Anytime Approximate Bi-Objective Search

Han Zhang[1], **Oren Salzman**[2], **T. K. Satish Kumar**[1], **Ariel Felner**[3], **Carlos Hernández Ulloa**[4], **Sven Koenig**[1]

[1] University of Southern California
[2] Technion - Israel Institute of Technology
[3] Ben-Gurion University
[4] Universidad San Sebastian
zhan645@usc.edu, osalzman@cs.technion.ac.il, tkskwork@gmail.com, felner@bgu.ac.il, carlos.hernandez@uss.cl, skoenig@usc.edu

Abstract

The Pareto-optimal frontier for a bi-objective search problem instance consists of all solutions that are not worse than any other solution in both objectives. The size of the Pareto-optimal frontier can be exponential in the size of the input graph, and hence finding it can be hard. Some existing works leverage a user-specified approximation factor ε to compute an approximate Pareto-optimal frontier that can be significantly smaller than the Pareto-optimal frontier. In this paper, we propose an anytime approximate bi-objective search algorithm, called Anytime Bi-Objective A*-ε (A-BOA$^*_\varepsilon$). A-BOA$^*_\varepsilon$ is useful when deliberation time is limited. It first finds an approximate Pareto-optimal frontier quickly, iteratively improves it while time allows, and eventually finds the Pareto-optimal frontier. It efficiently reuses the search effort from previous iterations and makes use of a novel pruning technique. Our experimental results show that A-BOA$^*_\varepsilon$ substantially outperforms baseline algorithms that do not reuse previous search effort, both in terms of runtime and number of node expansions. In fact, the most advanced variant of A-BOA$^*_\varepsilon$ even slightly outperforms BOA*, a state-of-the-art bi-objective search algorithm, for finding the Pareto-optimal frontier. Moreover, given only a limited amount of deliberation time, A-BOA$^*_\varepsilon$ finds solutions that collectively approximate the Pareto-optimal frontier much better than the solutions found by BOA*.

1 Introduction and Related Work

Bi-objective search is a generalization of the single-objective search used for shortest-path computations. We are given a directed graph with two costs annotating each edge, a start vertex, and a goal vertex. Given a solution path π, we use $c_1(\pi)$ and $c_2(\pi)$ to denote the accumulated first and second costs of the edges of π, respectively. A solution path π is better than, i.e., *dominates*, another solution path π' if and only if (i) $c_1(\pi) \leq c_1(\pi')$ and $c_2(\pi) < c_2(\pi')$ or (ii) $c_1(\pi) < c_1(\pi')$ and $c_2(\pi) \leq c_2(\pi')$. In bi-objective search, a typical task is to find the Pareto-optimal frontier, which consists of all those solution paths that are not dominated by any other solution path.

Bi-objective search is important in many domains, such as transportation and robotics (Bronfman et al. 2015; Bachmann et al. 2018; Fu et al. 2019; Fu, Salzman, and Alterovitz 2021). In such domains, search problems often have two objectives that cannot be optimized at the same time. For example, in the hazardous material transportation problem (Bronfman et al. 2015), we are required to plan a route for transporting hazardous material while considering the travel distance as well as the risk of exposure for residents.

State-of-the-art bi-objective search algorithms, such as BOA* (Hernandez et al. 2020), NAMOA* (Mandow and De La Cruz 2010), and NAMOA*dr (Pulido, Mandow, and Pérez-de-la Cruz 2015), can be used to compute the Pareto-optimal frontier. However, as the number of solution paths in the Pareto-optimal frontier can be exponential in the size of the input graph (Ehrgott 2005; Breugem, Dollevoet, and van den Heuvel 2017), the problem of finding the Pareto-optimal frontier is intrinsically hard. Therefore, exact approaches are often unacceptably slow. To this end, an alternative approach calls for computing an approximation of the Pareto-optimal frontier (Warburton 1987; Breugem, Dollevoet, and van den Heuvel 2017; Tsaggouris and Zaroliagis 2009). Here, we are given a user-specified approximation factor $\varepsilon \geq 0$. A path π *ε-dominates* another path π' if and only if each cost component of π is less than or equal to $(1 + \varepsilon)$ times the respective cost component of π'. An ε-approximate Pareto-optimal frontier is a set of solution paths Π_ε such that any Pareto-optimal solution path is ε-dominated by some path in Π_ε. In structured real-world problems, the ε-approximate Pareto-optimal frontier can have substantially fewer solution paths than the Pareto-optimal frontier even for small ε. Existing work (Goldin and Salzman 2021) has already demonstrated that search algorithms can find an ε-approximate Pareto-optimal frontier for road networks much faster than the Pareto-optimal frontier.

Another dimension used to characterize search algorithms is their *anytime* behavior. In anytime search, we are interested in quickly finding a "good" solution and progressively finding better solutions if time allows (Likhachev, Gordon, and Thrun 2003; van den Berg et al. 2011; Stern et al. 2014; Cohen et al. 2018). Ideally, an anytime search algorithm exhibits the "diminishing returns" property with regard to the quality of the current solution against increasing time, eventually converging to an optimal solution. Such an algorithm is useful when deliberation time is limited or unknown when a search query is provided.

In this paper, we present an anytime approximate bi-objective search algorithm, called Anytime Bi-Objective A*-ε (A-BOA$^*_\varepsilon$). A-BOA$^*_\varepsilon$ has the characteristics of an approximate bi-objective search algorithm as well as an anytime search algorithm. It derives its anytime behavior from progressively tightening the approximation factor.[1] A-BOA$^*_\varepsilon$ starts by quickly finding an initial approximate Pareto-optimal frontier, subsequently finds more solution paths to improve the approximation factor, and eventually finds the entire Pareto-optimal frontier.

Any approximate bi-objective search algorithm can be naively converted into an anytime variant by invoking the search algorithm for progressively smaller values of ε. However, this methodology is inefficient since search effort is duplicated across different values of ε. A-BOA$^*_\varepsilon$ addresses this inefficiency by reusing previous search effort. It does sufficient bookkeeping to allow each invocation of the search algorithm to build on the search effort of the previous invocation. In doing so, it is significantly more efficient than the naive approach. A-BOA$^*_\varepsilon$ also employs a novel pruning technique to further improve its efficiency. This pruning technique is based on a heuristic function designed to estimate a weighted sum of the two costs.

In later sections, we first provide proof sketches for the correctness and convergence properties of A-BOA$^*_\varepsilon$. We then compare it empirically against BOA* and two baseline algorithms that are derived from existing approximate bi-objective search algorithms and do not reuse previous search efforts. Our experimental results show that A-BOA$^*_\varepsilon$ significantly outperforms the baseline algorithms, both in terms of runtime and number of node expansions. In fact, the most advanced variant of A-BOA$^*_\varepsilon$ even slightly outperforms BOA* for finding the Pareto-optimal frontier. Moreover, with a limited amount of time, A-BOA$^*_\varepsilon$ finds solution sets that collectively approximate the Pareto-optimal frontier much better than the solutions found by BOA*.

