Focal Discrepancy Search for Learned Heuristics (Extended Abstract)

Matias Greco[1], **Pablo Araneda**[1], **Jorge A. Baier**[1,2]

[1] Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile, Chile
[2] Instituto Milenio Fundamentos de los Datos, Chile
{mogreco, pharaneda}@uc.cl, jabaier@ing.puc.cl

Abstract

Machine learning allows learning accurate, but inadmissible heuristics for hard combinatorial puzzles like the 15-puzzle, the 24-puzzle, and Rubik's cube. In this paper, we investigate how to exploit these learned heuristics in the context of heuristic search with suboptimality guarantees. Specifically, we study how Focal Search (FS), a well-known bounded-suboptimal search algorithm can be modified to better exploit inadmissible learned heuristics. We propose to use *Focal Discrepancy Search* (FDS) in the context of learned heuristics, which uses a discrepancy function, instead of the learned heuristic, to sort the focal list. In our empirical evaluation, we evaluate FS and FDS using DeepCubeA, an effective learned heuristic for the 15-puzzle. We show that FDS substantially outperforms FS. This suggests that in some domains, when a highly accurate heuristics is available, one should always consider using discrepancies for better search.

Introduction

Recent work has shown that machine learning is an effective approach to learning heuristics estimators in different scenarios, such as domain-independent automated planning (Ferber, Helmert, and Hoffmann 2020; Shen, Trevizan, and Thiébaux 2020) and combinatorial puzzles such as Rubik's cube and the sliding tile puzzle (Agostinelli et al. 2019; Arfaee, Zilles, and Holte 2011).

Given a learned heuristic function, a natural question to ask is *how to exploit such a learned heuristic within a bounded-suboptimal search algorithm*; i.e, a complete algorithm that provides guarantees on the quality of the solution. This question is challenging because learned heuristics, even if highly accurate, cannot be assumed to be admissible.

Learned inadmissible heuristics can be exploited within search in various ways (e.g. Thayer and Ruml 2011; Spies et al. 2019; Aine et al. 2016). One way of exploiting an inadmissible heuristic is with Focal Search (FS) (Pearl and Kim 1982), a bounded-suboptimal search algorithm which uses two priority queues, one which is sorted just like A*'s open using an admissible heuristics, and another—the `FOCAL` list—which can be sorted according to any priority, in particular, an inadmissible learned heuristic. In FS's main loop nodes are always extracted from `FOCAL` while the open list is used to provide the suboptimality guarantee.

Spies et al. (2019) proposed to use the learned heuristic to sort `FOCAL`. Araneda, Greco, and Baier (2021) proposed an approach to exploit learned *policies*, rather than a heuristic, in the context of FS. They show that an effective way to incorporate policies in FS is by exploiting the notion of discrepancy (Harvey and Ginsberg 1995), which given a state-action sequence $s_0a_0s_1a_1 \dots s_n$ counts the number of times a_i would have not been chosen by the learned policy at state s_i. Their work assumed no learned heuristic are available.

In this paper, strongly inspired by the work of Araneda, Greco, and Baier (2021), we propose to use Focal Discrepancy Search (FDS) in the context of bounded-suboptimal search with learned heuristics, which sorts `FOCAL` using a discrepancy score computed from the learned heuristic.

We evaluate our approach using the extremely effective (but inadmissible) learned heuristic of DeepCubeA (Agostinelli et al. 2019) for the 15-Puzzle. Our results show that the straightforward way of incorporating a heuristic into FS, that is, using it to sort the `FOCAL` list (as done by Spies et al. 2019) is *not* the best performing one and FDS, expands fewer nodes and returns better solutions.

Focal Discrepancy Search for Learned Heuristics

A natural way of using a learned heuristic h_{nn} in FS is to sort `FOCAL` by h_{nn}, as Spies et al. (2019) propose. Focal Discrepancy Search (FDS), instead, is a version of FS that sorts `FOCAL` by the discrepancy associated with the path of each state. More formally, if s is a state in `FOCAL`, at any point during the execution of FS, its priority is given by $disc(path(s))$. FDS has been proposed recently in the context of learned policies (Araneda, Greco, and Baier 2021), in which the notion of discrepancy is not defined in terms of a heuristic but rather in terms of a policy.

