K-Focal Search for Slow Learned Heuristics (Extended Abstract)

Matias Greco[1], **Jorge Toro**[1], **Carlos Hernández-Ulloa**[3], **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
[3] Facultad de Ingeniería y Tecnología, Universidad San Sebastián, Chile
{mogreco, jitoro4}@uc.cl, carlos.hernandez@uss.cl, jabaier@ing.puc.cl

Abstract

Learned heuristics, though inadmissible, can provide very good guidance for bounded-suboptimal search. Given a single search state s and a learned heuristic h, evaluating $h(s)$ is typically very slow relative to expansion time, since state-of-the-art learned heuristics are implemented as neural networks. However, by using a Graphics Processing Unit (GPU), it is possible to compute heuristics using batched computation. Existing approaches to batched heuristic computation are specific to satisficing search and have not studied the problem in the context of bounded-suboptimal search. In this paper, we present K-Focal Search, a bounded suboptimal search algorithm that in each iteration expands K nodes from the FOCAL list and computes the learned heuristic values of the successors using a GPU. We experiment over the Rubik's Cube domain using DeepCubeA, a very effective inadmissible learned heuristic. Our results show that K-Focal Search benefits both from batched computation and from the diversity in the search introduced by its expansion strategy. Over standard FS, it improves runtime by a factor of 6, expansions by up to three orders of magnitude, and finds better solutions, keeping the theoretical guarantees of Focal Search.

Introduction

Recent work has shown that neural networks can be trained to produce accurate and effective heuristics to guide search in a variety of search problems, including domain-independent planning (Shen et al. 2019; Ferber, Helmert, and Hoffmann 2020), Sokoban (Groshev et al. 2018), Rubik's cube, and the sliding-tile puzzles (Agostinelli et al. 2019).

Despite their effectiveness, neural-net heuristics have important drawbacks. A first drawback is that their computation is extremely slow relative to other operations carried out during search. Indeed, our data shows that up to 39% of search runtime can be consumed computing a neural-net heuristic. Slow heuristic computation has been tackled by previous work by exploiting the computational power of Graphics Processing Units (GPUs), which allows evaluating neural nets in parallel. Batched Weighted A* (BWAS) (Agostinelli et al. 2019) is an instance of KBFS (Felner, Kraus, and Korf 2003), that is similar to Weighted A* in which K states are extracted from the OPEN list in each iteration of the main loop. After extraction, if the goal has not been found, all K states are sequentially expanded, and the heuristic values for the union of their successors are evaluated via batched computation. Batched computation reduces heuristic computation times per state significantly, ultimately reducing total runtime.

BWAS does not provide suboptimality guarantees, as a consequence of a second drawback of neural-net heuristics: they are not admissible, thus they cannot be directly used by the many search algorithms that exploit admissibility to provide quality guarantees. Recently, Spies et al. [2019] and Araneda, Greco, and Baier; Greco, Araneda, and Baier [2021; 2022] proposed the use of neural-net heuristics in combination with admissible heuristics in Focal Search (FS). However, these approaches do not address the problem of slow heuristic computation.

In this paper we propose K-Focal Search (K-FS), the first search algorithm that exploits neural-net heuristics while (1) providing guarantees on solution quality and (2) addressing the issue of slow heuristic computation using a GPU. K-FS generalizes Focal Search (Pearl and Kim 1982). Instead of expanding the best node from FOCAL, it expands the best K nodes from FOCAL, producing a batch of successors, whose heuristic values are computed by the GPU.

We prove that K-FS is a complete bounded suboptimal algorithm; specifically, given a parameter $w \geq 1$, the solution returned are w-optimal, i.e., if a solution with cost c is returned and c^* is the cost of an optimal solution, then $c \leq wc^*$. Empirically, we show the applicability of K-FS at solving the Rubik's cube using DeepCubeA (Agostinelli et al. 2019), a very effective learned heuristic. We show that K-FS outperforms FS across a number of performance indicators, including runtime, and percentage of time spent in heuristic computation.