2 Terminology

In this paper, we use boldface font for 2-tuples (i.e., pairs). We use $v_i, i \in \{1,2\}$, to denote the i-th component of tuple $\mathbf{v}$. The addition of two 2-tuples $\mathbf{v}$ and $\mathbf{v}'$ is defined as $\mathbf{v}+\mathbf{v}' = (v_1+v'_1, v_2+v'_2)$. We define the following types of domination:

1. $\mathbf{v} \preceq \mathbf{v}'$ denotes that $v_1 \le v'_1 \wedge v_2 \le v'_2$. In this case, we say that $\mathbf{v}$ *weakly dominates* $\mathbf{v}'$.
2. $\mathbf{v} \prec \mathbf{v}'$ denotes that $\mathbf{v} \preceq \mathbf{v}' \wedge \mathbf{v} \neq \mathbf{v}'$. In this case, we say that $\mathbf{v}$ *(strictly) dominates* $\mathbf{v}'$.
3. An *approximation factor* ε is a non-negative real number. $\mathbf{v} \preceq_\epsilon \mathbf{v}'$ denotes that $v_1 \le (1+\varepsilon)v'_1 \wedge v_2 \le (1+\varepsilon)v'_2$. In this case, we say that $\mathbf{v}$ *ε-dominates* $\mathbf{v}'$. Note that, when $\varepsilon = 0$, ε-domination is equal to weak domination.

A *(bi-objective) search graph* is a tuple $\langle S, E, \mathbf{c}\rangle$, where S is a finite set of *states* and $E \subseteq S \times S$ is a finite set of *edges*. Cost function $\mathbf{c} : E \to \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ maps an edge to two positive real numbers. We define $c_i : E \to \mathbb{R}_{>0}, i \in \{1,2\}$, as the function that maps an edge e to the i-th component of $\mathbf{c}(e)$. A *path* is a sequence of states $\pi = [s_1, s_2 \dots s_\ell]$ such that $\langle s_j, s_{j+1}\rangle \in E$ for all $j \in \{1,2\dots\ell-1\}$. The cost of path π is $\mathbf{c}(\pi) = \sum_{j=1}^{\ell-1} \mathbf{c}(\langle s_j, s_{j+1}\rangle)$. We use $s(\pi)$ to denote the *ending state* of π, i.e., s_ℓ. We use $succ(s) = \{s' \in S | \langle s, s'\rangle \in E\}$ to denote the successors of state s. By *extending* π with an edge $\langle s_\ell, s_{\ell+1}\rangle$, we obtain a new path $[s_1, s_2 \dots s_\ell, s_{\ell+1}]$. We say that a path π' *extends* another path π if and only if π' can be obtained by applying a sequence of extend operations on π.

A *(bi-objective search) problem instance* is a tuple $P = \langle S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}}\rangle$, where $\langle S, E, \mathbf{c}\rangle$ is a search graph, $s_{\text{start}} \in S$ is the *start state*, and $s_{\text{goal}} \in S$ is the *goal state*. A path is a *solution* for problem instance P if and only if it is a path from s_{start} to s_{goal}. For problem instance P, a *heuristic function* $\mathbf{h} : S \to \mathbb{R}_{\ge 0} \times \mathbb{R}_{\ge 0}$ is an estimation of the cost of a path from a given state to s_{goal}. $\mathbf{h}$ is consistent if and only if $\mathbf{h}(s_{\text{goal}}) = \mathbf{0}$ and $\mathbf{h}(s) \preceq \mathbf{c}(\langle s, s'\rangle) + \mathbf{h}(s')$. In this paper, we limit our discussion to consistent heuristic functions. For any path π from s_{start} to some ending state $s(\pi)$, its $\mathbf{g}$-value is defined as $\mathbf{c}(\pi)$, and its $\mathbf{f}$-value is defined as $\mathbf{f}(\pi) = \mathbf{g}(\pi) + \mathbf{h}(s(\pi))$.

Let π and π' be two paths from state s_{start}. We say that π *dominates (resp. weakly dominates and ε-dominates)* π' if and only if $\mathbf{f}(\pi) \prec \mathbf{f}(\pi')$ (resp. $\mathbf{f}(\pi) \preceq \mathbf{f}(\pi')$ and $\mathbf{f}(\pi) \preceq_\epsilon \mathbf{f}(\pi')$). For a problem instance P, a *Pareto-optimal solution* is a solution that is not dominated by any other solution of P. The *Pareto-optimal frontier* Π^* is the set of all Pareto-optimal solutions, and a *cost-unique Pareto-optimal frontier* is a maximal subset of the Pareto-optimal frontier such that no two solutions in it have the same cost. Given approximation factor ε, an *ε-approximate Pareto-optimal frontier* Π_ε is a subset of the Pareto-optimal frontier such that , for any Pareto-optimal solution π of P, there exists a solution $\pi' \in \Pi_\varepsilon$ with $\pi' \preceq_\epsilon \pi$. Note that any cost-unique Pareto-optimal frontier is also an ε-approximate Pareto-optimal frontier for $\varepsilon = 0$.

We define the domination factor of a path π' over another path π as

$$\text{DF}(\pi', \pi) = \max\left\{\frac{f_1(\pi')}{f_1(\pi)} - 1, \frac{f_2(\pi')}{f_2(\pi)} - 1, 0\right\},$$

which measures how "good" $\mathbf{f}(\pi')$ approximates $\mathbf{f}(\pi)$. It is easy to verify that $\mathbf{f}(\pi') \preceq_\epsilon \mathbf{f}(\pi)$ if and only if $\varepsilon \ge \text{DF}(\pi', \pi)$. For a set of solutions Π, we define its *approximation factor* as

$$\varepsilon(\Pi) = \max_{\pi \in \Pi^*}\left\{\min_{\pi' \in \Pi} \text{DF}(\pi', \pi)\right\}. \quad (1)$$

We slightly abuse the ε notation and use it as a function here. Roughly speaking, for each Pareto-optimal solution π, we find a path π' in Π that approximately dominates π the best, compute the value of the domination factor, and then take the maximum of these values over all Pareto-optimal solutions. Π is an ε-approximate Pareto-optimal solution set if and only if $\varepsilon \ge \varepsilon(\Pi)$.

Example 1. *Fig. 1 shows a Pareto-optimal frontier with four solutions, denoted as* Π^*. *Let* $\Pi = \{\pi_1, \pi_2, \pi_4\}$ *be a subset*

[1] Therefore, ε is controlled by the algorithm and not user-specified.

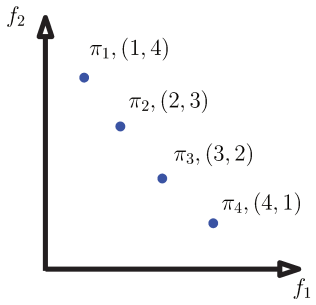


Figure 1: The $\mathbf{f}$-values of a Pareto-optimal frontier which consists of four solutions π_1, π_2, π_3, and π_4. The pair of numbers next to each path is its $\mathbf{f}$-value.

of Π^*. *For any solution* $\pi \in \Pi$, $\min_{\pi' \in \Pi} \mathrm{DF}(\pi', \pi) = 0$ *because* $\mathrm{DF}(\pi', \pi) = 0$ *by definition for* $\pi' = \pi$. *For solution* π_3, $\mathrm{DF}(\pi', \pi_3)$ *is minimized for* $\pi' = \pi_4$, *and we have* $\mathrm{DF}(\pi_4, \pi_3) = \frac{1}{3}$. *Therefore,* Π *is a* $\frac{1}{3}$*-approximate Pareto-optimal frontier.*

