Bi-objective Search with Bi-directional A* (Extended Abstract)*

Saman Ahmadi,[1,2] **Guido Tack,**[1] **Daniel Harabor,**[1] **Philip Kilby**[2]
[1] Department of Data Science and Artificial Intelligence, Monash University, Australia
[2] CSIRO Data61, Australia
{saman.ahmadi, guido.tack, daniel.harabor}@monash.edu, philip.kilby@data61.csiro.au

Abstract

Bi-objective search is a problem of finding a set of optimal solutions in a two-dimensional domain. This study proposes several enhancements to the state-of-the-art bi-objective search with A* and develops its bi-directional variant. Our experimental results on benchmark instances show that our enhanced algorithm is on average five times faster than the state of the art bi-objective search algorithms.

Introduction

Bi-objective search is a well-know problem in AI with various real-world applications such as planning paths for electric vehicles, where time and energy-efficiency must be simultaneously optimised (Shen et al. 2019). The main challenge in such cases is that solutions are seldom unique. Thus instead of computing a single best path, our task instead is to compute all non-dominated paths in a *Pareto-optimal set*. A survey of bi-objective one-to-all shortest path algorithms appears in Raith and Ehrgott (2009). For point-to-point bi-criteria problems there exist other more recent works which are considered state-of-the-art. Bi-objective Dijkstra (Sedeño-Noda and Colebrook 2019) and Bi-objective A* (BOA*) (Ulloa et al. 2020) are such algorithms. In contrast to eager dominance checking, as in the Bi-objective Dijkstra algorithm, BOA* uses a *lazy dominance checking*, improving on a routine originally developed for multi-objective search (Pulido, Mandow, and Pérez-de-la-Cruz 2015).

In this study, we present Bi-Objective Bi-directional A* (BOBA*), a bi-directional extension of the BOA* algorithm that uses different objective orders and includes several new heuristics to speed up the search. Our experiments on a set of 1,000 large test cases show that BOBA* can solve all of the cases to optimality, outperforming the state-of-the-art algorithms in both runtime and memory requirement.

Bi-objective Bi-directional Search

For a directed bi-objective graph $G = (S, E)$ with a finite set of states S and a set of edges $E \subseteq S \times S$, the point-to-point bi-objective search problem is to find the set of Pareto-optimal solution paths from $start \in S$ to $goal \in S$ that are not dominated by any solution for both objectives. Every edge $e \in E$ has two non-negative attributes accessed via the cost function $\mathbf{cost} : E \rightarrow \mathbb{R}^+ \times \mathbb{R}^+$. A path is a sequence of states $s_i \in S$ with $i \in \{1, \ldots, n\}$. The cost of path $p = \{s_1, s_2, s_3, \ldots, s_n\}$ is then defined as the sum of corresponding attributes on all the edges constituting the path as $\mathbf{cost}(p) = \sum_{i=1}^{n-1} \mathbf{cost}(s_i, s_{i+1})$. We define our search objects to be *nodes*. We perform a systematic search by *expanding* nodes in best first order. Each expansion operation *generates* a set of successor nodes, which are added into an *Open* list. The *Open* list sorts the nodes according to their $\mathbf{f}$-values in an ascending order, for the purpose of further expansion. As with other A*-based algorithms, we compute $\mathbf{f}$-values using a consistent and admissible heuristic function $\mathbf{h} : S \rightarrow \mathbb{R}^+ \times \mathbb{R}^+$ (Hart, Nilsson, and Raphael 1968). In other words, for node x associated with state $s(x)$, we have $\mathbf{f}(x) = \mathbf{g}(x) + \mathbf{h}(s(x))$ where $\mathbf{g}(x)$ is the cost of a concrete path from $start$ to state $s(x)$ and $\mathbf{h}(s(x))$ is a lower bound on the cost of paths from state $s(x)$ to $goal$. Moreover, in bi-objective search, cost values have two components which means that every (boldface) cost is a tuple, eg. $\mathbf{f} = (f_1, f_2)$.

Main idea: Our Bi-objective Bi-directional A* algorithm (BOBA* for short) relies on a set of single-objective one-to-all heuristics, which we compute in the forward and backward directions. These heuristics inform the primary objective costs of two subsequent uni-directional BOA* searches (one forward, one backward). The two BOA* are run in parallel and each has a different objective order: $((f_1, f_2)$ and $(f_2, f_1))$. The searches independently explore the graph towards the opposite end, with each primary objective cost settled in one direction being used to improve secondary objective bounds in the other direction. The output is the set of all non-dominated solutions, found in either direction.

