Anytime Approximate Bi-Objective Search

Han Zhang1, Oren Salzman2, T. K. Satish Kumar1, Ariel Felner3, Carlos Hernández Ulloa4, Sven Koenig1

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, tskwork@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 $\epsilon$ 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 $A^{\epsilon}$ (A-BOA$^\epsilon$). A-BOA$^\epsilon$ 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$^\epsilon$ 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$^\epsilon$ even slightly outperforms BOA$, a state-of-the-art bi-objective search algorithm, for finding the Pareto-optimal frontier. Moreover, given only a limited amount of deliberation time, A-BOA$^\epsilon$ finds solutions that collectively approximate the Pareto-optimal frontier much better than the solutions found by BOA$^\epsilon$.

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 $\pi$, we use $c_1(\pi)$ and $c_2(\pi)$ to denote the accumulated first and second costs of the edges of $\pi$, respectively. A solution path $\pi$ is better than, i.e., dominates, another solution path $\pi'$ 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$^\epsilon$ (Hernandez et al. 2020), NAMOA$^\epsilon$ (Mandow and De La Cruz 2010), and NAMOA$^\epsilon$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 Zafeiropoulos 2009). Here, we are given a user-specified approximation factor $\epsilon > 0$. A path $\pi$ $\epsilon$-dominates another path $\pi'$ if and only if each cost component of $\pi$ is less than or equal to $(1 + \epsilon)$ times the respective cost component of $\pi'$. An $\epsilon$-approximate Pareto-optimal frontier is a set of solution paths $\Pi_\epsilon$ such that any Pareto-optimal solution path is $\epsilon$-dominated by some path in $\Pi_\epsilon$. In structured real-world problems, the $\epsilon$-approximate Pareto-optimal frontier can have substantially fewer solution paths than the Pareto-optimal frontier even for small $\epsilon$. Existing work (Goldin and Salzman 2021) has already demonstrated that search algorithms can find an $\epsilon$-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 $A^*-\varepsilon$ (A-BOA$^*$). A-BOA$^*$ 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. A-BOA$^*$ 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 $\varepsilon$. However, this methodology is inefficient since search effort is duplicated across different values of $\varepsilon$. A-BOA$^*$ 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$^*$ 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$^*$. 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$^*$ significantly outperforms the baseline algorithms, both in terms of runtime and number of node expansions. In fact, the most advanced variant of A-BOA$^*$ even slightly outperforms BOA$^*$ for finding the Pareto-optimal frontier. Moreover, with a limited amount of time, A-BOA$^*$ 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 $v$. The addition of two 2-tuples $v$ and $v'$ is defined as $v + v' = (v_1 + v'_1, v_2 + v'_2)$. We define the following types of domination:

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

A (bi-objective) search graph is a tuple $(S, E, c)$, where $S$ is a finite set of states and $E \subseteq S \times S$ is a finite set of edges. Cost function $c : E \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ maps an edge to two positive real numbers. We define $c_i : E \to \mathbb{R}_{\geq 0}, i \in \{1, 2\}$, as the function that maps an edge $e$ to the $i$-th component of $c(e)$. A path is a sequence of states $\pi = \{s_1, s_2, \ldots, s_\ell\}$ such that $(s_j, s_{j+1}) \in E$ for all $j \in \{1, 2, \ldots, \ell - 1\}$.

We use $s(\pi)$ to denote the ending state of $\pi$, i.e., $s_\ell$. We use $\text{succ}(s) = \{s' \in S | (s, s') \in E\}$ to denote the successors of state $s$. By extending $\pi$ with an edge $(s_\ell, s_{\ell+1})$, we obtain a new path $\langle s_1, s_2, \ldots, s_\ell, s_{\ell+1} \rangle$. We say that a path $\pi'$ extends another path $\pi$ if and only if $\pi'$ can be obtained by applying a sequence of extend operations on $\pi$.