3 BOA* and BOA*$_\varepsilon$

BOA* (Hernandez et al. 2020) is a best-first bi-objective search algorithm. It maintains an OPEN list, containing the frontier of the search tree (i.e., the generated but not yet expanded nodes) and a set of Pareto-optimal solutions $sols$ that it has found so far. In BOA*, a node represents a path π from s_{start} to some state. In this paper, we therefore use "node" and "path" interchangeably. For each state s, BOA* uses $g_2^{\min}(s)$ to store the minimum g_2-value of all expanded paths that end at state s. Alg. 1 shows the pseudocode of BOA*. In the beginning, OPEN contains only one path $[s_{\text{start}}]$. At each iteration, BOA* extracts a path π with the lexicographically smallest $\mathbf{f}$-value from OPEN. The path π is pruned if there exists (i) an expanded path π' with the same ending state as π, i.e., $s(\pi) = s(\pi')$, and $\mathbf{g}(\pi') \preceq \mathbf{g}(\pi)$ or (ii) an expanded path π' with $s(\pi') = s_{\text{goal}}$ and $\mathbf{f}(\pi') \preceq \mathbf{f}(\pi)$. Hernandez et al. (2020) showed that testing (i) and (ii) can be done by testing if

$$g_2(\pi) \geq g_2^{\min}(s(\pi))$$

and

$$f_2(\pi) \geq g_2^{\min}(s_{\text{goal}}), \tag{2}$$

respectively. If $s(\pi) = s_{\text{goal}}$, BOA* then adds π to $sols$. Otherwise, BOA* expands π and generates a child path for each successor s' of $s(\pi)$. The child path is the path that extends π with edge $\langle s, s' \rangle$.

Goldin and Salzman (2021) showed that, by a slight modification, BOA* can be made to find an approximate Pareto-optimal frontier. To do so, we simply replace Eq. 2 with

$$(1 + \varepsilon) f_2(\pi) \geq g_2^{\min}(s_{\text{goal}}),$$

which means that path π is pruned if it is ε-dominated by a found solution. The resulting algorithm is called BOA*$_\varepsilon$. Lines 7 and 15 of Alg. 1 need to be changed accordingly for converting BOA* to BOA*$_\varepsilon$.

Algorithm 1: BOA*

Input : A problem instance $\langle S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}} \rangle$
Output: A cost-unique Pareto-optimal frontier

1 $sols \leftarrow \emptyset$
2 OPEN $\leftarrow \{[s_{\text{start}}]\}$
3 **for each** $s \in S$ **do**
4 $\quad g_2^{\min}(s) \leftarrow \infty$
5 **while** OPEN $\neq \emptyset$ **do**
6 $\quad \pi \leftarrow$ OPEN.extract_min()
7 $\quad$ **if** $g_2(\pi) \geq g_2^{\min}(s(\pi)) \vee f_2(\pi) \geq g_2^{\min}(s_{goal})$ **then**
8 $\quad\quad$ **continue**
9 $\quad g_2^{\min}(s(\pi)) \leftarrow g_2(\pi)$
10 $\quad$ **if** $s(\pi) = s_{goal}$ **then**
11 $\quad\quad$ add π to $sols$
12 $\quad\quad$ **continue**
13 $\quad$ **for each** $s' \in succ(s(\pi))$ **do**
14 $\quad\quad \pi' \leftarrow$ extend$(\pi, \langle s(\pi), s' \rangle)$
15 $\quad\quad$ **if** $g_2(\pi') \geq g_2^{\min}(s') \vee f_2(\pi') \geq g_2^{\min}(s_{goal})$ **then**
16 $\quad\quad\quad$ **continue**
17 $\quad\quad$ add π' to OPEN
18 **return** $sols$

4 A-BOA*$_\varepsilon$

Conceptually, BOA* can be viewed as an anytime algorithm. It can be stopped anytime and then return $sols$ as the set of solutions that it has found so far. However, BOA* finds Pareto-optimal solutions in lexicographically increasing order of their $\mathbf{f}$-values, and the solutions that it finds first are the ones with small c_1 and thus large c_2. Therefore, if stopped early, BOA* might return a solution set with a large approximation factor (Eq. 1). On the other hand, we can run BOA*$_\varepsilon$ multiple times with a sequence of decreasing ε-values until $\varepsilon = 0$, and this process of repeatedly running BOA*$_\varepsilon$ can be viewed as a basic anytime bi-objective search algorithm. However, rerunning BOA*$_\varepsilon$ from scratch is not efficient since it repeats the work of the previous iteration.

This motivates us to propose our new anytime bi-objective search algorithm, called Anytime BOA*$_\varepsilon$ (A-BOA*$_\varepsilon$). As we will see shortly, A-BOA*$_\varepsilon$ makes use of *intervals* to keep track of its progress in the previous iteration and avoid repeating work. We start by providing the definition and properties of intervals before we describe A-BOA*$_\varepsilon$.

4.1 Preliminaries: Intervals

Definition 1 (Interval). *Let* π_{tl} *and* π_{br} *be two Pareto-optimal solutions such that* $f_1(\pi_{tl}) < f_1(\pi_{br})$ *(and hence* $f_2(\pi_{tl}) > f_2(\pi_{br})$*). Furthermore, let* Π *be a set of paths from* s_{start} *(not necessarily ending at* s_{goal}*) such that,* $\forall \pi \in \Pi$ $f_1(\pi_{tl}) \leq f_1(\pi)$ *and* $f_2(\pi_{br}) \leq f_2(\pi)$. *We define its corresponding interval as the 3-tuple* $I = \langle \pi_{tl}, \pi_{br}, \Pi \rangle$ *and refer to* π_{tl}, π_{br}, *and* Π *as the top-left solution, the bottom-right solution, and the to-expand paths of* I, *respectively.*

The notion of an interval is visualized in Fig. 2. Roughly speaking, Π is a set of paths whose $\mathbf{f}$-values fall "between" the $\mathbf{f}$-values of π_{tl} and π_{br}.

Definition 2 (Approximation Factor of an Interval). *Given an interval* $I = \langle \pi_{tl}, \pi_{br}, \Pi \rangle$, *we define its approxi-*

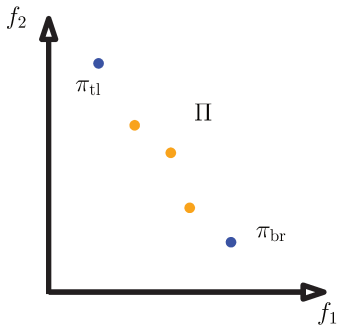


Figure 2: An example interval $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ in A-BOA$^*_\varepsilon$. The two blue dots represent the $\mathbf{f}$-values of Pareto-optimal solutions π_{tl} and π_{br}. The orange dots represent the $\mathbf{f}$-values of all paths in Π.

Algorithm 2: Anytime BOA$^*_\varepsilon$

Input : A problem instance $\langle S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}} \rangle$ and parameter d for decreasing approximation factor
Output: An approximate Pareto-optimal frontier

1 $\pi_{\text{tl}} \leftarrow$ a Pareto-optimal solution with the lexicographically smallest (c_1, c_2)
2 $\pi_{\text{br}} \leftarrow$ a Pareto-optimal solution with the lexicographically smallest (c_2, c_1)
3 $I_s \leftarrow \langle \pi_{\text{tl}}, \pi_{\text{br}}, \{[s_{\text{start}}]\} \rangle$
4 ILIST $\leftarrow \{I_s\}$
5 **while** $\exists I \in$ ILIST*:* $\hat{\varepsilon}(I) \neq 0$ **do**
6 $\quad I \leftarrow \text{argmax}_{I \in \text{ILIST}} \hat{\varepsilon}(I)$
7 $\quad$ ILIST$' \leftarrow$ Search$(I, \hat{\varepsilon}(I)/d)$
8 $\quad$ remove I from ILIST
9 $\quad$ add all intervals of ILIST$'$ to ILIST
10 **return** *ConstructSols(I)*

mation factor $\hat{\varepsilon}(I)$ *as*

$$\hat{\varepsilon}(I) = \max_{\pi \in \Pi} \{\min\left(DF(\pi_{tl}, \pi), DF(\pi_{br}, \pi)\right)\}$$

if $\Pi \neq \emptyset$*, and* $\hat{\varepsilon}(I) = 0$ *otherwise.*

Note that, for any path $\pi \in \Pi$, π is $\hat{\varepsilon}(I)$-dominated by either π_{tl} or π_{br} by definition.

