Extending Weighted Heuristic Search to Bi-Objective Search Problems

Hans Kühn Leiva[1] , **Jorge A. Baier**[1,2] , **Carlos Hernández Ulloa**[2,3] , **Oren Salzman**[4] , **Ariel Felner**[5] and **Sven Koenig**[6]
[1]Pontificia Universidad Católica de Chile
[2]Centro Nacional de Inteligencia Artificial
[3]Universidad San Sebastián
[4]Technion – Israel Institute of Technology
[5]Ben-Gurion University of the Negev
[6]University of California Irvine
hans.kuhn@uc.cl, jbaier@uc.cl, carlos.hernandez@uss.cl, osalzman@cs.technion.ac.il, felner@bgu.ac.il, skoenig@usc.edu

Abstract

In heuristic search, a well-known technique to speed up search while providing a suboptimality guarantee is to multiply the heuristic function by a weight $w > 1$. In this paper, we study the theoretical and practical implications of using such a technique in bi-objective heuristic search, a natural academic exercise that has remained unexplored. We introduce Weighted BOA$^*_\epsilon$ (WBOA$^*_\epsilon$), a weighted version of the BOA* algorithm, which uses two real parameters: a weight w for the heuristic and an approximation factor ϵ. Higher values of w and ϵ allow for faster computation of approximate Pareto-optimal solution sets. We prove that WBOA$^*_\epsilon$ returns a representative solution set containing $(w-1,\epsilon)$-approximate solutions. We empirically compare it to A*pex, the state-of-the-art approximate bi-objective search algorithm. We find that WBOA$^*_\epsilon$ is competitive with A*pex: when using perfect heuristic functions, in road maps WBOA$^*_\epsilon$ is faster for higher approximation factors, while in grid maps WBOA$^*_\epsilon$ dominates A*pex. With imperfect heuristic functions, WBOA$^*_\epsilon$ performs better than A*pex for lower approximation factors, while the opposite is true for larger approximation factors.

1 Introduction

In bi-objective search, we are given a graph G where each edge has an associated pair of costs, an initial state and a goal state. The task is to find a Pareto-optimal solution set, which contains all solution paths (paths that connect the start and goal states) that are not dominated by other solution paths. Bi-objective search is important in AI research. Many real-world decision problems can be represented as an instance of it, such as balancing distance and risk when transporting hazardous materials [Bronfman *et al.*, 2015] or ecological and financial impact when planning power grid expansions [Bachmann *et al.*, 2018].

Finding the Pareto-optimal solution set may be very computationally expensive. Indeed, its size can be exponential in the size of the graph [Breugem *et al.*, 2017]. A key approach to tackle this problem is to compute approximations of the solution set using a relaxed version of dominance called $\boldsymbol{\epsilon}$-dominance [Breugem *et al.*, 2017; Zhang *et al.*, 2022; Goldin and Salzman, 2021; Salzman *et al.*, 2023]. While a path π with cost $(\mathbf{c}_1, \mathbf{c}_2)$ *weakly dominates* path π' with cost $(\mathbf{c}'_1, \mathbf{c}'_2)$ when $\mathbf{c}_1 \leq \mathbf{c}'_1$ and $\mathbf{c}_2 \leq \mathbf{c}'_2$, we say that π $\boldsymbol{\epsilon}$-*dominates* path π' if $\mathbf{c}_1 \leq (1+\epsilon_1)\mathbf{c}'_1$ and $\mathbf{c}_2 \leq (1+\epsilon_2)\mathbf{c}'_2$, with $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2)$. An intuitive reason why search runs faster with $\boldsymbol{\epsilon}$-dominance is that such a relation allows more pruning than standard dominance.

In this paper, we propose an algorithm to compute approximate Pareto-optimal solution sets drawing inspiration from standard heuristic search. In single-objective search, costs are scalars numbers, and A* [Hart *et al.*, 1968], which uses an Open priority queue with the priority function $f(s) = g(s) + h(s)$, is guaranteed to find optimal solutions when the heuristic function is admissible. In many problems, faster execution times can be obtained when multiplying the heuristic function by a real number $w > 1$. The resulting algorithm, Weighted A* [Pohl, 1970], can be proven to return solutions whose suboptimality is bounded by w. Despite the practical advantage of weighted heuristics in single-objective search and the fact that algorithms in bi-objective search are based on A*, the theoretical and practical implications of multiplying the heuristic by a positive weight $w > 1$ in bi-objective search have remained unexplored.

We propose the Weighted Bi-Objective A* (WBOA$^*_\epsilon$) algorithm, which builds on top of the state-of-the-art Bi-Objective A* algorithm [Hernández *et al.*, 2020] and its approximate variant BOA$^*_\epsilon$ [Goldin and Salzman, 2021]. WBOA$^*_\epsilon$ multiplies the first component of the heuristic by a given scalar parameter w (with $w > 1$) and can be proven to return a representative approximation of the Pareto-optimal solution set. Specifically, for every optimal path π^* in the Pareto-optimal solution set, there is a solution returned by WBOA$^*_\epsilon$ that $(w-1,\epsilon)$-dominates π^*. For an illustration of this property, Figure 1 shows the regions in which WBOA$^*_\epsilon$ guarantees

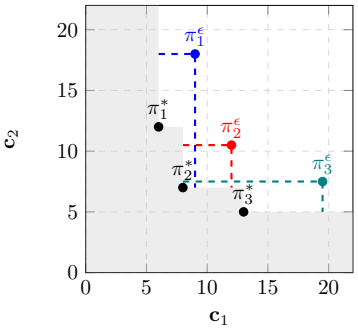


Figure 1: The plot shows a Pareto-optimal solution set with three paths $\{\pi_1^*, \pi_2^*, \pi_3^*\}$. The gray region represents impossible cost pairs for solution paths. Each path π_i^ϵ represents the cost of π_i multiplied by 1.5. With parameters $w = 1.5, \epsilon = 0.5$, our Theorem 1 establishes that WBOA$^*_\epsilon$ returns a solution set which contains, for every $i \in \{1, 2, 3\}$, a π_i^+ that weakly dominates π_i^ϵ.

at least one solution will be found.