Given a heuristic h, a discrepancy (Harvey and Ginsberg 1995) occurs when an algorithm chooses to descend to a child, which is not the one with minimum h-value among its siblings. Formally, the discrepancy of a path $\sigma = s_1s_2 \dots s_n$, denoted by $disc(\sigma)_h$, is $\sum_{i=1}^{n-1} disc(s_i, s_{i+1})$, where $disc(s,t)$ is equal to 0 if $h(t) = \min_{s' \in Succ(s)} h(s')$ and is equal to 1 otherwise.

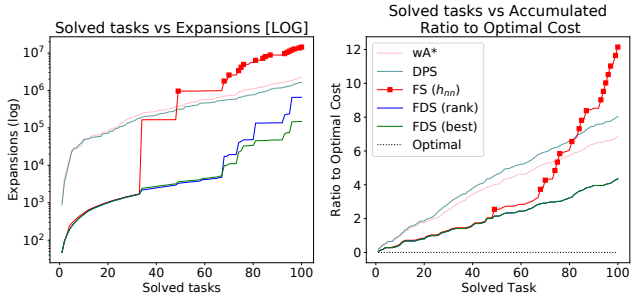

Figure 1: Results in 15-Puzzle using DeepCubeA as learned heuristic

	$w = 1.5$			$w = 2.0$		
	Cov. %	Exp	Subopt%	Cov. %	Exp	Subopt%
WA*	**100**	12325	7.59	**100**	3558	22.62
DPS	**100**	9957	8.61	**100**	2518	27.82
FS (h_{nn})	83	3284	4.31	**100**	56	6.18
FDS (best)	**100**	76	4.21	**100**	58	5.72
FDS (rank)	**100**	444	3.74	**100**	57	**5.47**

Table 1: Results for the 15-puzzle.

Some researchers (e.g., Karoui et al. 2007), for non-binary trees, have considered counting discrepancies according to their successor rank. Instead of considering a discrepancy as a binary function, they sort the successors in ascending order according to their h-value, and consider that each successor yields a discrepancy equal to the index in the order (we assume the first index is 0). Let $rank(s, t)$ denote the rank of child t of s, with respect to the heuristic function h. Then, we define the rank of a path $\sigma = s_1 s_2 \dots s_n$ as $r(\sigma) = \sum_{i=1}^{n-1} rank(s_i, s_{i+1})$.

Below *FDS(best)* and *FDS(rank)* denote the algorithm that result by sort its corresponding FOCAL list by $disc(path(s))$ and $r(path(s))$, respectively.

Experimental Results

We evaluate *FDS* in the 15-puzzle using the pre-trained model of DeepCubeA (Agostinelli et al. 2019), a very effective learned heuristic for different puzzle problems. The pre-trained models are publicly available[1]. We compare our algorithms with $FS(h_{nn})$ which use the learned heuristic as $h_{\texttt{FOCAL}}$ (as Spies et al. (2019) has described), WA* and DPS with the Linear Conflict (Hansson, Mayer, and Yung 1992) as admissible heuristic. For the evaluations, we use the 100 Korf's instances for the 15-Puzzle (Korf 1985). All algorithms were implemented in Python 3, and the experiments were run on an Intel Xeon E5-2630 machine with 64GB RAM, using a single CPU core and one GPU Nvidia Quadro RTX 5000. We use a 30-minute timeout.

Table 1 shows the percentage of instances each algorithm solves (coverage), the average number of expansions and the average suboptimality obtained on the solved instances by all algorithms for two suboptimality bounds. With suboptimality bound $w = 2.0$, all algorithms solve all instances, but the algorithms that use the learned heuristics outperform by 2 and 3 orders of magnitude the number of expansions WA* (LC) and WA* (Man), resp. In addition, they obtain results 16% better in quality. With a suboptimality bound $w = 1.5$ *FDS(best)* and *FDS(rank)* solve the complete problem set and outperform WA* by one and two order of magnitude with respect to the number of expansions on the solved problems by all algorithms, resp. In addition, they obtain results, on average, 4% better in terms of solution quality. FS(h_{nn}) shows poor performance, obtaining just 83% of coverage and perform on average, one order of magnitude more expansions than FDS.

