled to apply it and a set of effects (changes to the system it causes). The objective is to find a sequence of actions transforming the system into a state where certain state features hold. This process is often done using *heuristic search* (Höller et al. 2019).

Over the past two decades, heuristic search has established itself as the chosen method for optimal deterministic planning. This progress is due to the intense focus on developing domain-independent admissible heuristics, such as delete-relaxation, critical path, landmark, and operator-counting ones (Trevizan, Thiébaux, and Haslum 2018).

Hierarchical Task Network (HTN) planning is an AI planning technique that breaks with the tradition of classical planning (Nau, Ghallab, and Traverso 2004). The basic idea behind this technique includes an initial state description, a task network as an objective to be achieved, and domain knowledge consisting of networks of primitive and abstract tasks. A task network represents a hierarchy of tasks that can execute if the task is a primitive one, or if the task is an abstract one, it is decomposed into refined subtasks. The planning process starts by decomposing the initial task network and continues until it decomposes all abstract tasks; that is, we find a solution. The solution is a plan that equates to a set of primitive tasks applicable to the initial world state (Georgievski and Aiello 2015).

There are mainly two strategies to solve HTN planning problems. The first is i) a decomposition-based search in a plan-space, which searches a space of partial plans, and the second is ii) a progression-based search in state-space. The latter builds solutions in a forward manner, enabling a heuristic to base its calculation on a fully specified and continuously updated state. So far, plan-space search is the basis of most HTN planners that guide the search using heuristic functions. Those systems represent the set of search nodes more effectively by maintaining a partial ordering between tasks, but they have only limited information about the current state during the search. Unlike plan-space search, progression search systems can calculate their heuristics incorporating the current state because the system tracks it during the search. Some existing progression-based systems do not rely on heuristics (e.g., SHOP2). However, current heuristic planners for standard HTN models (e.g., PANDA) introduced progression-based search relying its computation on heuristics (Höller et al. 2019).

PANDA (Höller et al. 2021) is a solver that uses progression search as a search-based algorithm. It is a framework that combines different techniques related to HTN planning. Some of the existing heuristics implemented on PANDA are: delete- and ordering-relaxation, landmark counting, additive, maximization, fast-forward, and linear programming heuristics. In order to measure the performance of HTN solvers that use progression search and rely on heuristics, we consider using the heuristic function information of the progression-search-based PANDA solver.

Approach

Planning and Acting in a Network Decomposition Architecture (PANDA) (Höller et al. 2021) is a solver that uses progression search as a search-based algorithm and Hierarchical Domain Definition Language (HDDL) (Höller et al. 2020) as input language. We measure the efficiency of the additive, fast-forward, and landmark heuristics in HTN planning on PANDA planner.

We test the PANDA heuristics implementation with a Depth-First Search (DFS) implementation based on the SHOP2 (Simple Hierarchical Ordered Planner 2) (Nau et al. 2001), a progression-based search and domain-independent planning system for HTN planning. In its implementation, SHOP2 uses DFS but does not use the information of heuristic functions to select the task method of the fringe to expand. So, in order to improve the selection method process, we implemented a DFS algorithm that uses a stack fringe with move-ordering by following the heuristic function information to select the task method.

We explore the PANDA solver's efficiency with the use of heuristics. First, we test PANDA with existing domains to reproduce the results of Holler et al. paper (Höller et al. 2019). Then, we test these domains with an improved implementation of SHOP2's DFS algorithm on PANDA by adding the heuristic information as a method to explore the fringe.

To be able to use PANDA, first, we need to parse the HDDL information to the planner. Second, as the framework only supports ground planning for the progression search, the model passes the grounding process before planning. Then, using progression search, the planner builds the plan in a forward manner. We use the Relaxed Composition model (Höller et al. 2019), an approach to use classical heuristics in HTN by modeling the process of AND/OR trees into a classical planning problem. 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. That approach results in informed heuristics for HTN planning.

Implementation

On PANDA, the search is based on a fully grounded model; that is, it is based on the preprocessing, i.e., grounding and reachability analysis. The PANDA framework is written in C++, and all scripts we wrote to extract data from PANDA results were written in Python. We included the h^{add} (Bonet and Geffner 2001), h^{FF} (Hoffmann and Nebel 2011), and h^{LM-cut} (Helmert and Domshlak 2009) as classical heuristics. We test each heuristic with four search algorithms:

- Weighted A*;
- A*;
- Greedy Best-First; and
- Depth-First with move-ordering (DFS*).

The latter is our novel implementation. Technically, each strategy is determined by the behavior of its fringe. For example, the Depth-First with move-ordering has a stack fringe, and it expands task methods by relying on the heuristic functions computation. Also, we test a simple Depth-First search without relying on these heuristics.