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

We say that $\pi$ dominates (resp. weakly dominates and $\varepsilon$-dominates) $\pi'$ if and only if $f(\pi) < f(\pi')$ (resp. $f(\pi) \leq f(\pi')$ and $f(\pi) \preceq_{\varepsilon} f(\pi')$). 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 $\Pi^{\varepsilon}$ 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 $\varepsilon$, an $\varepsilon$-approximate Pareto-optimal frontier $\Pi^{\varepsilon}$ is a subset of the Pareto-optimal frontier such that, for any Pareto-optimal solution $\pi$ of $P$, there exists a solution $\pi' \in \Pi^{\varepsilon}$ with $\pi' \preceq_{\varepsilon} \pi$. Note that any cost-unique Pareto-optimal frontier is also an $\varepsilon$-approximate Pareto-optimal frontier for $\varepsilon = 0$.

We define the domination factor of a path $\pi'$ over another path $\pi$ as

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

which measures how “good” $f(\pi')$ approximates $f(\pi)$. It is easy to verify that $f(\pi') \preceq_{\varepsilon} f(\pi)$ if and only if $\varepsilon \geq DF(\pi', \pi)$. For a set of solutions $\Pi$, we define its approximation factor as

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

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

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

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 \( \text{sols} \) that it has found so far. In BOA*, a node represents a path \( \pi \) from \( \text{start} \) to some state. In this paper, we therefore use "node" and "path" interchangeably. For each state \( s \), BOA* uses \( g^\pi_{\text{min}}(s) \) to store the minimum \( g_{\text{f}} \)-value of all expanded paths that end at state \( s \). Alg. 1 shows the pseudocode of BOA*.

\[ g^\pi_{\text{min}}(s) \leq g^\pi_{\text{min}}(s'), \quad \text{for all } s', s' = s \text{ such that } g^\pi_{\text{min}}(s') < g^\pi_{\text{min}}(s) \]

for each state \( s \) do the following:

- \( s_{\text{goal}} \leq g^\pi_{\text{min}}(s) \)
- \( \text{OPEN} \leftarrow \{(s_{\text{start}})\} \)
- \( \text{sols} \leftarrow \emptyset \)
- \( \beta \leftarrow 0 \)
- \( \text{tmp} \leftarrow \emptyset \)
- \( \text{OPEN} \leftarrow \emptyset \)

while \( \text{OPEN} \neq \emptyset \) do:

- \( \beta \leftarrow \beta + 1 \)

for each \( s' \in \text{succ}(s) \) do:

- \( \text{If } g^\pi_{\text{min}}(s') + f^\pi_{\text{min}}(s') < \text{MIN} \) then:

- \( \text{MIN} \leftarrow g^\pi_{\text{min}}(s') + f^\pi_{\text{min}}(s') \)

- \( \text{OPEN} \leftarrow \text{OPEN} \cup \{(s', \text{MIN})\} \)

- \( \text{tmin} \leftarrow \text{MIN} \)

- \( \text{OPEN} \leftarrow \text{OPEN} \setminus \{s\} \)

- \( \text{OPEN} \leftarrow \text{OPEN} \cup \{(s', \text{MIN} + f^\pi_{\text{min}}(s'))\} \)

Algorithm 1: BOA*

Input: A problem instance \((S, E, c, s_{\text{start}}, s_{\text{goal}})\)
Output: A cost-unique Pareto-optimal frontier

3.3 Preliminaries: Intervals

Definition 1 (Interval). Let \( \pi_{dl} \) and \( \pi_{br} \) be two Pareto-optimal solutions such that \( f_1(\pi_{dl}) < f_1(\pi_{br}) \) (and hence \( f_2(\pi_{dl}) > f_2(\pi_{br}) \)). Furthermore, let \( \Pi \) be a set of paths from \( s_{\text{start}} \) (not necessarily ending at \( s_{\text{goal}} \)) such that, \( \forall \pi \in \Pi \)

\[ f_2(\pi_{dl}) \leq f_2(\pi) \leq f_2(\pi_{br}) \]