As we will see shortly, A-BOA$^*_\varepsilon$ maintains a set of *intervals* ILIST. For each interval $\langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ in ILIST, Π contains all generated but not yet expanded paths that have the potential to extend to a Pareto-optimal solution π_{sol} with $f_1(\pi_{\text{tl}}) \leq f_1(\pi_{\text{sol}})$ and $f_2(\pi_{\text{br}}) \leq f_2(\pi_{\text{sol}})$. Although A-BOA$^*_\varepsilon$ does not explicitly maintain OPEN, the to-expand paths of the intervals in ILIST can be conceptually viewed as paths in OPEN, grouped by their $\mathbf{f}$-values.

4.2 Algorithmic Framework

Alg. 2 shows the pseudocode of A-BOA$^*_\varepsilon$. The algorithm initializes ILIST with only one interval $I_s = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \{[s_{\text{start}}]\} \rangle$, where π_{tl} and π_{br} are the Pareto-optimal solutions with the lexicographically smallest (c_1, c_2)- and (c_2, c_1)-values, respectively (Line 4). Such solutions can be found quickly using any single-objective search algorithm, such as A* (Hart, Nilsson, and Raphael 1968). A-BOA$^*_\varepsilon$ then repeatedly picks the interval in ILIST with the largest approximation factor and invokes Alg. 3 to continue the search on its to-expand paths.

Algorithm 3: Search

Input : An interval $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi \rangle$ and an approximation factor ε
Output: A list of intervals, each with an approximation factor not larger than ε

1 OPEN $\leftarrow \Pi$, $\Pi_{\text{tmp}} \leftarrow \emptyset$, $\pi_{\text{tmp}} \leftarrow \pi_{\text{tl}}$
2 **for each** $s \in S$ **do**
3 $\quad g_2^{\min}(s) \leftarrow \infty$
4 **while** OPEN $\neq \emptyset$ **do**
5 $\quad \pi \leftarrow$ OPEN.extract_min()
6 $\quad$ **if** `is_dominated`(π) **then**
7 $\quad\quad$ **continue**
8 $\quad g_2^{\min}(s(\pi)) \leftarrow g_2(\pi)$
9 $\quad$ **if** $s(\pi) = s_{goal}$ **then**
10 $\quad\quad$ add $\langle \pi_{\text{tmp}}, \pi, \Pi_{\text{tmp}} \rangle$ to ILIST
11 $\quad\quad$ $\pi_{\text{tmp}} \leftarrow \pi$, $\Pi_{\text{tmp}} \leftarrow \emptyset$
12 $\quad\quad$ **continue**
13 $\quad$ **for each** $s' \in succ(s(\pi))$ **do**
14 $\quad\quad$ $\pi' \leftarrow$ extend$(\pi, \langle s(\pi), s' \rangle)$
15 $\quad\quad$ **if** `is_dominated`(π') **then**
16 $\quad\quad\quad$ **continue**
17 $\quad\quad$ add π' to OPEN
18 add $\langle \pi_{\text{tmp}}, \pi_{\text{br}}, \Pi_{\text{tmp}} \rangle$ to ILIST
19 **return** ILIST

20 **Define** `is_dominated`(π)**:**
21 $\quad$ **if** $f_1(\pi) \geq f_1(\pi_{br}) \vee f_2(\pi) \geq f_2(\pi_{tl}) \vee$ `can_prune_wsh`(π) **then**
22 $\quad\quad$ **return** True
23 $\quad$ **if** $g_2(\pi) \geq g_2^{\min}(s(\pi)) \vee (1+\varepsilon) f_2(\pi) \geq g_2^{\min}(s_{goal})$ **then**
24 $\quad\quad$ **if** $g_2(\pi) < g_2^{\min}(s(\pi)) \wedge f_2(\pi) < g_2^{\min}(s_{goal})$ **then**
25 $\quad\quad\quad$ add π to Π_{tmp}
26 $\quad\quad$ **return** True
27 $\quad$ **return** False

Alg. 3 takes an interval I and an approximation factor ε as inputs and returns a set of new intervals, each with an approximation factor not larger than ε. In A-BOA$^*_\varepsilon$, the input ε for Alg. 3 is set to $\hat{\varepsilon}(I)/d$ (Line 7), where d is a parameter that specifies how fast the approximation factor decreases in A-BOA$^*_\varepsilon$. Other strategies for decreasing ε could be easily implemented for A-BOA$^*_\varepsilon$ but are out of scope for this work. A-BOA$^*_\varepsilon$ terminates when all intervals in ILIST have an approximation factor of zero. As we will show in Sec. 6, this means that A-BOA$^*_\varepsilon$ has found a cost-unique Pareto-optimal frontier. At any point in time, the Pareto-optimal solutions that A-BOA$^*_\varepsilon$ has found so far can be constructed as the set of top-left and bottom-right solutions of all intervals in ILIST. This set is a $\max\{\hat{\varepsilon}(I) | I \in \text{ILIST}\}$-approximate Pareto-optimal frontier.

Alg. 3 is similar to BOA$^*_\varepsilon$ with differences only in initialization and dominance checking. Alg. 3 initializes OPEN with the to-expand paths of the input interval so that it does not start from scratch. Dominance checking is encapsulated in Function `is_dominated` (Lines 20-27). Aside from ap-

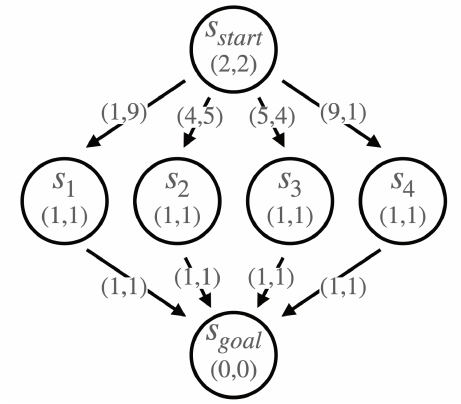


Figure 3: An example problem instance whose Pareto-optimal frontier consists of four paths. The pair of numbers inside each state is its $\mathbf{h}$-value. The pair of numbers annotating each edge is its cost.

plying the pruning conditions of A-BOA$^*_\varepsilon$ on Lines 23-26, Alg. 3 also prunes a path if it is dominated by either the top-left solution or the bottom-right solution of the input interval (Lines 21-22). Function `can_prune_wsh` on Line 21 encapsulates another pruning technique that we will explain in Sec. 5. Alg. 3 also stores those paths that are ε-dominated by a solution that has been found but do not satisfy the pruning conditions of BOA* (Lines 24-25). Such paths are safe to prune for the input ε-value but still have the potential to extend to Pareto-optimal solutions. A-BOA$^*_\varepsilon$ temporally stores these paths in Π_{tmp}. Each time a solution is found, a new interval is created with the previously found solution π_{tmp} as its top-left solution, the newly found solution as its bottom-right solution, and Π_{tmp} as its to-expand paths, π_{tmp} is then set to the newly found solution and Π_{tmp} is emptied. When OPEN becomes empty, Alg. 3 creates a new interval with the previously found solution π_{tmp} as its top-left solution, π_{br} as its bottom-right solution, and Π_{tmp} as its to-expand paths and returns the list of all new intervals. Each of these intervals has an approximation factor not larger than the input ε-value because all of its to-expand paths are ε-dominated by its top-left solution.