Our DFS* algorithm is a simple DFS (already developed for SHOP2) with a fringe stack ordered by heuristic function information, instead of just randomly extracting methods of the fringe, as common DFS algorithms do.

Experiments

We include the following HTN planners in our evaluation. In the following, we use the abbreviations GBFS, A*, WA*, and DFS* for Greedy Best-First search, A* search, Weighted A* search (with a weight of 2), and Depth-First Search with move-ordering, respectively. We include 12 heuristic search configurations of our system: $\{h^{LM-cut}, h^{Add}, h^{FF}\} \times \{$WA*, A*, GBFS, DFS*$\}$. We further include a DFS with no move ordering and no heuristic comparison. We run experiments on an Intel(R) Xeon(R) CPU E5-2620 v3 CPUs with @ 2.40GHz base frequency, 32 GB RAM, and 20 seconds of timeout. We use the following domains in our evaluation:

- The UM-TRANSLOG, SATELLITE, and WOODWORKING domains are HTN versions of the corresponding domains known from classical planning (Bercher, Keen, and Biundo-Stephan 2014);
- SMARTPHONE – A domain describing the task of operating a smartphone (Bercher, Keen, and Biundo-Stephan 2014).
- ROVER – A version of the SHOP2 domain that makes no use of the SHOP2-specific features but only the standard HTN features (Höller et al. 2020);
- TRANSPORT – An HTN version of the transport domain. In this version of the model, the hierarchy restricts the set of solutions to improve the solutions (it is, e.g., impossible to pick-up or drop a single package more than once), but the main physics are fully given by the actions (Höller et al. 2020);
- ENTERTAINMENT – It describes the problem of assembling a home entertainment system. This is a hierarchical model of the domain where signal flow between devices, e.g., a DVD player and a TV, is not represented in state, but via the hierarchy (Bercher et al. 2014); and
- PCP – A domain that models Post's Correspondence Problem. Since this is an undecidable problem, it can (in general) not be represented in classical planning but HTN planning (Höller et al. 2020).

Table 1 shows the coverage table, including the results of DFS with no move-ordering and no heuristics, as well as the WA* configuration, which has the highest coverage. In these configurations, we use a runtime limit of 20 seconds. Comparing our novel implementation of DFS with move-ordering (DFS*) with a simple DFS algorithm, we show that using heuristic information to manipulate the stack leads to a greater number of solved instances. Table 2 shows the coverage table results of Holler et al. paper (Höller et al. 2019), in which they use a runtime limit of 10 minutes. Comparing the results of WA*, A*, and GBFS, our experiments (Table 1) solve more instances than the state-of-art, even we using less timeout. However, we do not make any changes to the PANDA planner. Table 3 shows the search time required to solve each problem using a maximum time limit of 20 seconds. Figures 1 and 2 show the expanded nodes for the ROVER domain. The latter shows a scatter plot between WA* h^{FF} and WA* h^{LM-cut} expanded nodes. Points below the diagonal indicate that a change of the heuristic has a

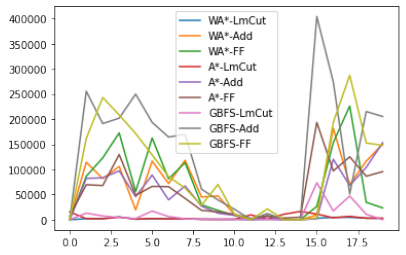

Figure 1: Expanded nodes.

positive effect on the performance, since the h^{FF} heuristic has a higher number of solved instances, points above indicate a negative effect.

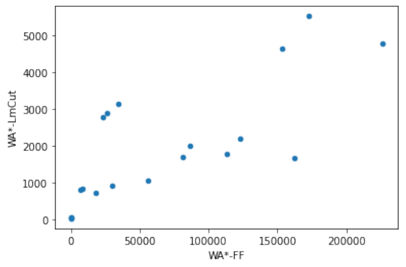

Figure 2: Expanded nodes between WA* h^{FF} and WA* h^{LM-cut}.

Figures 3 and 4 show the expanded nodes for the ROVER domain only using the DFS* with the three heuristics and DFS with no heuristic and no move-ordering, respectively. The latter figure shows the results only for the DFS* h^{Add}, which has the higher number of solved instances between the DFS(*) approaches.

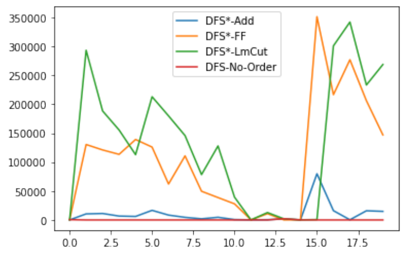

Figure 3: Expanded nodes for DFS(*) algorithms.