We evaluate WBOA$^*_\epsilon$ on two benchmark domains: DIMACS challenge road maps and Moving AI gridworld maps. For road maps we use perfect heuristic functions computed via Dijkstra's algorithm, while for gridworld maps we evaluate both perfect heuristics and the less informed Octile distance heuristic. WBOA$^*_\epsilon$ achieves substantial speedups over BOA*; for instance, on the LKS road map with 2,758,119 states, we observe speedups of 96× and 351× when using $w = 1.1$ and $w = 1.2$, respectively. In our empirical comparison with A*pex [Zhang *et al.*, 2022], the state-of-the-art approximate bi-objective search algorithm, we find that WBOA$^*_\epsilon$ is competitive across different scenarios. With perfect heuristic functions, WBOA$^*_\epsilon$ outperforms A*pex on road maps at the highest approximation factors, while on gridworld maps WBOA$^*_\epsilon$ dominates A*pex across nearly all approximation factors tested. When using the less informed Octile heuristic on gridworld maps, WBOA$^*_\epsilon$ maintains its advantage at lower approximation factors, though A*pex becomes superior as the approximation factor increases. Additionally, WBOA$^*_\epsilon$ tends to return fewer solutions than A*pex, which may be preferable in applications where concise solution sets are desired.

2 Notation

Boldface lower case letters are used to represent two-component vectors or vectorial functions. For $\mathbf{v}$ a vector, $\mathbf{v}_1$ and $\mathbf{v}_2$ represent the first and second components of $\mathbf{v}$ respectively. The addition of two such vectors is defined as $\mathbf{v} + \mathbf{w} = (\mathbf{v}_1 + \mathbf{w}_1, \mathbf{v}_2 + \mathbf{w}_2)$. We use $\mathbf{v} \preceq \mathbf{w}$ to denote that $\mathbf{v}$ *weakly dominates* $\mathbf{w}$ (that is, $\mathbf{v}_1 \leq \mathbf{w}_1$ and $\mathbf{v}_2 \leq \mathbf{w}_2$), and $\mathbf{v} \prec \mathbf{w}$ to denote that $\mathbf{v}$ *(strongly) dominates* $\mathbf{w}$ (that is, $\mathbf{v} \preceq \mathbf{w}$ and $\mathbf{v}_1 < \mathbf{w}_1$ or $\mathbf{v}_2 < \mathbf{w}_2$). We also use $\mathbf{v} \preceq_{\boldsymbol{\epsilon}} \mathbf{w}$ if, for a vector of approximation factors $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2)$, $\mathbf{v}_1 \leq (1 + \epsilon_1)\mathbf{w}_1$ and $\mathbf{v}_2 \leq (1 + \epsilon_2)\mathbf{w}_2$. In this case, we say that $\mathbf{v}$ $\boldsymbol{\epsilon}$-dominates $\mathbf{w}$.

A bi-objective search graph is a triple $(S, E, \mathbf{c})$, where S is a set of states, $E \subseteq S \times S$ is a set of directed edges and $c : E \to \mathbb{R}^+_0 \times \mathbb{R}^+_0$ is a cost function mapping every edge to a pair of non-negative numbers. A *path* π is a sequence $e_1, e_2, \ldots, e_n$ of edges in E such that there exists a sequence of states $s_1, s_2, \ldots, s_{n+1}$ such that $e_i = (s_i, s_{i+1})$, for every $i \in \{1, \ldots, n\}$, which we say *starts* in s_1 and *reaches* s_{n+1}. We denote the state reached by a non-empty path π by $\text{last}(\pi)$. The symbol λ denotes the empty sequence. The cost of a path π is denoted by $\mathbf{c}(\pi)$ and is defined as the sum of the costs of all edges in π.

A bi-objective search instance is a tuple $(S, E, \mathbf{c}, s_{\text{start}}, s_{\text{goal}})$, where $P = (S, E, \mathbf{c})$ is a bi-objective search graph, and where s_{start} and s_{goal} denote, respectively, the start and goal states (in S). A path π is a *solution* to a problem if its starting state is s_{start} and its final state is s_{goal}. A path π is an *optimal solution* if π is a solution and no other solution π' is such that $\pi' \prec \pi$. The Pareto-optimal solution set for a given instance P contains all optimal solutions to that instance, and is denoted by sols_P.

Bi-objective search algorithms utilize heuristic functions, defined as $\mathbf{h} : S \to \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$. For a state $s \in S$, $\mathbf{h}(s)$ serves as an estimate of the minimum cost for each component on all paths from s to s_{goal}. A heuristic function $\mathbf{h}$ is admissible if and only if, for every state s, $\mathbf{h}(s) \preceq \mathbf{c}(\pi)$ for every path π from s to s_{goal}. A heuristic function $\mathbf{h}$ is consistent if and only if $\mathbf{h}(s_{\text{goal}}) = (0, 0)$ and $\mathbf{h}(s) \preceq \mathbf{c}(s, t) + \mathbf{h}(t)$ for all $(s, t) \in E$. Consistency implies admissibility.

3 Background

3.1 Bi-Objective A*

Bi-Objective A* (BOA*) is a state-of-the-art best-first bi-objective search algorithm, whose algorithmic design closely resembles A*. BOA* has been shown to have better performance than other bi-objective search algorithms based on A* [Hernández *et al.*, 2020].

BOA* uses a priority queue Open, which stores paths in the search graph.[1] Each path π in Open starts in s_{start} and is associated with an $\mathbf{f}$-value defined as $\mathbf{f}(\pi) = \mathbf{c}(\pi) + \mathbf{h}(\text{last}(\pi))$, where $\mathbf{h}$ is a consistent heuristic function.

The execution of BOA* resembles that of A* in that in both a path is extracted from Open (Line 7), the state s reached by such a path is expanded (Line 15), and resulting paths reaching the successors of s are added to Open (Line 20).

Paths in Open are sorted according to their $\mathbf{f}$ values in lexicographically ascending order; that is, paths are sorted in ascending order by their first-component cost and, in case of ties, in ascending order by their second-component cost. Together with the fact that $\mathbf{h}$ is consistent, such an order guarantees that given any state s in the search graph, paths found by BOA* to s have a non-decreasing $\mathbf{c}_1$-value, which in turn allows implementing pruning in constant time by applying dimensionality reduction [Pulido *et al.*, 2015]. Specifically, BOA* keeps, for each state s, a $g_2^{\min}$-value that stores the

[1] Hernández *et al.* [2020] consider Open as composed by *nodes* which actually represent paths, albeit efficiently implemented via a state plus a parent pointer. In this paper, we define Open as containing paths rather than nodes, since this allows us to simplify our theoretical analysis. Nevertheless, as we see later, WBOA$^*_\epsilon$ can be implemented as efficiently as BOA* since it is only different from it in a few lines of code.

