Theoretical Study on Multi-objective Heuristic Search

Shawn Skyler[1], **Shahaf Shperberg**[1], **Dor Atzmon**[2], **Ariel Felner**[1], **Oren Salzman**[3], **Shao-Hung Chan**[4], **Han Zhang**[4], **Sven Keonig**[4], **William Yeoh**[5], **Carlos Hernandez Ulloa**[6]
[1]Ben-Gurion University of the Negev
[2]Bar-Ilan University
[3]Technion – Israel Institute of Technology
[4]University of Southern California
[5]Washington University in St. Louis
[6]Universidad San Sebastián
{shawn@post.bgu, shperbsh@bgu, dor.atzmon@biu, felner@bgu, osalzman@cs.technion}.ac.il,
{shaohung, zhan645, skoenig}@usc.edu, wyeoh@wustl.edu, carlos.hernandez@uss.cl

Abstract

This paper provides a theoretical study on Multi-Objective Heuristic Search. We first classify states in the state space into must-expand, maybe-expand, and never-expand states and then transfer these definitions to nodes in the search tree. We then formalize a framework that generalizes A* to Multi-Objective Search. We study different ways to order nodes under this framework and its relation to traditional tie-breaking policies and provide theoretical findings. Finally, we study and empirically compare different ordering functions.

1 Introduction

In *Single-Objective Search* (SOS), the task is to find a least-cost path between two states. *Multi-Objective Search* (MOS) generalize SOS to the case where each edge in the graph is associated with multiple costs. Examples include pathfinding applications with multiple competing resources, such as time and fuel as well as many other applications [Bachmann *et al.*, 2018; Fu *et al.*, 2019; Fu *et al.*, 2021]. In MOS, the task is to find a (cost-unique) *Pareto-optimal frontier* (POF), which is a set of *undominated* paths between $start$ and $goal$, i.e., there is no other path with lower cost in all objectives. Various MOS algorithms were developed [Clímaco and Pascoal, 2012; Current and Marsh, 1993; Skriver, 2000; Tarapata, 2007; Ulungu and Teghem, 1991; Mandow and Pérez-de-la-Cruz, 2010], as well as *Bi-Objective Search* (BOS) algorithms for solving the two-dimensional MOS problems [Ahmadi *et al.*, 2021; Skyler *et al.*, 2022; Hernández *et al.*, 2023b]. A recent survey including open challenges has also appeared [Salzman *et al.*, 2023].

A classic theory for SOS characterizes the set of states that any unidirectional search algorithm must expand to prove the optimality of the solution [Dechter and Pearl, 1985]. While investigating the NAMOA* algorithm, Mandow and Pérez-de-la-Cruz (2010) (we denote that paper by MPC10) provided theoretical insights into MOS and identified *must-expand* and *never-expand* nodes, where each node is associated with an undominated path to some state in the graph.

In this paper we provide a theoretical study of MOS algorithms. First, we extend the work of MPC10 and present a comprehensive and unifying theory for the MOS setting. We introduce an alternative analysis that shifts the focus from *nodes* to *states*, providing a theory that classifies MOS states into *must-expand*, *maybe-expand*, and *never-expand* categories. We demonstrate that an algorithm sharing information among nodes corresponding to the same state need not expand all *must-expand* nodes as defined by MPC10 but has to expand all *must-expand* states, as per our definition.

We then formalize an algorithmic framework called MOS-A*, which generalizes A* [Hart *et al.*, 1968] to MOS and encompasses various existing MOS algorithms. In MOS-A*, a node with an undominated **f**-value is expanded in each iteration. In MOS, **f**-values lack a total order, and thus, multiple nodes with different undominated **f**-values may exist simultaneously. Therefore, an *Ordering Function* (OF) is introduced to impose a total order on undominated **f**-values. Additionally, a *tie-breaking policy* (TB) can be employed to resolve ties between nodes with the same **f**-values. We restate the *must-expand* and *never-expand* nodes identified by MPC10 in the context of MOS-A* and characterize *maybe-expand* nodes. Moreover, we formally prove that the choice of OF does not impact which nodes are expanded during the search, while different TBs can result in different node expansions. However, OFs can influence the rate and order by which solutions are discovered, which can be crucial when aiming for quick solution discovery in an anytime manner. To this end, we introduce several OFs, analyze their impact on solution discovery, and empirically compare them on roadmaps from [DIMACS, 2006] for MOS with 2–4 objectives.