Finally, Figure 4 shows the generated nodes per second between the WA* h^{FF} and DFS* h^{Add}, which the first is the most efficient result from the state-of-art, and the latter has the highest coverage among the DFS(*) algorithms.

Related Work

We base our experiments on Holler et al.'s work (Höller et al. 2019). In such paper, the authors propose a progression-based heuristic search to solve HTN planning problems. Furthermore, they contribute with novel search algorithms and a family of heuristics on the PANDA framework. We use such heuristics in our experiments, and we show we can compute 4 instances more using less time limit.

Nau et al. (Nau et al. 2001) describe the SHOP2, a progression-search-based solver for HTNs. The main difference between SHOP2 and PANDA is that SHOP2 does not rely on heuristics to improve the performance of the HTN planner. So, to improve one of the algorithms used on SHOP2, we developed a DFS algorithm (already developed for SHOP2) with a fringe stack that is ordered by heuristic function information, instead of just randomly extracting a method of the fringe, as default DFS algorithms do.

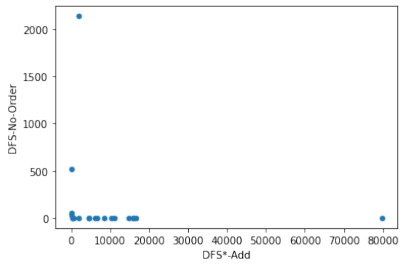

Figure 4: Expanded nodes for DFS* h^{Add} and a simple DFS.

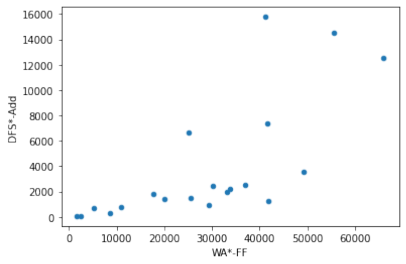

Figure 5: Generated nodes per second between WA* h^{FF} and DFS* h^{Add}.

Conclusion

In this paper, we compare the efficiency of using heuristics in progression-based search algorithms on HTNs. We compare existing results of the state-of-art with a novel implementation of a DFS algorithm with a move-ordering stack fringe (DFS*), where the order of the fringe is dictated by heuristic function information. Our results show that DFS* outperforms ordinary DFS algorithms; however, it solves fewer instances than the state-of-art.

Throughout this research, we (1) carried out a study on how PANDA planner works, (2) understood how each existing heuristics in PANDA works, (3) studied how SHOP2 works, (4) implemented a novel DFS algorithm based on heuristic information, and (5) showed a comparison of these approaches. In future work, we aim to study how good the heuristic function information is and how it influences each approach's fringe.

Domains	instances	WA*			A*			GBFS			DFS*			DFS
		h$^{LM^{cut}}$	h^{Add}	h^{FF}	h$^{LM^{cut}}$	h^{Add}	h^{FF}	h$^{LM^{cut}}$	h^{Add}	h^{FF}	h$^{LM^{cut}}$	h^{Add}	h^{FF}	
entertainment	12	11	**12**	**12**	9	**12**	11	**12**	**12**	**12**	10	10	11	5
pcp	17	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	2	2	1	0
rover	20	6	**7**	**7**	4	6	4	6	6	5	5	5	4	4
satellite	25	23	23	**25**	21	21	**25**	**25**	24	24	24	24	23	17
smartphone	7	**6**	5	5	5	5	5	5	5	5	5	5	5	4
transport	30	**12**	5	**12**	4	9	11	8	4	7	0	2	3	0
umtranslog	21	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
woodworking	11	**10**	**10**	**10**	8	**10**	**10**	**10**	**10**	9	9	**10**	8	6
Total	143	93	97	**106**	95	98	101	101	98	97	76	79	76	57

Table 1: Solved instances from our results with timeout of 20 seconds.

Domains	instances	WA*			A*			GBFS			DFS
		h$^{LM^{cut}}$	h^{Add}	h^{FF}	h$^{LM^{cut}}$	h^{Add}	h^{FF}	h$^{LM^{cut}}$	h^{Add}	h^{FF}	
entertainment	12	11	11	11	8	11	8	**12**	**12**	**12**	6
pcp	17	13	**14**	13	13	13	13	2	2	2	1
rover	20	4	**7**	5	0	4	2	4	4	4	3
satellite	25	23	24	**25**	23	24	24	23	24	24	24
smartphone	7	**5**	**5**	**5**	**5**	**5**	4	**5**	4	**5**	4
transport	30	**13**	4	11	3	7	1	1	1	2	0
umtranslog	21	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**
woodworking	11	**10**	**10**	**10**	8	**10**	**10**	9	9	9	8
Total	144	101	97	**102**	82	96	84	78	78	80	68

Table 2: Solved instances with timeout of 10 minutes from Holler's et al. paper (Höller et al. 2019).