We define its corresponding interval as the 3-tuple \( I = (\pi_{dl}, \pi_{br}, \Pi) \) and refer to \( \pi_{dl}, \pi_{br}, \Pi \) 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, \( \Pi \) is a set of paths whose \( f \)-values fall “between” the \( f \)-values of \( \pi_{dl} \) and \( \pi_{br} \).
Algorithm 2: Anytime BOA*  
\begin{itemize}
  \item Input: A problem instance \((S, E, c, s_{\text{start}}, s_{\text{goal}})\) and parameter \(d\) for decreasing approximation factor
  \item Output: An approximate Pareto-optimal frontier
  \item \(1\) \(\pi_t \leftarrow \) a Pareto-optimal solution with the lexicographically smallest \((c_1, c_2)\)
  \item \(2\) \(\pi_b \leftarrow \) a Pareto-optimal solution with the lexicographically smallest \((c_2, c_1)\)
  \item \(3\) \(I_s \leftarrow \langle \pi_t, \pi_b, \{s_{\text{start}}\} \rangle \)
  \item \(4\) \(\text{ILIST} \leftarrow \{I_s\} \)
  \item \(5\) while \(\exists I \in \text{ILIST}: \hat{\varepsilon}(I) \neq 0\) do
  \item \(6\) \(I \leftarrow \text{argmax}_{I \in \text{ILIST}} \hat{\varepsilon}(I)\)
  \item \(7\) \(\text{ILIST}' \leftarrow \text{Search}(I, \hat{\varepsilon}(I)/d)\)
  \item \(8\) remove \(I\) from \(\text{ILIST}\)
  \item \(9\) add all intervals of \(\text{ILIST}'\) to \(\text{ILIST}\)
  \item \(10\) return \(\text{ConstructSols}(I)\)
\end{itemize}

Algorithm 3: Search  
\begin{itemize}
  \item Input: An interval \(I = \langle \pi_t, \pi_b, \Pi \rangle\) and an approximation factor \(\varepsilon\)
  \item Output: A list of intervals, each with an approximation factor not larger than \(\varepsilon\)
  \item \(1\) \(\text{OPEN} \leftarrow \Pi, \Pi_{\text{imp}} \leftarrow \emptyset, \pi_{\text{imp}} \leftarrow \pi_t\)
  \item for each \(s \in S\) do
  \item \(2\) \(g^\text{min}_2(s) \leftarrow \infty\)
  \item while \(\text{OPEN} \neq \emptyset\) do
  \item \(3\) \(\pi \leftarrow \text{OPEN}, \text{extract}_\text{min}()\)
  \item if \(\text{is-dominated}()\) then
  \item \(4\) \(\hat{\varepsilon}_2(s(\pi)) \leftarrow g_2(\pi)\)
  \item \(5\) \(s(\pi) = s_{\text{goal}}\) then
  \item \(6\) add \(\langle \pi_{\text{imp}}, \pi, \Pi_{\text{imp}} \rangle\) to \(\text{ILIST}\)
  \item \(7\) \(\pi_{\text{imp}} \leftarrow \pi, \Pi_{\text{imp}} \leftarrow \emptyset\)
  \item continue
  \item for each \(s' \in \text{succ}(s(\pi))\) do
  \item \(8\) \(\pi' \leftarrow \text{extend}(\pi, (s(\pi), s'))\)
  \item if \(\text{is-dominated}()\) then
  \item \(9\) \(\hat{\varepsilon}_2(s(\pi)) \leftarrow g_2^\text{min}(s(\pi)) \vee (1 + \varepsilon)g_2(\pi) \geq g_2^\text{min}(s_{\text{goal}})\)
  \item \(10\) return \(\text{True}\)
  \item if \(g_2(\pi) \geq g_2^\text{min}(s(\pi)) \vee (1 + \varepsilon)g_2(\pi) \geq g_2^\text{min}(s_{\text{goal}})\) then
  \item \(11\) add \(\pi\) to \(\text{OPEN}\)
  \item \(12\) return \(\text{True}\)
  \item \(13\) return \(\text{False}\)
\end{itemize}
The pruning conditions of BOA* 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 \texttt{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 \( \varepsilon \)-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 \( \varepsilon \)-value but still have the potential to extend to Pareto-optimal solutions. A-BOA* temporally stores these paths in \( \Pi_{\text{tmp}} \). Each time a solution is found, a new interval is created with the previously found solution \( \pi_{\text{tmp}} \) as its top-left solution, the newly found solution as its bottom-right solution, and \( \Pi_{\text{tmp}} \) as its to-expand paths, \( \pi_{\text{tmp}} \) is then set to the newly found solution and \( \Pi_{\text{tmp}} \) is emptied. When OPEN becomes empty, Alg. 3 creates a new interval with the previously found solution \( \pi_{\text{tmp}} \) as its top-left solution, \( \pi_{\text{br}} \) as its bottom-right solution, and \( \Pi_{\text{tmp}} \) 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 \( \varepsilon \)-value because all of its to-expand paths are \( \varepsilon \)-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* finds two Pareto-optimal solutions \( \pi_1 = [s_{\text{start}}, s_1, s_{\text{goal}}] \) and \( \pi_4 = [s_{\text{start}}, s_4, s_{\text{goal}}] \) and initializes ILIST with interval \( I_s = \{\pi_1, \pi_4, [s_{\text{start}}]\} \). The \( t \)-values for \( \pi_1, \pi_4, \) and path \([s_{\text{start}}]\) are \((2,10), (10,2), \) and \((2,2), \) respectively. The approximation factor for \( I_s \) is \( \hat{e}(I_s) = \min(DF(\pi_1, [s_{\text{start}}]), DF(\pi_4, [s_{\text{start}}])) = 4 \).