K-Focal Search

K-Focal Search (K-FS(k)) is a generalization of Focal Search and inherits some properties from FS and K-BFS (Felner, Kraus, and Korf 2003). Instead of selecting one node for expansion, K-FS(k) extracts the best k nodes from FOCAL, and unless the goal is among the extracted states, it expands all such states, computing the heuristic values of all their successors using GPU batched computation. In case

	w=2.5						
	Cov.	EC	Exp.	Cost	Time	h time	%
WA*	0	-	252753	-	1800.00	-	-
DPS	0	-	263865	-	1800.00	-	-
FS (seq)	9	148981	148981	26.22	1695.49	1.95E-03	39.0
K-FS(1)	11	201654	201654	26.36	1671.49	5.43E-04	14.7
K-FS(5)	25	40362	201805	26.24	1504.82	1.62E-04	5.5
K-FS(10)	31	18808	188074	26.23	1373.95	8.10E-05	2.9
K-FS(25)	50	5965	149093	25.80	1059.78	3.31E-05	1.3
K-FS(100)	75	839	83673	25.44	598.45	1.10E-05	0.4
K-FS(120)	77	667	79838	25.51	566.62	9.83E-06	0.4
K-FS(240)	85	245	58212	24.92	415.59	7.65E-06	0.4
K-FS(480)	89	85	39507	24.35	278.50	6.55E-06	0.4

Table 1: Results on Rubik's Cube

where the FOCAL list contains less than k nodes, it selects for expansion all nodes in the FOCAL list. K-Focal Search with $k = 1$ [K-FS(1)] executes the exact procedure that Focal Search, but explodes the GPU resource.

Like FS, K-FS receives two heuristics as input: an admissible heuristic h, and a neural-net heuristic $h_{\texttt{FOCAL}}$. In addition, it receives a parameter k to control the number of expansions in each expansion cycle. If a node s selected for expansion is a goal state, this is returned; otherwise, it is added to a TO−EXPAND set.

In the successor generation step, the algorithm generates the successors of all states in TO−EXPAND and adds them to OPEN. If the successor $t \in Succ(s)$ satisfies the suboptimality bound (i.e. it is such that $f(t) \leq w * f_{min}$), then t is added to the BATCH set. The value of f_{min} may increase during execution, and since there exist states in OPEN that are not in FOCAL, the algorithm verifies if f_{min} increased and inserts in BATCH those states whose f-value is lower than or equal to $w * f_{min}$. Finally, the $h_{\texttt{FOCAL}}$-values of all states in BATCH are computed using the GPU and added to FOCAL.

Besides the computation of $h_{\texttt{FOCAL}}$ using batched computation, an important difference between K-FS and FS is that more nodes are expanded in the same *expansion cycle*; that is, in the same iteration of the search loop. This has the potential of radically changing the order in which nodes are expanded, since the children of nodes that would have not been expanded by FS are added to FOCAL. We prove that KFS is a complete bounded-suboptimal algorithm.

Theorem 1. *K-FS is complete and w-optimal when an admissible heuristic is used to sort* OPEN.

Because K-FS does not change the conditions under which a state is added to FOCAL or OPEN, every state s in FOCAL is such that $f(s) < w \min_{t \in Open}\{g(t) + h(t)\}$ and thus when a goal is extracted from FOCAL, it follows that the suboptimality bound is met.

Empirical Evaluation

Our empirical evaluation seeks to evaluate the performance of our algorithm and the impact of the k parameter in the performances with two different suboptimality bounds. We evaluated our algorithm on the Rubik's cube domain and we use the trained models of DeepCubeA (Agostinelli et al. 2019) as learned heuristic. We use a Pattern Database (Korf 1997) as admissible heuristic. For the evaluations, we use 100 random instances from the DeepCubeA test set, which was generated by randomly scrambling the goal state between 1,000 and 10,000 times. The 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. For all experiments, we use a 30-minute timeout. Our algorithms are compared with Focal Search (*FS*) using the same neural-net heuristic; and two other state-of-the-art bounded-suboptimal search algorithms: Weighted A* (*WA**) and Dynamic Potential Search (*DPS*) (Gilon, Felner, and Stern 2016).