Example 2. *Fig. 3 shows a problem instance whose Pareto-optimal frontier consists of four solutions. Assume that $d = 4$. In the beginning, A-BOA$^*_\varepsilon$ finds two Pareto-optimal solutions $\pi_1 = [s_{start}, s_1, s_{goal}]$ and $\pi_4 = [s_{start}, s_4, s_{goal}]$ and initializes* ILIST *with interval $I_s = \langle \pi_1, \pi_4, \{[s_{start}]\}\rangle$. The $\mathbf{f}$-values for π_1, π_4, and path $[s_{start}]$ are $(2, 10)$, $(10, 2)$, and $(2, 2)$, respectively. The approximation factor for I_s is $\hat{\varepsilon}(I_s) = \min(DF(\pi_1, [s_{start}]), DF(\pi_4, [s_{start}])) = 4$.*

In the first iteration, A-BOA$^*_\varepsilon$ invokes Alg. 3 with I_s and $\varepsilon = 1$. Alg. 3 expands path $[s_{start}]$ and generates paths $[s_{start}, s_1]$, $[s_{start}, s_2]$, $[s_{start}, s_3]$, and $[s_{start}, s_4]$. Paths $[s_{start}, s_1]$ and $[s_{start}, s_4]$ are pruned on Lines 21-22 because they are weakly dominated by π_1 and π_4, respectively. A-BOA$^*_\varepsilon$ then expands $[s_{start}, s_2]$ and $\pi_2 = [s_{start}, s_2, s_{goal}]$ in sequence and finds a new solution π_2. $g_2^{min}(s_{goal})$ is updated to $g_2(\pi_2) = 6$. Path $[s_{start}, s_3]$ is pruned and put into Π_{tmp}. Alg. 3 returns the two intervals $I_1 = \langle \pi_1, \pi_2, \emptyset\rangle$ and $I_2 = \langle \pi_2, \pi_4, \{[s_{start}, s_3]\}\rangle$. Thus, up until this point,, A-BOA$^*_\varepsilon$ has found the three Pareto-optimal solutions π_1, π_2, and π_4.

We have $\hat{\varepsilon}(I_1) = 0$ by definition and $\hat{\varepsilon}(I_2) = \min(DF(\pi_2, [s_{start}, s_3]), DF(\pi_4, [s_{start}, s_3])) = 0.2$. In the second iteration, A-BOA$^*_\varepsilon$ invokes Alg. 3 with I_2 and $\varepsilon = 0.05$. Eventually, Alg. 3 finds solution $\pi_3 = [s_{start}, s_3, s_{goal}]$ and returns two intervals $\langle \pi_2, \pi_3, \emptyset\rangle$ and $\langle \pi_3, \pi_4, \emptyset\rangle$. A-BOA$^*_\varepsilon$ then terminates with all four Pareto-optimal solutions found.

5 Weighted-Sum Heuristic Pruning

In this section, we describe a novel pruning technique for A-BOA$^*_\varepsilon$, called weighted-sum heuristic pruning, that allows it to discard paths that cannot be extended to Pareto-optimal solutions. Many bi-objective search algorithms (Hernandez et al. 2020; Goldin and Salzman 2021) use a heuristic function to estimate the cost from a given state to the goal state for each of the two objectives individually. Such a heuristic function can be computed quickly via a single-objective search, such as Dijkstra's algorithm, from the goal state. This motivates us to design other heuristic functions that can be computed via single-objective searches to further aid the bi-objective search, even though they do not use heuristic functions to guide the search but to prune paths.

Let $w > 0$ be a user-defined real-valued parameter. The *weighted-sum heuristic* function $h_w : S \to \mathbb{R}$ estimates the weighted sum of the cost $c_1 + w \cdot c_2$ from a given state to the goal state. This heuristic function can be computed quickly via a single-objective search from the goal state using $c_1(e) + w \cdot c_2(e)$ as the cost of each edge e. This way, the exact minimum weighted-sum values are obtained for all states and used as an admissible (that is, non-overestimating) heuristic function. In this paper, we limit our discussion to admissible weighted-sum heuristic functions.

Property 1. *Let π be some path, and let π' and π'' be two Pareto-optimal solutions that have been found by A-BOA$^*_\varepsilon$ and satisfy (i) $f_1(\pi') \le f_1(\pi)$, (ii) $f_2(\pi'') \le f_2(\pi)$, and (iii) $f_1(\pi') < f_1(\pi'')$. If*

$$g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \ge f_1(\pi'') + w \cdot f_2(\pi'), \quad (3)$$

then any solution extending π is weakly dominated by π' or π''.

Proof. Assume π, π', and π'' satisfy Eqs. (i), (ii), (iii), and 3. Let π_{sol} be any solution that extends π. Because the heuristic function $\mathbf{h}$ is consistent, we have $f_1(\pi_{\text{sol}}) \ge f_1(\pi) \ge f_1(\pi')$ and $f_2(\pi_{\text{sol}}) \ge f_2(\pi) \ge f_2(\pi'')$. Assuming that π_{sol} is not weakly dominated by π' or π'', we have

$$f_2(\pi_{\text{sol}}) < f_2(\pi') \text{ and } f_1(\pi_{\text{sol}}) < f_1(\pi'') \quad (4)$$

because π_{sol} is not worse than or equal to π' or π'' in both objectives. Given that h_w is admissible and Eq. 3 is satisfied, we have

$$\begin{aligned} f_1(\pi_{\text{sol}}) + w \cdot f_2(\pi_{\text{sol}}) &= c_1(\pi_{\text{sol}}) + w \cdot c_2(\pi_{\text{sol}}) \\ &\ge g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \\ &\ge f_1(\pi'') + w \cdot f_2(\pi'). \end{aligned} \quad (5)$$

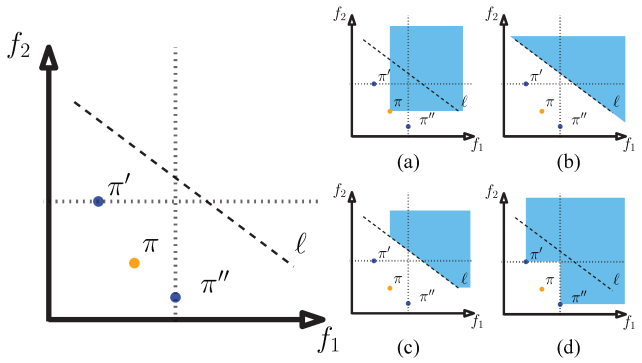


Figure 4: An example of weighted-sum heuristic pruning. π is the path that we consider for pruning. π' and π'' are two known solutions. The $\mathbf{f}$-value of any path that extends π must lie in the shaded area of (a) because the heuristic function $\mathbf{h}$ is consistent. The shaded area in (b) represents $\mathbf{f}$-values (f_1, f_2) that satisfy $f_1 + w \cdot f_2 \geq g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi))$ for a given weighted-sum heuristic function h_w. The $\mathbf{f}$-value of any solution that extends π must lie in this area since h_w is admissible. The shaded area in (c) is the intersection of the shaded areas in (a) and (b). The shaded area in (d) represents the $\mathbf{f}$-values that are weakly dominated by the $\mathbf{f}$-value of π' or π''. The shaded area in (c) is contained in the shaded area in (d), which means that any solution that extends π is weakly dominated by π' or π'', and hence π is safe to prune.