Figure 1 summarizes the results (cumulative expansions, and cumulative suboptimality) obtained by the algorithms for suboptimality bound $w = 1.5$. We use a square mark to indicate the search algorithm failed to solve a particular instance. When a problem is not solved, we add the expansions performed until the timeout, and we consider the suboptimality of such a solution to be w times the optimal cost.

The results show that *FDS(best)* outperforms *WA** by one order of magnitude with respect to the number of expansions. Nevertheless, both require a similar time. This is because expansions are one order of magnitude slower when using the learned heuristic. We observe that *FDS(best)* loses advantage over *WA** for the last 30 instances. This may be due to the fact that many search states have the same f-values, and many expansions are needed to make progress in the search. Regarding to the quality of the solutions, we observe that both versions of *FDS* obtain solutions of better quality than other algorithms

Summary and Conclusions

We presented Focal Discrepancy Search (FDS), a method that uses FS to exploit a given learned heuristic more effectively than FS used directly with the learned heuristic. We evaluated our approach using DeepCubeA, a recent framework that learns very accurate heuristics for puzzle games with reinforcement learning. We show that FDS outperforms other BSS algorithms, such as *WA**, up to four orders of magnitude. Also, we show that FDS outperforms *FS(h_{nn})*, which uses the heuristic value to sort the FOCAL list. We perform experiments using different admissible heuristic functions and suboptimality bounds, but it can be also applied to bounded cost. Due to calculating the learned heuristic value may be time-consuming, in a future work, we will study how to accelerate the time spent to calculate the heuristic value in the context of bounded suboptimal search.

Acknowledgements

Matias Greco was supported by the National Agency for Research and Development (ANID) / Doctorado Nacional / 2019 - 21192036. We also thank to Centro Nacional de Inteligencia Artificial CENIA, FB210017, BASAL, ANID, for partially funding the authors.

[1]https://github.com/forestagostinelli/DeepCubeA

References

Agostinelli, F.; McAleer, S.; Shmakov, A.; and Baldi, P. 2019. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8): 356–363.

Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; and Likhachev, M. 2016. Multi-Heuristic A*. *International Journal of Robotics Research*, 35(1-3): 224–243.

Araneda, P.; Greco, M.; and Baier, J. 2021. Exploiting Learned Policies in Focal Search. In *Proceedings of the 14th Symposium on Combinatorial Search (SoCS)*.

Arfaee, S. J.; Zilles, S.; and Holte, R. C. 2011. Learning heuristic functions for large state spaces. *Artificial Intelligence*, 175(16-17): 2075–2098.

Ferber, P.; Helmert, M.; and Hoffmann, J. 2020. Neural Network Heuristics for Classical Planning: A Study of Hyperparameter Space. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 2346–2353. IOS Press.

Hansson, O.; Mayer, A.; and Yung, M. 1992. Criticizing solutions to relaxed models yields powerful admissible heuristics. *Information Sciences*, 63(3): 207–227.

Harvey, W. D.; and Ginsberg, M. L. 1995. Limited discrepancy search. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*.

Karoui, W.; Huguet, M.; Lopez, P.; and Naanaa, W. 2007. YIELDS: A Yet Improved Limited Discrepancy Search for CSPs. In Hentenryck, P. V.; and Wolsey, L. A., eds., *4th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, volume 4510 of *LNCS*, 99–111. Springer.

Korf, R. E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27(1): 97–109.

Pearl, J.; and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(4): 392–399.

Shen, W.; Trevizan, F. W.; and Thiébaux, S. 2020. Learning Domain-Independent Planning Heuristics with Hypergraph Networks. In Beck, J. C.; Buffet, O.; Hoffmann, J.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 574–584. AAAI Press.

Spies, M.; Todescato, M.; Becker, H.; Kesper, P.; Waniek, N.; and Guo, M. 2019. Bounded Suboptimal Search with Learned Heuristics for Multi-Agent Systems. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 2387–2394. AAAI Press.

Thayer, J. T.; and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 674–679. IJCAI/AAAI.