References

Bercher, P.; Biundo, S.; Geier, T.; Hoernle, T.; Richter, F.; Schattenberg, B.; and Nothdurft, F. 2014. Plan, Repair, Execute, Explain — How Planning Helps to Assemble Your Home Theater. In *Proceedings of the Twenty-Fourth International Conferenc on International Conference on Automated Planning and Scheduling*, ICAPS'14, 386–394. AAAI Press.

Bercher, P.; Keen, S.; and Biundo-Stephan, S. 2014. Hybrid Planning Heuristics Based on Task Decomposition Graphs. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*.

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

Georgievski, I.; and Aiello, M. 2015. HTN planning: Overview, comparison, and beyond. *Artificial Intelligence* 222: 124–156.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?

Hoffmann, J.; and Nebel, B. 2011. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research - JAIR* 14.

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

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA Framework for Hierarchical Planning. *KI - Künstliche Intelligenz* .

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. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(06): 9883–9891.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019. On Guiding Search in HTN Planning with Classical Planning Heuristics. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 6171–6175. International Joint Conferences on Artificial Intelligence Organization.

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory amp; Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Nau, D.; Muñoz Avila, H.; Cao, Y.; Lotem, A.; and Mitchell, S. 2001. Total-Order Planning with Partially Ordered Subtasks. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, 425–430. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Trevizan, F.; Thiébaux, S.; and Haslum, P. 2018. Operator Counting Heuristics for Probabilistic Planning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 5384–5388. International Joint Conferences on Artificial Intelligence Organization.

Rover	WA*			A*			GBFS			DFS*			DFS
	$h^{LM^{cut}}$	h^{Add}	h^{FF}	$h^{LM^{cut}}$	h^{Add}	h^{FF}	$h^{LM^{cut}}$	h^{Add}	h^{FF}	$h^{LM^{cut}}$	h^{Add}	h^{FF}	
p1	0.011	0.002	0.003	6.60	0.005	0.453	0.008	0.002	0.002	0.001	0.008	0.012	0.01
p2	0.014	0.007	0.004	4.29	0.014	0.204	0.018	0.003	0.003	0.002	0.022	0.004	0.001
p3	0.027	0.007	0.008	4.41	0.007	0.158	0.009	0.007	0.002	0.046	0.826	0.002	0.038
p4	0.021	0.003	0.004	6.00	0.002	0.3	0.012	0.002	0.006	0.003	0.006	0.016	0.01
p5	5.45	0.602	1.72	22.73	0.132	21.01	21.03	19.33	0.141	0.019	21.09	21.06	-
p6	21.66	21.04	21.01	24.88	21.11	21.02	21.19	21.05	21.05	21.03	21.10	21.03	-
p7	13.39	8.04	21.01	23.93	8.87	21.07	21.43	3.31	21.04	21.00	0.311	21.00	-
p8	23.15	21.03	4.67	21.41	21.06	21.11	21.43	21.04	21.04	21.05	21.81	21.03	-
p9	21.97	21.03	2.65	21.53	21.06	21.05	0.406	21.04	21.06	21.02	21.31	21.13	-
p10	26.00	21.13	21.06	23.35	21.12	21.17	21.14	21.05	21.05	21.10	21.91	21.10	-
p11	30.41	21.17	21.01	29.88	21.05	21.05	21.64	21.07	21.05	21.00	21.77	21.01	-
p12	24.43	21.09	21.00	23.04	21.07	21.01	4.63	21.04	21.02	21.03	22.44	21.12	-
p13	23.53	4.82	21.21	22.37	21.07	21.00	39.82	21.06	21.09	21.10	23.48	21.19	-
p14	23.73	21.01	21.04	29.55	21.04	21.11	21.11	21.06	21.13	21.05	22.80	21.15	-
p15	27.11	21.03	21.08	28.65	21.08	21.09	22.10	21.06	21.09	21.00	22.17	21.18	-
p16	39.49	21.12	21.05	40.26	21.01	21.06	33.84	21.05	21.13	21.05	23.91	21.00	-
p17	49.04	21.06	21.14	46.49	21.20	21.29	27.81	21.18	21.21	21.15	29.80	21.12	-
p18	66.20	21.18	21.25	60.08	21.25	21.10	21.11	21.09	21.02	21.06	25.50	21.03	-
p19	152.06	21.02	22.25	134.56	21.52	22.00	136.45	21.29	21.94	21.22	38.16	21.19	-
p20	252.07	21.92	22.55	278.93	22.12	22.55	239.88	21.26	21.15	21.34	45.40	21.33	-

Table 3: Search time for the ROVER domain.