However, given Eq. 4, we have

$$f_1(\pi_{\text{sol}}) + w \cdot f_2(\pi_{\text{sol}}) < f_1(\pi'') + w \cdot f_2(\pi'),$$

which contradicts Eq. 5

□

Fig. 4 shows a visualization of the proof of Property 1. Here, π, π', and π'' are paths as defined in Property 1. With only information about h_1 and h_2, the shaded area in Fig. 4a shows where the $\mathbf{f}$-value of a solution extending π can possibly be. π still seems to have the potential to extend to a solution that is not weakly dominated by π' or π'', that is, whose $\mathbf{f}$-value lies outside of the shaded area in Fig. 4d. However, with a weighted-sum heuristic function, we show that π is safe to prune.

Alg. 3 can incorporate this pruning technique by using π_{tmp} and π_{br} as π' and π'', respectively. A path π can be pruned if, given weighted-sum heuristic function h_w,

$$g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi_{\text{br}}) + w \cdot f_2(\pi_{\text{tmp}}).$$

Function `can_prune_wsh` on Line 21 of Alg. 3 returns true iff this pruning condition holds. Since BOA* expands paths in lexicographic order of their $\mathbf{f}$-values, it is not straightforward to see which path can be used as π'' for weighted-sum heuristic pruning. One might consider using a solution with the lexicographically smallest (c_2, c_1)-value as π''. However, by doing this, weighted-sum heuristic pruning only happens near the end of the search, and our preliminary results show that the total runtime of BOA* usually become worse due to the overhead of computing the weighted-sum heuristic function.

6 Theoretical Results

In this section, we provide theoretical results about A-BOA$^*_\varepsilon$. After describing properties of ILIST and intervals, we show that (i) the set of solutions that A-BOA$^*_\varepsilon$ has found at any point of time is a $\max\{\hat{\varepsilon}(I) | I \in \text{ILIST}\}$-approximate Pareto-optimal frontier (Thm. 1) and that (ii) A-BOA$^*_\varepsilon$ eventually finds a cost-unique Pareto-optimal frontier (Thm. 2).

Property 2. *Let $[I_1, I_2 \dots I_M]$ be the sequence of intervals obtained when sorting* ILIST *in ascending order according to the f_1-value of the top-left solution of each interval. For any $i \in \{1, 2 \dots M-1\}$, the bottom-right solution of I_i is also the top-left solution of I_{i+1}.*

Proof. The property is trivially true in the beginning when ILIST is initialized on Line 4 of Alg. 2. Assume that the property holds in the beginning of an iteration of Alg. 2. An interval I is then chosen and sent to Alg. 3. Alg. 3 creates intervals in ascending order of the f_1-values of their top-left solutions. When an interval is created, its bottom-right solution is stored in π_{tmp} and becomes the top-left solution of the interval that is created next. Moreover, the first and last created intervals have the same top-left and bottom-right solutions as I, respectively. Therefore, after replacing I in ILIST with the list of intervals returned by Alg. 3, Property 2 still holds. □

Property 3. *Consider the sequence of intervals $[I_1, I_2 \dots I_N]$ in Property 2, and let $\Pi = [\pi_1, \pi_2 \dots \pi_{N+1}]$ be the sequence of solutions such that, for every $i = 1 \dots N$, π_i and π_{i+1} are the top-left and bottom-right solutions of I_i, respectively. The solutions in Π have strictly increasing f_1-values and strictly decreasing f_2-values.*

Proof sketch. This property holds because, according to Definition 1, the top-left solution has a smaller f_1-value and a larger f_2-value than the bottom-right solution. We skip the details due to space limit. □

Lemma 1. *When A-BOA$^*_\varepsilon$ prunes a path in Alg. 3 and this prevents it in the future from adding a solution π_{sol} (that extends π) to the solution set, then A-BOA$^*_\varepsilon$ has already found or will find in the future a solution that weakly dominates π_{sol}.*

Proof. There are three cases when A-BOA$^*_\varepsilon$ prunes a path π:

1. It prunes path π on Line 21 of Alg. 3 because $f_1(\pi) \geq f_1(\pi_{\text{br}}) \vee f_2(\pi) \geq f_2(\pi_{\text{tl}})$. Then, A-BOA$^*_\varepsilon$ has already found a solution, namely π_{tl} or π_{br}, that weakly dominates π (and hence also π_{sol}). This solution will eventually be included in the solution set.
2. It prunes path π on Line 21 of Alg. 3 because can_prune_wsh(π) holds. Then, according to Property 1, π_{tmp} or π_{br} weakly dominates π_{sol}. Both π_{tmp} or π_{br} are solutions and will eventually be included in the solution set.

3. It prunes path π on Line 26 of Alg. 3 after storing this path in Π_{tmp}. Then, the pruned path is stored as a to-expand path for an interval and extracted again in the future for expansion.
4. It prunes path π on Line 26 of Alg. 3 without storing it in Π_{tmp}. Then, path π satisfies $g_2(\pi) \geq g_2^{min}(s(\pi)) \vee f_2(\pi) \geq g_2^{min}(s_{\text{goal}})$. This is the pruning condition of BOA*, and thus the proof of Lemma 7 of Hernandez et al. (2020) applies.

□

Lemma 2. *In A-BOA$^*_\varepsilon$, the top-left and bottom-right solutions of any interval are Pareto-optimal solutions.*

Proof. Assume that a solution π_{sol} is the top-left or bottom-right solution of an interval in ILIST but is dominated by a solution π'. From Lemma 1, A-BOA$^*_\varepsilon$ eventually finds a solution π'' that weakly dominates π' (and hence dominates π_{sol}) to the solution set. Then, both π_{sol} and π'' are the top-left or bottom-right solutions of some intervals in ILIST. This contradicts Property 3 because π_{sol} and π'' cannot be part of a sequence of solutions with increasing f_1-values and decreasing f_2-values at the same time. □

The following two theorems show that A-BOA$^*_\varepsilon$ finds approximate Pareto-optimal frontiers with approximation factor guarantees and eventually finds a cost-unique Pareto-optimal frontier.

Theorem 1. *The set of solutions that A-BOA$^*_\varepsilon$ finds after each iteration (namely, the set of top-left and bottom-right solutions of all intervals in* ILIST*) is a* $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$*-approximate Pareto-optimal frontier.*

Proof. Let Π denote the set of solutions that A-BOA$^*_\varepsilon$ finds after an iteration and Π^* denote the Pareto-optimal frontier. From Lemma 1, for any $\pi \in \Pi^* \setminus \Pi$, there exists a to-expand path $\hat{\pi}$ that can be extended to a solution π_{sol} that weakly dominates π. Because A-BOA$^*_\varepsilon$ uses consistent heuristic functions, we have $\mathbf{f}(\hat{\pi}) \preceq \mathbf{f}(\pi_{\text{sol}}) \preceq \mathbf{f}(\pi)$, and hence $\text{DF}(\pi', \pi) \leq \text{DF}(\pi', \hat{\pi})$ for any path π'. Let $I = \langle \pi_{\text{tl}}, \pi_{\text{br}}, \Pi_I \rangle$ denote the interval with $\hat{\pi} \in \Pi_I$. From Definition 1, we have $f_1(\pi_{\text{tl}}) \leq f_1(\hat{\pi}) \leq f_1(\pi_{\text{br}})$ and $f_2(\pi_{\text{br}}) \leq f_2(\hat{\pi}) \leq f_2(\pi_{\text{tl}})$ and hence

$$\text{DF}(\pi_{\text{tl}}, \hat{\pi}) = \frac{f_2(\pi_{\text{tl}})}{f_2(\hat{\pi})} - 1 \text{ and } \text{DF}(\pi_{\text{br}}, \hat{\pi}) = \frac{f_1(\pi_{\text{br}})}{f_1(\hat{\pi})} - 1.$$