Initialisation Phase

During an initialisation step BOA* computes a lower-bounding heuristic function to guide its main search. This procedure computes reverse/backward distances for the primary objective, from the $goal$ to all other states in the graph. BOBA* proceeds similarly but requires 4 heuristics: one for each objective and in each direction. Here we rely on a fast initialisation scheme that replaces Dijkstra search with cost

*See our full paper (Ahmadi et al. 2021a) for details.
Research at Monash University is supported by the Australian Research Council (ARC) under grant numbers DP190100013 and DP200100025 as well as a gift from Amazon.

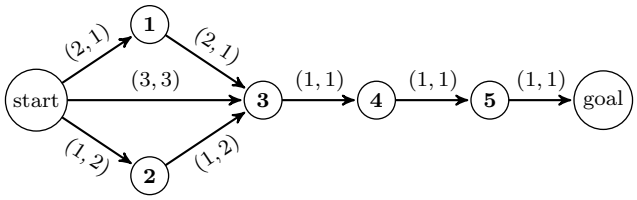

Figure 1: An example graph with **cost** on the edges

bounded A* (Ahmadi et al. 2021b). For a further speedup we exploit already computed heuristic values. Thus, after completing a search in the one direction, the next cost-bounded search, in the opposite direction, can ignore as out-of-bound any states not expanded by the previous search.

Main Search

There are four main ingredients that allow BOBA* to compute unique Pareto-optimal solutions more efficiently.

Early solution update: This strategy allows the search to identify non-dominated solutions before expanding the *goal* state. This is done by coupling nodes with a complementary shortest path. Consider Figure 1 where the forward BOA* search expands nodes in (f_1, f_2) order; i.e. f_1 is the primary cost, f_2 is the secondary cost and the initial upper bounds are $\overline{f_1}, \overline{f_2} = \infty$. Before expanding *start* we exploit the heuristic function for the primary objective to identify an f_1-optimal complementary path with $\mathbf{cost} = (5, 7)$. Since this solution is within the upper bounds we add *start* to the solution set and update the secondary upper bound $\overline{f_2} = 7$ (the complementary path is extracted at the end by following stored backpointers). This strategy can be further improved by avoiding repeated expansion of states with only one non-dominated path to *goal* (eg. states $3, 4, 5$ would normally be expanded thrice). In this example BOBA* expands only one node (resp. *start*) before it can return a unique Pareto-optimal set with optimum costs $\{(5, 7), (6, 6), (7, 5)\}$.

Secondary heuristic tuning: Each time we expand a node in one direction its primary objective cost can be propagated to improve the secondary bounds for the search in the opposite direction. For example, when the forward search expands *start* in Figure 1 it can immediately prune state (2) as its estimated secondary cost is not within the previously established bound; i.e., $f_2(\mathbf{2}) = 2 + 5 \not< 7$. Notice that state (3) is now no longer reachable via its cost-optimum path, originally via state (2). When the forward search eventually picks state (3) from *Open*, we compare the achieved f_1 cost to the f_1 lower-bound estimate of state (3) and notice it is larger. This information can now be propagated so as to better inform the search in the opposite direction. In other words, we raise the secondary objective cost, from *start* to state (3). After this update, the backward search can potentially avoid redundant expansion of state (3) depending on its secondary upper bound $\overline{f_1}$ (as it runs in (f_2, f_1) order).

More efficient *Open* list: To improve the performance of search we propose to replace the conventional heap-based *Open* list of BOA* with a bucket-based implementation (Denardo and Fox 1979). In contrast to other similar algorithms, where the bucket list is regularly resized and the list is sparsely populated, for the significant number of (cost-bounded) nodes in our problem we expect to see almost all of the buckets filled. This is because the upper and lower bounds on the f-values of the nodes in BOBA* are known prior to its main searches, which allows us to use a small, fixed range of bucket values.

Memory efficient backtracking: Since BOBA* only expands nodes once, we recycle the memory used to store nodes and keep their backtracking information in other compact data structures. We do this by only storing nodes' parent state and parent path id.