2 Definitions and Background

A Multi-Objective Search graph G with d objectives is a tuple $\langle S, E, \mathbf{c} \rangle$, where S is the finite set of *states*, $E \subseteq S \times S$ is the finite set of edges, and $\mathbf{c} : E \rightarrow (\mathbb{R}_{\geq 0})^d$ is a *cost function* in the form of a d-dimensional vector (one dimension per objective) of non-negative costs. A *path* π from s_1 to s_m

is a sequence of states $s_1, s_2, \ldots, s_m$ such that $(s_i, s_{i+1}) \in E$ for all $i \in \{1, \ldots, m-1\}$. **Boldface** font is used to represent d-dimensional vectors in the form $\mathbf{v} = (v_1, v_2, \ldots, v_d)$. For two vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u}+\mathbf{v} = (u_1+v_1, \ldots, u_d+v_d)$. $\mathbf{c}(\pi) = \sum_{i=1}^{m-1} \mathbf{c}(s_i, s_{i+1})$ is the cost of path $\pi = s_1, \ldots, s_m$. Given two d-dimensional vectors $\mathbf{u}$ and $\mathbf{v}$, we say that $\mathbf{u}$ *weakly dominates* $\mathbf{v}$, denoted as $\mathbf{u} \preceq \mathbf{v}$, if $u_i \leq v_i$, for every $i \in \{1, \ldots, d\}$. Vector $\mathbf{u}$ *(strictly) dominates* $\mathbf{v}$, denoted as $\mathbf{u} \prec \mathbf{v}$, if $\mathbf{u} \preceq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$ (i.e., for at least one dimension i, $u_i < v_i$). If $u_i > v_i$ for some $i \in \{1, \ldots, d\}$, then we say that $\mathbf{u}$ *undominates* $\mathbf{v}$ (likewise $\mathbf{v}$ is *undominated by* $\mathbf{u}$), and denote this by $\mathbf{u} \not\preceq \mathbf{v}$. For example, $(4, 5) \preceq (4, 5)$ (they are in fact equal), $(4, 5) \prec (4, 7)$ (as $5 < 7$) and $(4, 7) \not\preceq (4, 5)$. Finally, we say that two vectors $\mathbf{u}$ and $\mathbf{v}$ are *mutually undominated* if both $\mathbf{u} \not\preceq \mathbf{v}$ and $\mathbf{v} \not\preceq \mathbf{u}$. For example, $(4, 8)$ and $(5, 6)$ are *mutually undominated*. Similarly, we say that path π *dominates* (*weakly dominates*) path π', denoted as $\pi \prec \pi'$ ($\pi \preceq \pi'$) if $\mathbf{c}(\pi) \prec \mathbf{c}(\pi')$ ($\mathbf{c}(\pi) \preceq \mathbf{c}(\pi')$).

A *search instance* is a tuple $I = \langle S, E, \mathbf{c}, start, goal, \mathbf{h} \rangle$, where $\langle S, E, \mathbf{c} \rangle$ is a MOS graph, $start, goal \in S$ are the *start* and *goal* states, and $\mathbf{h}$ is a heuristic function. A *solution* is a path from $start$ to $goal$. A solution is *Pareto-optimal* if it is undominated by any other solution. A *Pareto-optimal solution set* for I, denoted by POS, is the (maximum) set of Pareto-optimal solutions. Let $\mathcal{POC} = \{\mathbf{c}(\pi) \mid \pi \in POS\}$ denote the set of path costs in POS. By definition, any two members of $\mathcal{POC}$ are mutually undominated. However, there can be two paths with equal costs in POS. We thus define a *cost unique Pareto-optimal solution set*, also called the *Pareto-optimal frontier*, denoted by POF, as a subset of POS such that no two paths have equal costs. That is, there is only one path for every value in $\mathcal{POC}$. In the special case of SOS (where $d = 1$), POS includes *all* optimal-cost paths while POF includes only one optimal-cost path. Figure 1 shows different cost vectors of a BOS problem instance (where $d = 2$). The C points denote the $\mathcal{POC}$. This means that there are no solution costs in zones A or B. Similarly, solutions in C are undominated by any solution in Zone D.