From Property 3, for any solution $\pi' \in \Pi \setminus \{\pi_{\text{tl}}, \pi_{\text{br}}\}$, we either have (1) $f_1(\pi') < f_1(\pi_{\text{tl}})$ and $f_2(\pi') > f_2(\pi_{\text{tl}})$, which implies that $\text{DF}(\pi', \hat{\pi}) = \frac{f_2(\pi')}{f_2(\hat{\pi})} - 1 > \text{DF}(\pi_{\text{tl}}, \hat{\pi})$, or (2) $f_1(\pi') > f_1(\pi_{\text{br}})$ and $f_2(\pi') < f_2(\pi_{\text{br}})$, which implies that $\text{DF}(\pi', \hat{\pi}) = \frac{f_1(\pi')}{f_1(\hat{\pi})} - 1 > \text{DF}(\pi_{\text{br}}, \hat{\pi})$. Therefore, we have $\min_{\pi' \in \Pi}\{\text{DF}(\pi', \hat{\pi})\} = \min(\text{DF}(\pi_{\text{tl}}, \hat{\pi}), \text{DF}(\pi_{\text{br}}, \hat{\pi}))$. From Definition 2, we have $\min(\text{DF}(\pi_{\text{tl}}, \hat{\pi}), \text{DF}(\pi_{\text{br}}, \hat{\pi})) \leq \hat{\varepsilon}(I)$ because $\hat{\pi} \in \Pi_I$. To summarize, $\min_{\pi' \in \Pi}\{\text{DF}(\pi', \pi)\}$ is upper-bounded by the approximation factor of some interval in ILIST. Therefore, $\varepsilon(\Pi)$ is upper-bounded by $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$. □

Theorem 2. *A-BOA$^*_\varepsilon$ eventually finds a cost-unique Pareto-optimal frontier.*

Proof. In each iteration, Alg. 3 invokes Alg. 3 with an interval I and an ε-value that is strictly smaller than the approximation factor of I. All intervals that are outputted by Alg. 3 have approximation factors that are not larger than the input ε-value because, for each one of them, the top-left solution ε-dominates every path in set of to-expand paths. From Thm. 1, the solution set found by A-BOA$^*_\varepsilon$ is a $\max\{\hat{\varepsilon}(I)|I \in \text{ILIST}\}$-approximate Pareto-optimal frontier. Therefore, A-BOA$^*_\varepsilon$ progressively reduces the approximation factor of the solution set by replacing an interval with intervals that have smaller approximation factors. Since the graph is finite, the number of distinctive costs of all paths that are not dominated by any Pareto-optimal solution is finite. Eventually, the approximation factor of the solution set is so small that none of the found solutions ε-dominates any such path. No path is stored as a to-expand path, and hence all intervals in ILIST have an approximation factor of zero and Alg. 2 terminates. □

7 Experimental Results

In this section, we evaluate A-BOA$^*_\varepsilon$ experimentally by comparing it with the following algorithms:

1. BOA*.
2. Basic-A-BOA$^*_\varepsilon$: Basic-A-BOA$^*_\varepsilon$ iteratively invokes BOA$^*_\varepsilon$, each time with the input approximation factor divided by a constant d. Basic-A-BOA$^*_\varepsilon$ is similar to A-BOA$^*_\varepsilon$ except that it does not reuse previous search effort.
3. Iterative-PP-A*: Iterative-PP-A* is based on PP-A* (Goldin and Salzman 2021), a state-of-the art approximate bi-objective search algorithm. Given problem instance P and approximation factor ε, PP-A* finds a set of solutions Π such that, for any solution π of P, there exists a solution $\pi' \in \Pi$ that ε-dominates π, although π' is not necessarily a Pareto-optimal solution.[2] PP-A* uses more complicated data structures than BOA*, and hence we do not know how to reuse the search effort of PP-A*. In our experiments, Iterative-PP-A* iteratively invokes PP-A* with a decreasing sequence of ε-values, namely $0.01, 0.001, 0.0001$, and 0.

We evaluate two variants of A-BOA$^*_\varepsilon$, one with the weighted-sum heuristic pruning technique (denoted as A-BOA$^*_\varepsilon$-w) and one without this optimization (denoted as A-BOA$^*_\varepsilon$). All algorithms were implemented in C++ and shared the code base as much as possible.[3]

We use three road maps from the 9th DIMACS Implementation Challenge[4], namely BAY (321,270 states, 794,830 edges), FLA (1,070,376 states, 2,712,798 edges), and NE (1,524,453 states, 3,897,636 edges). For each road map, we

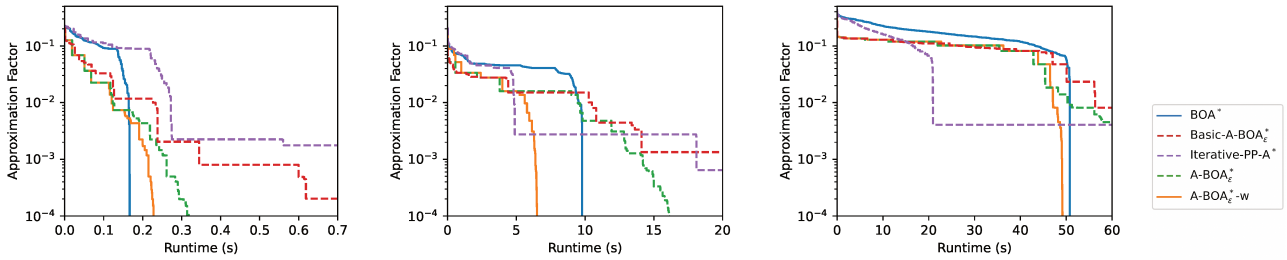
[2] This statement corrects a statement by Goldin and Salzman (2021).

[3] https://github.com/HanZhang39/anytime_BOA.git.

[4] http://users.diag.uniroma1.it/challenge9/download.shtml

	BAY $\lvert sols \rvert = 102$ on average			FLA $\lvert sols \rvert = 457$ on average			NE $\lvert sols \rvert = 930$ on average		
	Solved	Runtime	#Expansion	Solved	Runtime	#Expansion	Solved	Runtime	#Expansion
BOA*	25/25	**0.20**	161K	25/25	4.17	1,985K	25/25	37.28	10,072K
Basic-A-BOA$^*_\varepsilon$	25/25	9.45	1,641K	19/25	97.80	14,194K	15/25	178.57	29,567K
Iterative-PP-A*	25/25	2.41	530K	23/25	45.44	4,124K	18/25	123.23	9,517K
A-BOA$^*_\varepsilon$	25/25	0.34	271K	25/25	5.74	2,843K	23/25	61.23	12,403K
A-BOA$^*_\varepsilon$-w	25/25	0.29	**153K**	25/25	**3.80**	**1,759K**	25/25	**34.58**	**8,729K**

Table 1: Experimental results for different road maps.