Table 1 shows the coverage (percentage of solved instances), average expansion cycles (EC), average nodes expansions (exp.), average cost in the solved instances by the algorithm, average runtime, average time spent to compute the learned heuristic per state (h time), and the percentage of the search time which was used to compute the learned heuristic (%), for all instances in the domain. If an algorithm does not solve an instance, we add the node expansions, expansion cycles, and time spent until the time limit.

The results show that, with suboptimality bound $w = 2.5$, WA* and DPS can not solve instances within the deadline. On the other hand, FS (seq) could solve 9% of the instances and K-FS(1) just two more because batch processing accelerates the computation of the learned heuristic, per state. On average, FS (seq) spent 39% of the search time calculating the learned heuristic, and K-FS(1) reduces the time to 14.7%. As k increases, the coverage increases up to 89% using K-FS(480) and performs, on average, one order of magnitude fewer expansions than K-FS(1), finding better cost solutions. The time spent calculating the heuristic, per each state, was reduced by almost three orders of magnitude as k increases, and it went from 39% of the search time to only 0.4

Conclusions and Future Work

In this paper, we presented K-Focal Search, a generalization of Focal Search algorithm which expands k nodes from FOCAL at every expansion cycle. The algorithm builds a batch of states to calculate a learned heuristic exploiting the batch processing capability of a GPU.

On the experimental side, we demonstrate the effectiveness of our algorithm the Rubik's Cube domain using DeepCubeA, a very effective inadmissible learned heuristic. We show that our approach outperforms others bounded-suboptimal heuristic search algorithms such as WA* and DPS and FS using the learned heuristic by two orders of magnitude in the number of expansions and three orders of magnitude in the time spent computing the heuristic. As the k value increases, the number of expansions cycles always decrease, but the number of expansions may increase. For that reason, in future work we will explore how to calibrate the k parameter. As future work, we seek to move this approach to a concurrent algorithm that can generate the successors and explore the different zones of the state space in parallel, like to PRA* (Evett et al. 1995) does.

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.

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.

Araneda, P.; Greco, M.; and Baier, J. A. 2021. Exploiting Learned Policies in Focal Search. In Ma, H.; and Serina, I., eds., *Proceedings of the Fourteenth International Symposium on Combinatorial Search, SOCS 2021, Virtual Conference [Jinan, China], July 26-30, 2021*, 2–10. AAAI Press.

Evett, M. P.; Hendler, J. A.; Mahanti, A.; and Nau, D. S. 1995. PRA*: Massively Parallel Heuristic Search. *J. Parallel Distributed Comput.*, 25(2): 133–143.

Felner, A.; Kraus, S.; and Korf, R. E. 2003. KBFS: K-Best-First Search. *Annals of Math and Artificial Intelligence*, 39(1-2): 19–39.

Ferber, P.; Helmert, M.; and Hoffmann, J. 2020. Reinforcement Learning for Planning Heuristics. In *Proceedings of the 1st Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*, 119–126. University_of_Basel.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic Potential Search - A New Bounded Suboptimal Search. In Baier, J. A.; and Botea, A., eds., *Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, Tarrytown, NY, USA, July 6-8, 2016*, 36–44. AAAI Press.

Greco, M.; Araneda, P.; and Baier, J. 2022. Focal Discrepancy Search for Learned Heuristics. In *Proceedings of the Fourteenth International Symposium on Combinatorial Search, SOCS 2022, Vienna, Austria, July 21-23, 2022*.

Groshev, E.; Goldstein, M.; Tamar, A.; Srivastava, S.; and Abbeel, P. 2018. Learning Generalized Reactive Policies Using Deep Neural Networks. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M. T. J., eds., *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, 408–416. AAAI Press.

Korf, R. E. 1997. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. In Kuipers, B.; and Webber, B. L., eds., *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*, 700–705. AAAI Press / The MIT Press.

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.; Toyer, S.; Thiébaux, S.; and Xie, L. 2019. Guiding Search with Generalized Policies for Probabilistic Planning. In Surynek, P.; and Yeoh, W., eds., *Proceedings of the 12th Symposium on Combinatorial Search (SoCS)*, 97–105. 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 *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2387–2394. AAAI Press.