We consider *Single-Valued Heuristic* functions which assign to each state a d-dimensional vector $\mathbf{h} = \{h_1, \ldots, h_d\}$ where each value h_i estimates the cost along dimension i. $\mathbf{h}$ is *admissible* if h_i is a lower bound of all the costs of dimension i in any member of $\mathcal{POC}$. Similarly, $\mathbf{h}$ is *consistent* iff (*i*) $\mathbf{h}(goal) = \mathbf{0}$ and (*ii*) $\mathbf{h}(s) \preceq \mathbf{c}(s, t) + \mathbf{h}(t)$ for all $(s, t) \in E$. The strongest possible single-valued heuristic, which is commonly used, is the *individual shortest path* heuristic, also called *(Single) Point* heuristic [Goldin and Salzman, 2021; Hernández *et al.*, 2023b; Pulido *et al.*, 2015; Skyler *et al.*, 2022]. That is, for $\mathbf{h}(n)$, $h_i(n)$ is the cost-minimal path from n to $goal$ using the i-th objective only. In Figure 1, S represents the value of the Point heuristic, and Zone $H \subset A$ includes all other admissible heuristics.

2.1 Classes of States in Single-Objective Search

Following Dechter and Pearl (1985), let I_{AD} denote problem instances that have an admissible heuristic, and let $I_{CON} \subset I_{AD}$ denote the instances that have a consistent heuristic. *Admissible algorithms* are algorithms that guarantee to return optimal solutions on all problem instances from I_{AD}. Likewise, a search algorithm is said to be DXBB [Eckerle *et al.*, 2017] if it is deterministic and has only black-box access to the graph and the heuristic, i.e., it can only discover states, edges, and cost by continuously applying an expansion operator from either $start$ or $goal$.

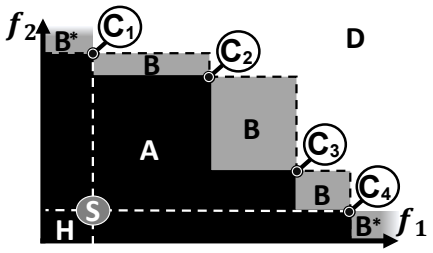


Figure 1: Different zones in a BOS problem instance. Every point in the figure is a two dimensional cost from $start$ to $goal$.

Admissible SOS algorithms must accomplish two tasks: (*i*) find a solution of cost C^* and (*ii*) prove that the returned solution is optimal. A classic theory is that any admissible DXBB unidirectional heuristic search algorithm *must expand* all states s such that there exists a path U from $start$ to s for which $c(U) + h(s) < C^*$, when running on an instance from I_{CON} [Dechter and Pearl, 1985]. Such states are called *Must-Expand States* (MESs). To fulfill task (*ii*), algorithms must expand every MES s or they might miss an optimal solution that passes through s. In addition, if $c(U)+h(s) > C^*$ for all paths U from $start$ to s, then state s should *never be expanded*, as it cannot lead to an optimal solution and, thus, does not contribute to accomplishing either tasks—such states are called *Never-Expand States* (NESs). Finally, depending on the *tie-breaking policy* (TB), some states s, where $c(U)+h(s) = C^*$, *might* be expanded to find an optimal solution (task (*i*)) and are called *Maybe-Expand States* (MBESs).

In the next sections, we generalize this theory from SOS to MOS. We distinguish between states in the graph and nodes in the search tree. We dedicate a separate section to each, motivated by the inherent differences between the two entities, as demonstrated in Section 4.4. It is noteworthy that the theoretical study of MCP10 (while exploring the NAMOA* algorithm) partially overlaps with our current work. Specifically, they defined the concepts of *must-expand paths* and *never-expand paths* (where paths in their notation align with nodes in our notation). As their definitions and corresponding proofs are pertinent to our discussion (Definitions 1,2 for states and Definitions 4,5 for nodes), we restate them here using our terminology for the sake of completeness.

3 Classes of States in Multi-Objective Search

In MOS, the two aforementioned tasks are modified as follows: (*i*) find all values of $\mathcal{POC}$ and (*ii*) prove that they are undominated by any other costs of solutions.

Definition 1 (Must-Expand States (MESs))**.** *A state s is a MES iff there exists a Pareto-optimal path U from $start$ to s such that $\mathbf{p} \not\preceq \mathbf{c}(U) + \boldsymbol{h}(s)$ for all path costs $\mathbf{p} \in \mathcal{POC}$.*

We partition the set of MESs into: *Optimality States* and *Completeness States*; their union is the set of MESs.