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

We have \( \hat{e}(I_1) = 0 \) by definition and \( \hat{e}(I_2) = \min(DF(\pi_2, [s_{\text{start}}, s_3]), DF(\pi_4, [s_{\text{start}}, s_3])) = 0.2 \). In the second iteration, A-BOA* invokes Alg. 3 with \( I_2 \) and \( \varepsilon = 0.05 \). Eventually, Alg. 3 finds solution \( \pi_3 = [s_{\text{start}}, s_3, s_{\text{goal}}] \) and returns two intervals \( \langle \pi_2, \pi_3, \emptyset \rangle \) and \( \langle \pi_3, \pi_4, \emptyset \rangle \). A-BOA* 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*, 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 \( h_w : S \rightarrow \mathbb{R} \) 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 \( \pi \) be some path, and let \( \pi' \) and \( \pi'' \) be two Pareto-optimal solutions that have been found by A-BOA*, and satisfy (i) \( f_1(\pi') \leq f_1(\pi) \), (ii) \( f_2(\pi'') \leq f_2(\pi) \), and (iii) \( f_1(\pi'') < f_1(\pi') \). If \( g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi'') + w \cdot f_2(\pi'') \), then any solution extending \( \pi \) is weakly dominated by \( \pi' \) or \( \pi'' \).

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

\[
f_2(\pi_{\text{sol}}) < f_2(\pi') \quad \text{and} \quad f_1(\pi_{\text{sol}}) < f_1(\pi'')
\]

because \( \pi_{\text{sol}} \) is not worse than or equal to \( \pi' \) or \( \pi'' \) in both objectives. Given that \( h_w \) is admissible and Eq. 3 is satisfied, we have