Algorithm 1: Bi-Objective A* (BOA*)

```
   Input : A search problem (S, E, c, s_start, s_goal) and a
           consistent heuristic function h
1  sols ← Empty list
2  for each s ∈ S do
3  |  g_2^min(s) ← ∞
4  f(λ) ← h(s_start)
5  Initialize Open and insert the empty path λ into it
6  while Open ≠ ∅ do
7  |  Remove a path π from Open with the lexicographically
   |    smallest f-value
8  |  s ← state reached by π or s_start if π is empty
9  |  if c_2(π) ≥ g_2^min(s) ∨ f_2(π) ≥ g_2^min(s_goal) then
10 |  |  continue
11 |  g_2^min(s) ← c_2(π)
12 |  if s = s_goal then
13 |  |  Append π to sols
14 |  |  continue
15 |  for each (s, t) ∈ E do
16 |  |  π' ← π, (s, t)
17 |  |  f(π') ← c(π') + h(t)
18 |  |  if c_2(π') ≥ g_2^min(t) ∨ f_2(π') ≥ g_2^min(s_goal) then
19 |  |  |  continue
20 |  |  Insert π' to Open
21 return sols
```

$\mathbf{c}_2$-value of the last path found that reached s. A path π is pruned either when $\mathbf{c}_2(\pi) \geq g_2^{\min}(\text{last}(\pi))$ (*local pruning*) or when $\mathbf{c}_2(\pi) \geq g_2^{\min}(s_{goal})$ (*global pruning*). Hernández *et al.* [2020] show that this form of pruning is *sound*, which means that a prefix of an optimal solution may be pruned only if another solution of equal cost is returned.

3.2 BOA$^*_\epsilon$

Goldin and Salzman [2021] propose a simple modification to BOA* that allows it to compute a subset of the Pareto-optimal solution set. In BOA$^*_\epsilon$, the global pruning rule of BOA* is changed to $(1+\epsilon)f_2(\pi) \geq g_2^{\min}(s_{goal})$. It can be proven that for any given instance, BOA$^*_\epsilon$ returns a subset of the solutions returned by BOA* such that no pair of solutions in such a set $(0, \epsilon)$-dominates each other [Goldin and Salzman, 2021].

4 Weighted BOA$^*_\epsilon$

Weighted BOA$^*_\epsilon$ (WBOA$^*_\epsilon$) is obtained by making two simple modifications to BOA$^*_\epsilon$. First, it associates the following $\mathbf{f}$-value to each node in Open.

$$\mathbf{f}^w(x) = \mathbf{c}(x) + (w\mathbf{h}_1(x), \mathbf{h}_2(x)),$$

with $w > 1$. Second, we replace Line 13 in Algorithm 1 with Algorithm 2. WBOA$^*_\epsilon$ is conceptually simple because it requires almost no additional modification with respect to BOA*. In contrast, A*pex [Zhang *et al.*, 2022], the state-of-the-art approximate search algorithm for bi-objective (and multi-objective) search problems, has to check if it can merge two paths at each step. This checking technique provides a performance advantage to A*pex when there are many similarities between paths [Zhang *et al.*, 2022], but it comes with computational overhead.

Algorithm 2: Handling new solutions in WBOA$^*_\epsilon$

```
1 while |sols| > 0 and c_1(last(sols)) ≥ c_1(π) do
2 |  Remove the last element from sols
3 Append π to sols
```

Before we continue with the theoretical analysis, we make two important observations. Firstly, in the definition of $\mathbf{f}^w$, the second component is not affected by w. There are two reasons for this. First, BOA$^*_\epsilon$ already has an approximation factor associated with the second component and thus by not changing $\mathbf{f}$'s second component we allow the theoretical properties of BOA$^*_\epsilon$ to be leveraged by WBOA$^*_\epsilon$ (see Theorem 1). Second, from a practical perspective, in WBOA$^*_\epsilon$, like BOA*, Open is sorted lexicographically and thus the weight w affects search mainly in the first component, since the second component of $\mathbf{f}^w$ is only used as a tiebreaker.

The second observation is that WBOA$^*_\epsilon$, in practice, does not use a consistent heuristic even if its input heuristic $\mathbf{h}$ is consistent, since multiplying the first component of $\mathbf{h}$ by w can make its first component inconsistent. This implies that an important property of BOA$^*_\epsilon$ is broken: paths to a given state s are not guaranteed to be found in monotonically non-decreasing $\mathbf{c}_1$ order.

One implication of using a non-monotonic $\mathbf{f}$ function is that BOA$^*_\epsilon$'s technique of using $g_2^{\min}$ value of s to prune in constant time does not allow pruning all dominated paths to s. This means that while a path π may be initially found to a certain state, subsequently another path π' may be found to the same state such that $\mathbf{c}_1(\pi') < \mathbf{c}_1(\pi)$ while at the same time $\mathbf{c}_2(\pi') < \mathbf{c}_2(\pi)$. Even though this means that π' dominates π, WBOA$^*_\epsilon$ *does not* prune π from Open. This is why WBOA$^*_\epsilon$ uses Algorithm 2 to filter out dominated paths from the returned solution set. Observe that after running Algorithm 2, the paths in sols remain sorted by ascending $\mathbf{c}_1$-value. This, together with the local pruning rule, which still guarantees that solutions found have a decreasing $\mathbf{c}_2$ cost, ensures that sols contains a set of non-dominated paths.

While BOA* never prunes a prefix of an optimal path from Open, WBOA$^*_\epsilon$ can do so, which is important to note since it influences our theoretical analysis. Indeed, consider that π is a path that passes through state u; for simplicity, let us represent π as the concatenation of two paths $\pi_1\pi_2$ where π_1 is the sequence that reaches state u. Since $\mathbf{f}^w$ is non-monotonic, during the execution of WBOA$^*_\epsilon$ it may be possible to extract a path ρ to u, before extracting π_1, even if $\mathbf{c}_1(\rho) > \mathbf{c}_1(\pi_1)$. After finding ρ, $g_2^{\min}(u)$ is set to $\mathbf{c}_2(\rho)$. If it is also the case that $\mathbf{c}_2(\rho) < \mathbf{c}_2(\pi_1)$ then, due to the local pruning rule, π will be pruned if it is ever extracted from Open, meaning the optimal solution π will not be found.