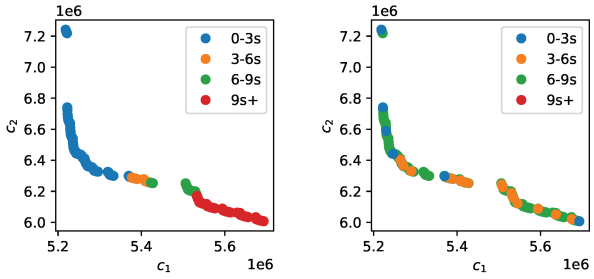
(a) BAY, $s_{\text{start}} = 75608$, $s_{\text{goal}} = 287676$ (b) FLA, $s_{\text{start}} = 450000$, $s_{\text{goal}} = 497452$ (c) NE, $s_{\text{start}} = 447786$, $s_{\text{goal}} = 1331803$

Figure 5: Anytime behaviors of different algorithms on three representative problem instances. The x-axis represents the deliberation time, and each graph shows the approximation factor of the solution set found over time. The faster the approximation factor decreases, the better.

generated 25 instances with randomly selected start and goal states. The $\mathbf{h}$-values are the exact minimum costs to the goal state for each single objective, computed with Dijkstra's algorithm. These $\mathbf{h}$-values are used by all algorithms, and the reported runtimes do not include this computation. However, the reported runtimes do include the computation of the weighted-sum heuristic function for A-BOA$^*_\varepsilon$-w, the only algorithm that uses it. For all algorithms, we set a five-minute runtime limit for solving each problem instance. We set parameter d for both Basic-A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$ to 4. A-BOA$^*_\varepsilon$-w uses the weighted-sum heuristic function with $w = 1$. We run all experiments on a MacBook Pro with an M1 Pro chip and 32GB of memory.

Table 1 shows the number of problem instances that are solved (that is, for which the algorithm finds a cost-unique Pareto-optimal frontier) within the runtime limit, the average runtime (in seconds, where timeout instances are counted as five minutes), and the average number of expanded nodes for different algorithms. Both Basic-A-BOA$^*_\varepsilon$ and Iterative-PP-A* have much larger average runtimes than A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$-w on all three road maps because they do not reuse previous search effort. A-BOA$^*_\varepsilon$-w has smaller average runtimes than A-BOA$^*_\varepsilon$ on all three road maps. It also has smaller average numbers of node expansions than BOA* on all three road maps and a smaller average runtime than BOA* on FLA and NE. Overall, for larger road maps, the overhead of computing the weighted-sum heuristic function is outweighed by the resulting acceleration of the search.

Fig. 5 shows the anytime behaviors of all algorithms for three problem instances of different difficulties, each from a different road network. A-BOA$^*_\varepsilon$-w has a similar behaviour as A-BOA$^*_\varepsilon$ in the beginning of the anytime search and then reduces the approximation factor much faster A-BOA$^*_\varepsilon$, which shows that the weighted-sum heuristic pruning is most effective in the later stages of the anytime search. In both problem instances (b) and (c), Iterative-PP-A* is the first algorithm to find a solution set with an approximation factor of less than 0.01 because it does not guarantee to find Pareto-optimal solutions and hence solves an arguably easier problem than the other four algorithms. However, since Iterative-PP-A* does not reuse its previous search effort, it needs more time than A-BOA$^*_\varepsilon$ and A-BOA$^*_\varepsilon$-w to decrease the approximation factor to below 0.001.



(a) BOA* (b) A-BOA$^*_\varepsilon$-w

Figure 6: The solution sets that BOA* and A-BOA$^*_\varepsilon$-w find for the problem instance of Fig. 5b. Colors indicate the runtime when solutions were found.

Fig. 6 shows the Pareto-optimal frontier for the problem instance in Figure 5b, marked with different colors accord-

ing to the runtime when each solution was found. BOA* finds solutions in a lexicographic order, while A-BOA$^*_\varepsilon$-w first finds a set of solutions with diverse costs and then adds more solutions to the solution set.

8 Conclusion

In this paper, we proposed an anytime approximate bi-objective search algorithm, called A-BOA$^*_\varepsilon$. It efficiently reuses its search effort from previous iterations and uses a novel pruning technique. Our experimental results show that it is substantially more efficient than an anytime approximate bi-objective search algorithm that does not reuse previous search effort. When given a limited amount of deliberation time, A-BOA$^*_\varepsilon$ often finds Pareto-optimal solution sets with much smaller approximation factors than those found by BOA*.

There are several interesting directions for future work. One direction is to develop anytime bi-objective search algorithms which find solutions that are not necessarily Pareto-optimal. PP-A* leverages this relaxed objective to find a set of solutions with a small approximation factor quickly. However, it is currently unclear how to reuse the search effort of PP-A* efficiently in this case. Another direction is to generalize anytime approximate bi-objective search algorithms to search problems with more than two objectives.

9 Acknowledgments

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779, 1935712, 2112533, and 2121028. The research was also supported by the United States-Israel Binational Science Foundation (BSF) under grant number 2021643 and Centro Nacional de Inteligencia Artificial CENIA, FB210017, BASAL, ANID. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

Bachmann, D.; Bökler, F.; Kopec, J.; Popp, K.; Schwarze, B.; and Weichert, F. 2018. Multi-Objective Optimisation Based Planning of Power-Line Grid Expansions. *ISPRS International Journal of Geo-Information*, 7(7): 258.

Breugem, T.; Dollevoet, T.; and van den Heuvel, W. 2017. Analysis of FPTASes for the Multi-Objective Shortest Path Problem. *Computers & Operations Research*, 78: 44–58.

Bronfman, A.; Marianov, V.; Paredes-Belmar, G.; and Lüer-Villagra, A. 2015. The Maximin HAZMAT Routing Problem. *European Journal of Operational Research*, 241(1): 15–27.

Cohen, L.; Greco, M.; Ma, H.; Hernández, C.; Felner, A.; Kumar, T. K. S.; and Koenig, S. 2018. Anytime Focal Search with Applications. In *IJCAI*, 1434–1441.

Ehrgott, M. 2005. *Multicriteria Optimization*. Springer, 2nd edition.

Fu, M.; Kuntz, A.; Salzman, O.; and Alterovitz, R. 2019. Toward Asymptotically-Optimal Inspection Planning via Efficient Near-Optimal Graph Search. In *RSS*.

Fu, M.; Salzman, O.; and Alterovitz, R. 2021. Computationally-Efficient Roadmap-Based Inspection Planning via Incremental Lazy Search. In *ICRA*, 7449–7456.

Goldin, B.; and Salzman, O. 2021. Approximate Bi-Criteria Search by Efficient Representation of Subsets of the Pareto-Optimal Frontier. In *ICAPS*, 149–158.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hernandez, C. U.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazoy, L.; and Koenig, S. 2020. A Simple and Fast Bi-Objective Search Algorithm. In *ICAPS*, 143–151.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2003. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *NIPS*, 767–774.

Mandow, L.; and De La Cruz, J. L. P. 2010. Multiobjective A* Search with Consistent Heuristics. *Journal of the ACM (JACM)*, 57(5): 1–25.

Pulido, F.-J.; Mandow, L.; and Pérez-de-la Cruz, J.-L. 2015. Dimensionality Reduction in Multiobjective Shortest Path Search. *Computers & Operations Research*, 64: 60–70.

Stern, R.; Felner, A.; van den Berg, J.; Puzis, R.; Shah, R.; and Goldberg, K. 2014. Potential-Based Bounded-Cost Search and Anytime Non-Parametric A*. *Artificial intelligence*, 214: 1–25.

Tsaggouris, G.; and Zaroliagis, C. D. 2009. Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications. *Theory of Computing Systems*, 45(1): 162–186.

van den Berg, J.; Shah, R.; Huang, A.; and Goldberg, K. Y. 2011. Anytime Nonparametric A*. In *AAAI*, 105–111.

Warburton, A. 1987. Approximation of Pareto Optima in Multiple-Objective, Shortest-Path Problems. *Operations Research*, 35(1): 70–79.