\[
f_1(\pi_{\text{sol}}) + w \cdot f_2(\pi_{\text{sol}}) = c_1(\pi_{\text{sol}}) + w \cdot c_2(\pi_{\text{sol}}) \geq g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi'') + w \cdot f_2(\pi').
\]
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 f-value of any path that extends π must lie in the shaded area of (a) because the heuristic function h is consistent. The shaded area in (b) represents f-values \( f_1 + w \cdot f_2 \geq g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \) for a given weighted-sum heuristic function h_w.
The 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 f-values that are weakly dominated by the 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
\[
f_1(\pi_{sol}) + w \cdot f_2(\pi_{sol}) < f_1(\pi′′) + w \cdot f_2(\pi′),
\]
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 \( h_3 \), the shaded area in Fig. 4a shows where the 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 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 \( \pi_{imp} \) and \( \pi_{br} \) as π′ and π′′, respectively. A path π can be pruned if, given weighted-sum heuristic function h_w,
\[
g_1(\pi) + w \cdot g_2(\pi) + h_w(s(\pi)) \geq f_1(\pi_{br}) + w \cdot f_2(\pi_{imp}).
\]
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 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 \( (g_2, c_1) \)-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*. After describing properties of ILIST and intervals, we show that (i) the set of solutions that A-BOA* has found at any point of time is a \( \max \{\varepsilon(I) | I \in \text{ILIST}\} \)-approximate Pareto-optimal frontier (Thm. 1) and that (ii) A-BOA* eventually finds a cost-unique Pareto-optimal frontier (Thm. 2).

Property 2. Let \( \{I_1, I_2, \ldots, I_M\} \) 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 \( i \in \{1, 2, \ldots, M - 1\} \), 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 \( \pi_{imp} \) 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 \( \{I_1, I_2, \ldots, I_N\} \) in Property 2, and let \( \Pi = \{\pi_1, \pi_2, \ldots, \pi_{N+1}\} \) be the sequence of solutions such that, for every \( i = 1, 2, \ldots, N \), \( \pi_i \) and \( \pi_{i+1} \) are the top-left and bottom-right solutions of \( I_i \), respectively. The solutions in \( \Pi \) 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 \( A-BOA^* \) prunes a path in Alg. 3 and this prevents it in the future from adding a solution \( \pi_{sol} \) that extends \( \pi \) to the solution set, then \( A-BOA^* \) has already found or will find in the future a solution that weakly dominates \( \pi_{sol} \).

Proof. There are three cases when \( A-BOA^* \) prunes a path \( \pi \):

1. It prunes path \( \pi \) on Line 21 of Alg. 3 because \( f_1(\pi) \geq f_1(\pi_{br}) \lor f_2(\pi) \geq f_2(\pi_{imp}) \). Then, \( A-BOA^* \) has already found a solution, namely \( \pi_1 \) or \( \pi_{br} \), that weakly dominates \( \pi \) (and hence also \( \pi_{sol} \)). This solution will eventually be included in the solution set.

2. It prunes path \( \pi \) on Line 21 of Alg. 3 because can_prune_wsh(\( \pi \)) holds. Then, according to Property 1, \( \pi_{imp} \) or \( \pi_{br} \) weakly dominates \( \pi_{sol} \). Both \( \pi_{imp} \) or \( \pi_{br} \) are solutions and will eventually be included in the solution set.
3. It prunes path \( \pi \) on Line 26 of Alg. 3 after storing this path in \( \Pi_{\text{imp}} \). 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 \( \pi \) on Line 26 of Alg. 3 without storing it in \( \Pi_{\text{imp}} \). Then, path \( \pi \) satisfies \( g_2(\pi) \geq g_2^{\min}(s(\pi)) \lor f_2(\pi) \geq g_2^{\min}(s_{\text{goal}}) \). This is the pruning condition of BOA\(^*\), and thus the proof of Lemma 7 of Hernandez et al. (2020) applies.

\[ \square \]

Lemma 2. In A-BOA\(_c\), the top-left and bottom-right solutions of any interval are Pareto-optimal solutions.

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

\[ \square \]

The following two theorems show that A-BOA\(_c\) 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\(_c\) finds after each iteration (namely, the set of top-left and bottom-right solutions of all intervals in ILIST) is a \( \max \{ \varepsilon(I) | I \in \text{ILIST} \} \)-approximate Pareto-optimal frontier.