4.1 An Example Execution

Figure 2 and Table 1 show an example of an execution of WBOA$^*_\epsilon$ over a search graph, using the parameters $w = 2$ and $\epsilon = 1$. For the execution, we consider a perfect heuristic distance, which can be computed using Dijkstra's algorithm. For ease of understanding, in this section we represent a path $(s_1, s_2)\ldots(s_{n-1}, s_n)$ by its sequence of states in the form

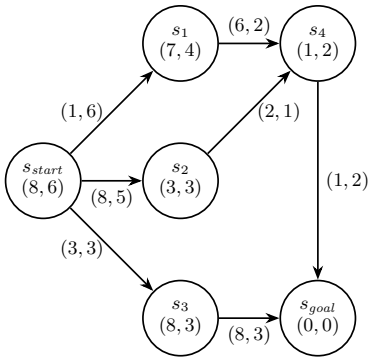


Figure 2: Example search graph. Each state contains a pair of values representing its (perfect) $\mathbf{h}$-value.

of $s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_{n-1} \rightarrow s_n$. Although the Pareto-optimal paths of the search instance are $s_{start} \rightarrow s_1 \rightarrow s_4 \rightarrow s_{goal}$ (with a cost of $(8, 10)$) and $s_{start} \rightarrow s_3 \rightarrow s_{goal}$ (with a cost of $(11, 6)$), neither of these is returned by the algorithm as their prefixes end up being pruned. The events in each iteration of the algorithm are as follows:

1. The initial empty path λ at s_{start} is expanded, adding to the Open list the paths $s_{start} \rightarrow s_1$, $s_{start} \rightarrow s_2$, and $s_{start} \rightarrow s_3$.
2. $s_{start} \rightarrow s_2$ is expanded. $g_2^{\min}(s_2)$ is set to 5 and $s_{start} \rightarrow s_2 \rightarrow s_4$ is added to Open.
3. $s_{start} \rightarrow s_2 \rightarrow s_4$ is expanded. $g_2^{\min}(s_4)$ is set to 6, which means that the path $s_{start} \rightarrow s_1$, if expanded, will have its only successor be locally pruned. $s_{start} \rightarrow s_2 \rightarrow s_4 \rightarrow s_{goal}$ is added to Open.
4. $s_{start} \rightarrow s_2 \rightarrow s_4 \rightarrow s_{goal}$ is expanded and found as a solution with a cost of $(11, 8)$. $g_2^{\min}(s_{goal})$ is set to 8.
5. $s_{start} \rightarrow s_1$ is extracted from open. As it has a $\mathbf{f}_2^w$ value of 10 and $g_2^{\min}(s_{goal}) = 8$, it is pruned according to the global prune no matter the value of ϵ.
6. $s_{start} \rightarrow s_3$ is extracted from open. As it has a $\mathbf{f}_2^w$ value of 6, $g_2^{\min}(s_{goal}) = 8$, and $\epsilon = 1$, it is pruned according to the global prune because $(1 + 1) \cdot 6 \geq 8$.
7. Finally, as Open is empty, the execution of WBOA$^*_\epsilon$ finishes. The approximate Pareto-optimal solution set returned is the single solution path $s_{start} \rightarrow s_2 \rightarrow s_4 \rightarrow s_{goal}$.

5 Theoretical Results

In this section we prove our main theoretical result. Based on the previous discussion, in which we saw that optimal paths may be pruned by a run of WBOA$^*_\epsilon$, we formally define when a solution path locally prunes another solution path, a notion that is useful later for our proof.

Definition 1. *Given an execution of WBOA$^*_\epsilon$ over a bi-objective search instance P we say that path π* is locally pruned by path *ρ if and only if:*

1. *both π and ρ reach a common intermediate state u, from where they continue with the same path. Formally, $\pi = \pi'\sigma$ and $\rho = \rho'\sigma$, such that both π' and ρ' reach the same state u;*
2. *during execution, path ρ' is extracted from* Open *before π' and not pruned, while a (not necessarily proper) prefix of π' is in* Open*; finally,*
3. $\mathbf{c}_2(\pi') \geq \mathbf{c}_2(\rho')$.

$w = 2, \epsilon = 1$		
Iteration	Open list $(\text{last}(\pi), g(\pi), \mathbf{f}^w(\pi))$	Update of $g_2^{\min}(s(\pi))$
1	$(s_{start}, (0,0), (16,6)) \leftarrow$	$g_2^{\min}(s_{start}) = 0$
2	$(s_1, (1,6), (15,10))$ $(s_2, (8,5), (14,8)) \leftarrow$ $(s_3, (3,3), (19,6))$	$g_2^{\min}(s_2) = 5$
3	$(s_1, (1,6), (15,10))$ $(s_3, (3,3), (19,6))$ $(s_4, (10,6), (12,8)) \leftarrow$	$g_2^{\min}(s_4) = 6$
4	$(s_1, (1,6), (15,10))$ $(s_3, (3,3), (19,6))$ $(s_{goal}, (11,8), (11,8)) \leftarrow$	$g_2^{\min}(s_{goal}) = 8$
5	$(s_1, (1,6), (15,10)) \leftarrow$ $(s_3, (3,3), (19,6))$	
6	$(s_3, (3,3), (19,6)) \leftarrow$	
7	empty	

Table 1: A trace of the Open priority queue and updates of $g_2^{\min}$ during each iteration. $\leftarrow$ marks the path extracted at each iteration. No $g_2^{\min}$ update indicates that the extracted path was pruned.

Additional notation To simplify presentation, in the remainder of the proof we assume that if π is a path and $\mathbf{h}$ is a heuristic function, then $\mathbf{h}(\pi)$ abbreviates the notation $\mathbf{h}(s_\pi)$, where s_π is the state reached by π. Furthermore, if a path π is the concatenation of ρ and σ, then the notation $\pi - \sigma$ corresponds to removing the suffix σ from π; therefore $\pi - \sigma = \rho$.

We start off our theoretical analysis with Lemma 1, which is a standard result of consistent heuristics.