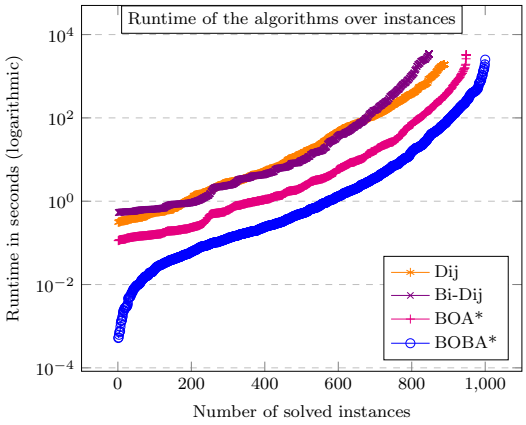

Figure 2: Cactus plot of algorithms' performance.

Experimental Results

We implemented our BOBA* algorithms based on a parallel framework in C++ and compared that with bi-objective Dijkstra (Dij) and bi-directional Dijkstra (Bi-Dij), and bi-objective A* (BOA*) from the literature, all implemented in C. We evaluated all of the algorithms on 1,000 random test cases from 10 instances in the 9th DIMACS challenge (DIMACS 2005) with (*distance*, *time*) as objectives. We ran our experiments on an Intel Xeon E5-2660V3 processor running at 2.6 GHz and with 128 GB of RAM, in a one-hour timeout.

Figure 2 depicts the algorithms' performance over the solved instances for CPU time. The results show that BOBA* delivers superior performance to its competitors by solving all of the instances to optimality within the time limit. Compared to BOA*, BOBA* is on average five times faster and completes the task eight times more efficiently in terms of memory. BOBA* also shows a massive speed up in the easy cases due to its efficient initialisation. It solves 282 cases before BOA* solves its easiest case.

Conclusion

This paper introduced BOBA*, a bi-directional algorithm for bi-objective search. We enrich BOBA* with more efficient approaches for both the initial heuristics and the main search. Our experiments show that BOBA* outperforms the state-of-the-art algorithms in both runtime and memory use, solving all of the benchmarks to optimality.

References

Ahmadi, S.; Tack, G.; Harabor, D.; and Kilby, P. 2021a. Bi-objective Search with Bi-directional A. *CoRR* abs/2105.11888. URL https://arxiv.org/abs/2105.11888.

Ahmadi, S.; Tack, G.; Harabor, D. D.; and Kilby, P. 2021b. A Fast Exact Algorithm for the Resource Constrained Shortest Path Problem. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 12217–12224. AAAI Press. URL https://ojs.aaai.org/index.php/AAAI/article/view/17450.

Denardo, E. V.; and Fox, B. L. 1979. Shortest-Route Methods: 1. Reaching, Pruning, and Buckets. *Oper. Res.* 27(1): 161–186. doi:10.1287/opre.27.1.161. URL https://doi.org/10.1287/opre.27.1.161.

DIMACS. 2005. 9th DIMACS Implementation Challenge - Shortest Paths. http://users.diag.uniroma1.it/challenge9. Accessed: 2021-05-24.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 4(2): 100–107. doi:10.1109/TSSC.1968.300136. URL https://doi.org/10.1109/TSSC.1968.300136.

Pulido, F. J.; Mandow, L.; and Pérez-de-la-Cruz, J. 2015. Dimensionality reduction in multiobjective shortest path search. *Comput. Oper. Res.* 64: 60–70. doi:10.1016/j.cor.2015.05.007. URL https://doi.org/10.1016/j.cor.2015.05.007.

Raith, A.; and Ehrgott, M. 2009. A comparison of solution strategies for biobjective shortest path problems. *Comput. Oper. Res.* 36(4): 1299–1331. doi:10.1016/j.cor.2008.02.002. URL https://doi.org/10.1016/j.cor.2008.02.002.

Sedeño-Noda, A.; and Colebrook, M. 2019. A biobjective Dijkstra algorithm. *Eur. J. Oper. Res.* 276(1): 106–118. doi:10.1016/j.ejor.2019.01.007. URL https://doi.org/10.1016/j.ejor.2019.01.007.

Shen, L.; Shao, H.; Wu, T.; Lam, W. H.; and Zhu, E. C. 2019. An energy-efficient reliable path finding algorithm for stochastic road networks with electric vehicles. *Transportation Research Part C: Emerging Technologies* 102: 450–473.

Ulloa, C. H.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazo, L.; and Koenig, S. 2020. A Simple and Fast Bi-Objective Search Algorithm. 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*, 143–151. AAAI Press. URL https://aaai.org/ojs/index.php/ICAPS/article/view/6655.