Proof. Let \( \Pi \) denote the set of solutions that A-BOA\(_c\) finds after an iteration and \( \Pi^* \) 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 into a solution \( \pi_{\text{sol}} \) that weakly dominates \( \pi \). Because A-BOA\(_c\) uses consistent heuristic functions, we have \( f(\hat{\pi}) \leq f(\pi_{\text{sol}}) \leq f(\pi) \), and hence \( DF(\pi', \pi) \leq DF(\pi'', \hat{\pi}) \) for any path \( \pi' \). Let \( I = (\pi_{\text{tl}}, \pi_{\text{br}}, \Pi_I) \) 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 \( f_2(\pi_{\text{tl}}) \leq f_2(\hat{\pi}) \leq f_2(\pi_{\text{br}}) \) and hence

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

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

\[ \square \]

Theorem 2. A-BOA\(_c\) eventually finds a cost-unique Pareto-optimal frontier.

Proof. In each iteration, Alg. 3 invokes Alg. 3 with an interval \( I \) and an \( \varepsilon \)-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 \( \varepsilon \)-value because, for each one of them, the top-left solution \( \varepsilon \)-dominates every path in set of to-expand paths. From Thm. 1, the solution set found by A-BOA\(_c\) is a \( \max \{ \varepsilon(I) | I \in \text{ILIST} \} \)-approximate Pareto-optimal frontier. Therefore, A-BOA\(_c\) 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 \( \varepsilon \)-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.

\[ \square \]

7 Experimental Results

In this section, we evaluate A-BOA\(_c\) experimentally by comparing it with the following algorithms:

1. BOA\(^*\).
2. Basic-A-BOA\(_c\): Basic-A-BOA\(_c\) iteratively invokes BOA\(_c\), each time with the input approximation factor divided by a constant \( d \). Basic-A-BOA\(_c\) is similar to A-BOA\(_c\) 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 \( \varepsilon \), PP-A\(^*\) finds a set of solutions \( \Pi \) such that, for any solution \( \pi \in \Pi \), there exists a solution \( \pi'' \in \Pi \) that \( \varepsilon \)-dominates \( \pi \), although \( \pi'' \) is not necessarily a Pareto-optimal solution. PP-A\(^*\) uses more complicated data structures than BOA\(_c\), 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 \( \varepsilon \)-values, namely 0.01, 0.001, 0.0001, and 0.

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

\(^3\)This statement corrects a statement by Goldin and Salzman (2021).

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

\(^4\)https://github.com/HanZhang39/anytime_BOA.git.
Table 1: Experimental results for different road maps.

<table>
<thead>
<tr>
<th></th>
<th>BAY</th>
<th>FLA</th>
<th>NE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sols</td>
<td>sols</td>
<td>sols</td>
</tr>
<tr>
<td>Solved</td>
<td>Runtime</td>
<td>#Expansion</td>
<td>Runtime</td>
</tr>
<tr>
<td>BOA*</td>
<td>25/25</td>
<td>0.20</td>
<td>161K</td>
</tr>
<tr>
<td>Iterative-PP-A*</td>
<td>25/25</td>
<td>2.41</td>
<td>530K</td>
</tr>
<tr>
<td>A-BOA*</td>
<td>25/25</td>
<td>0.34</td>
<td>271K</td>
</tr>
<tr>
<td>A-BOA*-w</td>
<td>25/25</td>
<td>0.29</td>
<td>153K</td>
</tr>
</tbody>
</table>

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

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 h-values are the exact minimum costs to the goal state for each single objective, computed with Dijkstra’s algorithm. These 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*-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* and A-BOA* to 4. A-BOA*-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* and Iterative-PP-A* have much larger average runtimes than A-BOA* and A-BOA*-w on all three road maps because they do not reuse previous search effort. A-BOA*-w has smaller average runtimes than A-BOA* 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*-w has a similar behaviour as A-BOA* in the beginning of the anytime search and then reduces the approximation factor much faster A-BOA*, 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* and A-BOA*-w to decrease the approximation factor to below 0.001.

Fig. 6 shows the Pareto-optimal frontier for the problem instance in Fig. 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∗-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∗. 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∗ 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