Lemma 1. *Let π_1 and π_2 be paths such that $\pi_1\pi_2$ is also a path. Then* $\mathbf{h}(\pi_1) \leq \mathbf{c}(\pi_2) + \mathbf{h}(\pi_1\pi_2)$.

Our first original theoretical result below shows that whenever a path is locally pruned by another path, then the pruner path's cost $(w - 1, 0)$-dominates the pruned path's cost. This holds, in particular, when the pruned path is an optimal path.

Lemma 2. *In an execution of WBOA$^*_\epsilon$, if $\pi_1, \pi_2, \ldots, \pi_n$ is a sequence of paths such that π_{i+1} locally prunes π_i, for every $i \in \{1, \ldots, n-1\}$, then* $\mathbf{c}(\pi_n) \preceq_{(w-1,0)} \mathbf{c}(\pi_1)$.

Proof. The proof is by induction on n. For the base case, $n = 1$, we have trivially $\mathbf{c}(\pi_1) \preceq_{(w-1,0)} \mathbf{c}(\pi_1)$.

For the induction, we assume the property holds for $n = k$, and we prove for $n = k + 1$. Since $\mathbf{c}(\pi_k) \preceq_{(w-1,0)} \mathbf{c}(\pi_1)$, we have:

$$\mathbf{c}_1(\pi_k) \leq w\mathbf{c}_1(\pi_1), \tag{1}$$
$$\mathbf{c}_2(\pi_k) \leq \mathbf{c}_2(\pi_1). \tag{2}$$

To prove $\mathbf{c}(\pi_{k+1}) \preceq_{(w-1,0)} \mathbf{c}(\pi_1)$ we need to prove the two inequalities:

$$\mathbf{c}_1(\pi_{k+1}) \leq w\mathbf{c}_1(\pi_1) \quad (3)$$
$$\mathbf{c}_2(\pi_{k+1}) \leq \mathbf{c}_2(\pi_1) \quad (4)$$

We start by proving (4). Observe that, by Definition 1, since π_{k+1} locally prunes π_k, $\mathbf{c}_2(\pi_{k+1}) \leq \mathbf{c}_2(\pi_k)$, which, together with (2), implies (4).

We continue with the proof for (3). From Definition 1, we know that π_{k+1} and π_k share a common suffix; therefore $\pi_k = \rho_k\sigma$ and $\pi_{k+1} = \rho_{k+1}\sigma$, where both ρ_k and ρ_{k+1} reach the same state u_k. Because ρ_{k+1} locally prunes ρ_k at state u_k, we observe that a (not necessarily proper) prefix of ρ_k, say ρ'_k, is in Open at the time ρ_{k+1} is extracted; that is $\rho_k = \rho'_k\rho''_k$. At that point in execution $\mathbf{f}_1^w(\rho_{k+1}) \leq \mathbf{f}_1^w(\rho'_k)$, thus:

$$\mathbf{c}_1(\rho_{k+1}) + w\mathbf{h}_1(\rho_{k+1}) \leq \mathbf{c}_1(\rho'_k) + w\mathbf{h}_1(\rho'_k), \quad (5)$$

Observe that the sequence of k paths given by:

$$\tau_1, \tau_2, \ldots, \tau_k, \quad (6)$$

where $\tau_i = \pi_i - \rho''_k\sigma$ is also such that τ_{i+1} locally prunes τ_i, for every $i \in \{1, \ldots, k-1\}$. [2] Therefore, we apply the inductive hypothesis over the sequence in (6) and obtain:

$$\mathbf{c}_1(\tau_k) = \mathbf{c}_1(\rho'_k) \leq w\mathbf{c}_1(\pi_1 - \rho''_k\sigma) = w\mathbf{c}_1(\tau_1). \quad (7)$$

Now we instantiate Lemma 1 as follows:

$$\mathbf{h}_1(\rho'_k) \leq \mathbf{c}_1(\rho''_k) + \mathbf{h}_1(\rho''_k). \quad (8)$$

By adding (5), (7) and the multiplication of (8) by w, and by using the fact that $\mathbf{h}_1(\rho_{k+1}) = \mathbf{h}_1(\rho''_k)$, since both ρ_{k+1} and ρ''_k reach the same state, we obtain:

$$\mathbf{c}_1(\rho_{k+1}) \leq w(\mathbf{c}_1(\pi_1 - \rho''_k\sigma) + \mathbf{c}_1(\rho''_k)) = w\mathbf{c}_1(\pi_1 - \sigma). \quad (9)$$

Now we add $\mathbf{c}_1(\sigma)$ to the left-hand side of (9) and $w\mathbf{c}_1(\sigma)$ to the right-hand side of (9) and obtain the desired result. □

Theorem 1. *Let π^* be an optimal solution to a bi-objective search instance P. Then the solution set returned by WBOA$^*_\epsilon$ over P contains a path that $(w-1, \epsilon)$-dominates π^*.*

Proof. An execution of WBOA$^*_\epsilon$ defines a sequence of paths $\rho_1, \rho_2, \ldots, \rho_n$ ($n \geq 1$) such that $\rho_1 = \pi^*$ and ρ_{i+1} locally prunes ρ_i, for every $i \in \{1, \ldots, n-1\}$, and such that ρ_n is not locally pruned by any other path. From Lemma 2,

$$\mathbf{c}(\rho_n) \preceq_{(w-1,0)} \mathbf{c}(\rho_1) = \mathbf{c}(\pi^*). \quad (10)$$

If ρ_n is in the solution set returned by WBOA$^*_\epsilon$, then the result is straightforward since Inequality (10) implies $\mathbf{c}(\rho_n) \preceq_{(w-1,\epsilon)} \mathbf{c}(\pi^*)$.

If ρ_n is not in the solution set returned by WBOA$^*_\epsilon$, then a prefix ρ'_n of ρ_n is pruned by the global pruning rule. Therefore, when ρ'_n is extracted, there is a solution π in sols, such that:

$$\mathbf{c}_2(\pi) \leq (1+\epsilon)(\mathbf{c}_2(\rho'_n) + \mathbf{h}_2(\rho'_n)) \quad (11)$$

Because $\mathbf{h}$ is admissible, Inequality (11) implies:

$$\mathbf{c}_2(\pi) \leq (1+\epsilon)\mathbf{c}_2(\rho_n) \quad (12)$$

In addition, when WBOA$^*_\epsilon$ extracts π from Open there is a prefix of ρ_n, ρ''_n in Open such that $\mathbf{f}_1^w(\pi) \leq \mathbf{f}_1^w(\rho''_n)$. This implies that $\mathbf{c}_1(\pi) \leq \mathbf{c}_1(\rho''_n) + w\mathbf{h}_1(\rho''_n)$. By admissibility of $\mathbf{h}$:

$$\mathbf{c}_1(\pi) \leq w\mathbf{c}_1(\rho_n) \quad (13)$$

Inequalities (12) and (13) imply:

$$\mathbf{c}(\pi) \preceq_{(w-1,\epsilon)} \mathbf{c}(\rho_n) \quad (14)$$

In turn, (10) and (14) imply:

$$\mathbf{c}(\pi) \preceq_{(w-1,\epsilon)} \mathbf{c}(\pi^*) \quad (15)$$

Finally, after ρ_n is globally pruned by π in sols, it could be the case that π is removed from sols by another solution π_2, and later on π_2 removed by π_3, and so on and so forth. More formally $\pi_1, \ldots, \pi_m$ can be a sequence of solutions such that $\pi_1 = \pi$, and π_{i+1} removes π_i from sols via Algorithm 2, and π_m is not removed by any solution and therefore $\pi_m \in$ sols when sols is returned. But Algorithm 2 guarantees that $\mathbf{c}(\pi_{i+1}) \prec \mathbf{c}(\pi_i)$. Thus, $\mathbf{c}(\pi_m) \preceq \mathbf{c}(\pi)$, which together with (15) yields $\mathbf{c}(\pi_m) \preceq_{(w-1,\epsilon)} \mathbf{c}(\pi^*)$, which is what we wanted to prove. □

6 Experimental Results

We compared WBOA$^*_\epsilon$ to the approximate multi-objective search algorithm A*pex,[3] as it has recently been shown to be the most efficient algorithm of the state of the art and surpasses even dedicated bi-objective algorithms like PP-A* [Goldin and Salzman, 2021]. A*pex uses two approximation factors (ϵ_1, ϵ_2) as its parameters for bi-objective search, each corresponding to their respective objective.

We ran all tests on a Linux machine with a 4.4GHz Intel® Core™ i5-12400 CPU and 32GB RAM. WBOA$^*_\epsilon$ was implemented in C++[4] and we used the C++ implementation of A*pex provided by the authors. To compare the algorithms, we used the standard road maps benchmarks obtained from the 9th DIMACS Implementation Challenge - Shortest Paths[5], with the cost components representing travel distance ($\mathbf{c}_1$) and time ($\mathbf{c}_2$). We compare the algorithms in two large maps of the dataset. We used 76 search instances of LKS road map and 50 instances of CAL road map, which represent the available queries that ran to completion for every approximation value with both algorithms. Additionally, we used 8-neighbor grid maps from the Moving AI Lab repository,[6] with $\mathbf{c}_1$ representing travel distance and $\mathbf{c}_2$ being a random number between 1 and 6. We compare the algorithms in two maps of size 1024×1024 (Expedition and Paris-1) and in two maps of size 512×512 (Random-512-20-0 and Maze-512-16-0). We used 50 random search instances for each map.

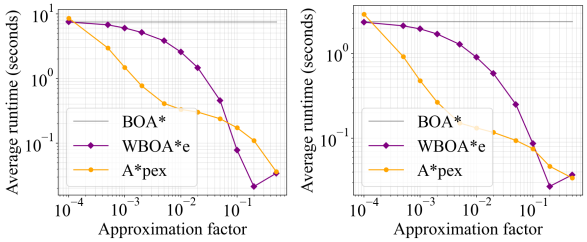
[2] Note that all paths $\pi_1, \ldots, \pi_k$ share a common suffix representing the path from the state u_{k-1} (where π_{k-1} was locally pruned) to last(π_1). As the local pruning of π_k occurs after that of π_{k-1}, the state u_{k-1} is traversed in ρ'_k and $\rho''_k\sigma$ is a suffix of all paths $\pi_1, \ldots, \pi_k$.

[3] There exist three main implementations of A*pex which vary in merge strategy. We use the *Greedy* variant as described by Zhang *et al.* [2022], as this one tends to be superior in their evaluation.

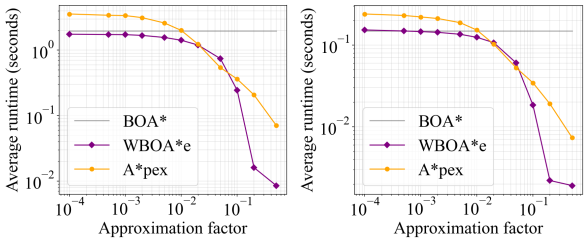
[4] https://github.com/hansjkl/WBOAE

[5] https://www.diag.uniroma1.it/challenge9/download.shtml

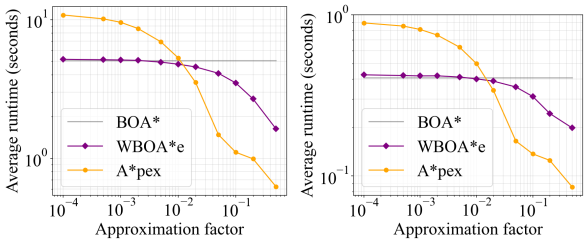
[6] https://movingai.com/benchmarks/grids.html



(a) LKS map, Dijkstra (b) CAL map, Dijkstra

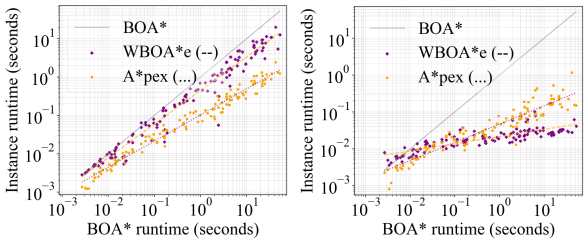


(c) Expedition map, Dijkstra (d) Random map, Dijkstra

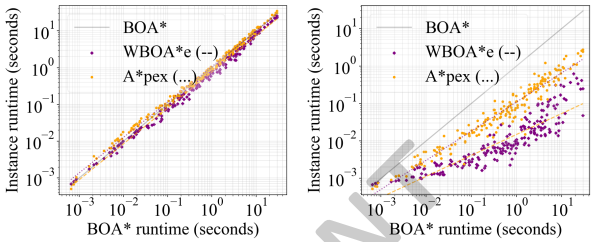


(e) Expedition map, Octile (f) Random map, Octile

Figure 3: Average runtimes of WBOA$^*_\epsilon$ and A*pex across all instances for a given map and heuristic plotted against ϵ'.



(a) Road (Dijkstra), $\epsilon' = 0.01$ (b) Road (Dijkstra), $\epsilon = 0.2$



(c) Grid (Dijkstra), $\epsilon = 0.01$ (d) Grid (Dijkstra), $\epsilon = 0.2$

Figure 4: Runtimes of WBOA$^*_\epsilon$ and A*pex plotted against the runtime of BOA* for a given ϵ' and a type of map.

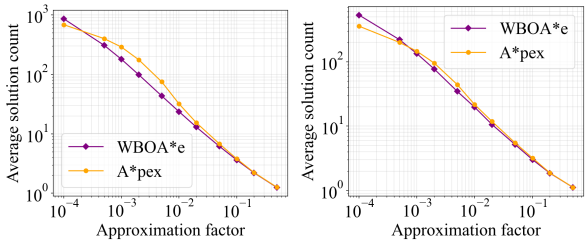
The heuristic $\mathbf{h}$ used corresponds to the exact minimum travel distances and minimum travel times to s_{goal}, computed using Dijkstra's algorithm. In grid maps, we also run experiments with the Octile distance. For $\mathbf{h}_1$ we used the standard Octile distance and for $\mathbf{h}_2$ we use a cost equal to one for both orthogonal and diagonal moves. The computation time of $\mathbf{h}$ is omitted from the reported runtime of each test, as both algorithms compute it in the same manner. To provide equivalent theoretical guarantees from both algorithms, we use a single approximation factor $\epsilon' > 0$ so that WBOA$^*_\epsilon$ uses parameters $(w = 1 + \epsilon', \epsilon = \epsilon')$ and A*pex uses $\epsilon = (\epsilon', \epsilon')$. We evaluated 11 ϵ' values: 0.0001, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, and 0.5.

6.1 Runtime

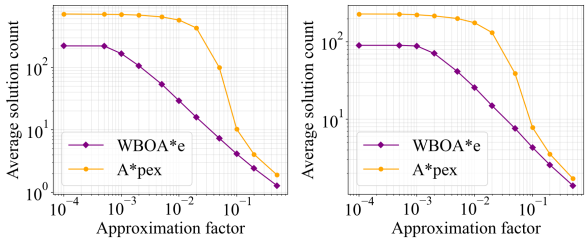
Figure 3 shows the average runtime of both algorithms on all search instances for each map and each approximation value tested. For the grid maps, we have chosen Expedition and Random as representatives, because the results from Paris and Maze512 are similar. We observed that when the algorithms used a Dijkstra (i.e., perfect) heuristic function on road maps A*pex outperform WBOA$^*_\epsilon$ in most of the ϵ values evaluated (7 of 11 values), and on grid maps the opposite occurs: WBOA$^*_\epsilon$ outperforms A*pex on 10 out of the 11 ϵ values evaluated. When the algorithms were run with the Octile heuristic function on the grid maps, we observed that WBOA$^*_\epsilon$ outperforms A*pex for the 6 smaller ϵ values evaluated, and A*pex outperforms WBOA$^*_\epsilon$ for the remaining values.

Figure 4 shows the runtimes of each instance using a Dijkstra heuristic for values of $\epsilon' \in \{0.01, 0.2\}$, sorted along the x-axis in ascending order according to the runtime of BOA* for each. We observed that on road maps with ϵ' equal to 0.01, A*pex is faster than WBOA$^*_\epsilon$ in most of the instances, and on grid maps the opposite happens. On the other hand, with ϵ' equal to 0.2, WBOA$^*_\epsilon$ is faster than A*pex in road and grid maps in almost all instances. Figure 4b also shows us that for some approximation values, such as 0.2, WBOA$^*_\epsilon$ scales better than A*pex as the runtime of BOA* increases.

These differences in performance by map type and heuristic arise from the different pruning mechanisms of each algorithm. A*pex acts on a local level by merging paths which share a last state, which means that graphs where each state has a higher degree (such as grid maps, where states usually have degree 7 or 8) lead to paths in Open less often sharing a final state during the execution of the algorithm, allowing for fewer path mergers. Meanwhile, maps with lower state degrees (such as road maps, with degrees around 2 or 3) allow for more mergers and better time optimization. Similarly, as A*pex relies on the merger of paths in Open, the exact path extraction order as determined by the used heuristic is less likely to stop pruning. Furthermore, the cost components in DIMACS road maps have been shown to have a strong positive correlation [Halle *et al.*, 2025], which allows A*pex to more easily merge paths even while using lower approximation factors. WBOA$^*_\epsilon$, meanwhile, executes its local prunes according to a locally determined $g_2^{\min}$ value, which means high-quality paths being extracted early are vital for efficient pruning. The algorithm also allows for paths to be explored in depth according to the given weight to allow solutions to be found earlier, which means the global pruning can act earlier



(a) LKS map, Dijkstra (b) CAL map, Dijkstra



(c) Expedition map, Dijkstra (d) Random map, Dijkstra

Figure 5: Average number of solutions found for bi-objective search instances for different maps and approximation factors ϵ'. Results for grid maps with Octile heuristic are similar to those with Dijkstra.

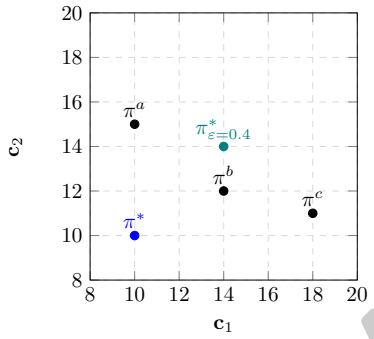


Figure 6: An optimal solution π^* and an example approximate Pareto-optimal solution set $\{\pi^a, \pi^b, \pi^c\}$. The point $\pi^*_{\varepsilon=0.4}$ shows the costs of the optimal solution multiplied by 1.4. Such a point is dominated by π^b, so we know that π^b (0.4, 0.4)-dominates π^*.

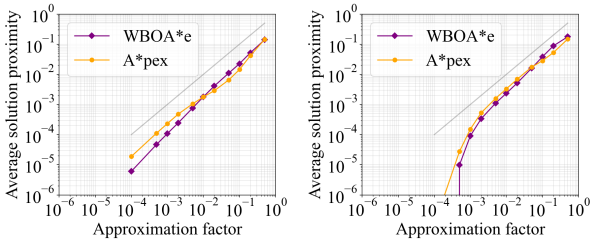
in a map with higher degree states when compared to A*pex.

6.2 Number and quality of solutions

Figure 5 shows the average solution quantity found by both algorithms for each map and approximation factor. We observe that WBOA$^*_\epsilon$ tends to return fewer solutions than A*pex, for each search instance, with the difference being more pronounced for grid maps.

We evaluate the quality of a returned solution set using so-called *proximity values*. Formally, the proximity value of an optimal solution path π^* with respect to a solution set S is the smallest real number ε such that there exists one path π in S such that $\mathbf{c}(\pi) \preceq_{(\varepsilon,\varepsilon)} \mathbf{c}(\pi^*)$. To illustrate this concept, Figure 6 shows an optimal solution π^* and a set of found solutions, with the smallest possible value for ε being 0.4.

To evaluate the quality of an approximate solution set S, we average the proximity value of every solution path π in a



(a) Road, Dijkstra (b) Grid, Dijkstra

Figure 7: Average proximity value across all instances for each algorithm's solution sets according to their approximation factors for a given type of map and algorithm. Results for grid maps with Octile heuristic are similar to those with Dijkstra.

Pareto-optimal solution set with respect to S. Figure 7 shows the average proximity value of all found solution sets for each algorithm and approximation factor in both types of map. We generally see WBOA$^*_\epsilon$ finding more representative solutions for lower approximation factors, while A*pex gains an advantage at higher values such as 0.05, 0.1, 0.2 and 0.5.

7 Summary and Conclusions

We introduced Weighted BOA$^*_\epsilon$ (WBOA$^*_\epsilon$), a novel approximate bi-objective search algorithm that extends the weighted heuristic technique from single-objective search to the bi-objective setting. Our approach builds on the state-of-the-art BOA* algorithm and its approximate variant BOA$^*_\epsilon$ by multiplying the first component of the heuristic function by a weight $w > 1$. This simple modification, combined with an approximation factor ϵ, enables faster computation of approximate Pareto-optimal solution sets while providing theoretical guarantees on solution quality. We prove WBOA$^*_\epsilon$ returns a $(w-1, \epsilon)$-representative solution set. This result establishes a relationship between the weight parameter and the approximation quality, filling a gap in the literature where weighted heuristics had not been studied in bi-objective search.

Our experimental evaluation compared WBOA$^*_\epsilon$ against A*pex, the current state-of-the-art approximate bi-objective search algorithm, across two benchmark domains: DIMACS road maps and Moving AI gridworld maps. The results show that WBOA$^*_\epsilon$ is competitive with A*pex, with performance depending on map characteristics, heuristic quality, and approximation factors. The performance differences can be attributed to the distinct mechanisms of the two algorithms. A*pex operates through local path merging, which benefits from lower state degrees and correlated cost components, while WBOA$^*_\epsilon$ relies on global pruning with $g_2^{\min}$ values, which excels when high-quality paths are explored early and when state degrees are higher.

Our proximity analysis to measure quality of the solution set showed that WBOA$^*_\epsilon$ finds better-quality solution sets at lower approximation factors, while A*pex can be closer to the Pareto-optimal solution set at higher factors. Additionally, WBOA$^*_\epsilon$ generally returns fewer solutions than A*pex, which may be advantageous in scenarios where concise solution sets are preferred.

Acknowledgements

Jorge A. Baier and Carlos Hernández are grateful to the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. The research at the University of California, Irvine and the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 2544613, 2434916, and 2321786, as well as gifts from the Donald Bren Foundation. This research was supported by the United States-Israel Binational Science Foundation (BSF) grants no. 2021643. The work of Ariel Felner was supported by the Israel Science Foundation (ISF) grant #909/23.

References

[Bachmann *et al.*, 2018] Daniel Bachmann, Fritz Bökler, Jakob Kopec, Kira Popp, Björn Schwarze, and Frank Weichert. Multi-objective optimisation based planning of power-line grid expansions. *ISPRS International Journal of Geo-Information*, 7(7):258, 2018.

[Breugem *et al.*, 2017] Thomas Breugem, Twan Dollevoet, and Wilco van den Heuvel. Analysis of fptases for the multi-objective shortest path problem. *Computers & Operations Research*, 78:44–58, 2017.

[Bronfman *et al.*, 2015] Andrés Bronfman, Vladimir Marianov, Germán Paredes-Belmar, and Armin Lüer-Villagra. The maximin hazmat routing problem. *European Journal of Operational Research*, 241(1):15–27, 2015.

[Goldin and Salzman, 2021] Boris Goldin and Oren Salzman. Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, pages 149–158, Palo Alto, California, May 2021. AAAI Press.

[Halle *et al.*, 2025] Yaron Halle, Ariel Felner, Sven Koenig, and Oren Salzman. A preprocessing framework for efficient approximate bi-objective shortest-path computation in the presence of correlated objectives. In *Proceedings of the 18th International Symposium on Combinatorial Search*, pages 65–73, July 2025.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[Hernández *et al.*, 2020] Carlos Hernández, William Yeoh, Jorge A. Baier, Han Zhang, Luis Suazo, and Sven Koenig. A simple and fast bi-objective search algorithm. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 143–151, May 2020.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3–4):193–204, 1970.

[Pulido *et al.*, 2015] Francisco-Javier Pulido, Lawrence Mandow, and José-Luis Pérez-de-la-Cruz. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64:60–70, 2015.

[Salzman *et al.*, 2023] Oren Salzman, Ariel Felner, Carlos Hernández, Han Zhang, Shao-Hung Chan, and Sven Koenig. Heuristic-search approaches for the multi-objective shortest-path problem: Progress and research opportunities. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6759–6768. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Survey Track.

[Zhang *et al.*, 2022] Han Zhang, Oren Salzman, T. K. Satish Kumar, Ariel Felner, Carlos Hernández Ulloa, and Sven Koenig. A*pex: Efficient approximate multi-objective search on graphs. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 394–403, Palo Alto, California, June 2022. AAAI Press.