Definition 1.1 (Optimality States). *A state s is an* optimality

state *iff there exists a path U from $start$ to s and* $\exists \mathbf{p} \in \mathcal{POC}$ *such that* $\mathbf{c}(U) + \boldsymbol{h}(s) \prec \mathbf{p}$.

Optimality states are expanded to prove that any path in a POF is Pareto-optimal, i.e., that there is no path π' from $start$ to $goal$ that dominates some path $\pi \in POF$. Optimality states are analogous to MESs in SOS (with $f < C^*$). Zone A in Figure 1 corresponds to costs of optimality states (each via a relevant path U with the corresponding **f**-value).

In SOS, there is a total order between the cost of edges (and therefore between paths in the graph). However, in MOS, there is only a partial order because the costs of two edges may be mutually undominated. Thus, MOS algorithms must also consider states such that the costs of paths through them are mutually undominated with any cost in $\mathcal{POC}$ and prove that such costs cannot be added to $\mathcal{POC}$. Such states are called *completeness states*.

Definition 1.2 (Completeness States). *A state s is a* completeness state *iff there exists a path U from $start$ to s such that* $\forall \mathbf{p} \in \mathcal{POC}$*:* $\mathbf{c}(U) + \boldsymbol{h}(s) \not\preceq \mathbf{p}$ *and* $\mathbf{p} \not\preceq \mathbf{c}(U) + \boldsymbol{h}(s)$

Zone B corresponds to costs of completeness states (via a relevant path U). These states must be expanded to ensure that there are no more solution costs in $\mathcal{POC}$ (i.e., $\mathcal{POC}$ is complete). Completeness states have no analogy in SOS.

Definition 2 (Never-Expand States (NESs))**.** *A state s is a NES iff for every path U from $start$ to s there exists* $\mathbf{p} \in \mathcal{POC}$ *such that* $\mathbf{p} \prec \mathbf{c}(U) + \boldsymbol{h}(s)$.

Zone D in Figure 1 corresponds to NESs. They are analogous to the Never-Expand States in SOS.

Definition 3 (Maybe-Expand States (MBESs))**.** *A state s is a MBES iff s is not a MES and there exists a path U from $start$ to s and a cost* $\mathbf{p} \in \mathcal{POC}$ *such that* $\mathbf{c}(U) + \boldsymbol{h}(s) = \mathbf{p}$.

Zones C_1 to C_4 correspond to the costs of MBESs, analogous to states with $f = C^*$ in SOS. At least one goal state from each C_i zone is expanded, but depending on the tie-breaking policy, other states with the same costs may be expanded too.

3.1 Theoretical Analysis

We now prove that all MESs are required to be expanded to prove the optimality of solutions (task (*ii*)), and all NESs cannot possibly lead to optimal solutions and are thus irrelevant for achieving both task (*i*) and task (*ii*).

Lemma 1. *Given a problem instance $I \in I_{CON}$, any admissible MOS Algorithm A must expand all MESs (Def. 1).*

Proof. Let $\mathcal{POC}_I$, be the set of POF path costs of instance I. We need to show that every state s for which there exists a Pareto-optimal path U from $start$ to s and there exists $\mathbf{p} \in \mathcal{POC}_I$ s.t. $\mathbf{p} \not\preceq \mathbf{c}(U) + \mathbf{h}(s)$, then s must be expanded by Algorithm A. We prove the contrapositive. Suppose that (*i*) $I_1 = \langle G_1 = \langle V, E, \mathbf{c_1} \rangle, start, goal, \mathbf{h} \rangle$ is a problem instance in I_{CON} whose POF has a set of costs $\mathcal{POC}_{I_1}$, (*ii*) U is a Pareto-optimal path from $start$ to s in I_1 such that $\mathbf{C}^* \not\preceq \mathbf{c_1}(U) + \mathbf{h}(s)$ for all $\mathbf{C}^* \in \mathcal{POC}_{I_1}$, (i.e., $\mathbf{c_1}(U) + \mathbf{h}(s)$ is in Zone $A \cup B$), and (*iii*) A is an admissible DXBB algorithm that returns a POF for I_1, denoted $\mathsf{A}(I_1)$ (with cost vectors $\mathcal{POC}_{I_1}$), without expanding s. Then, construct a new instance $I_2 \in I_{AD}$ whose POF has a set of cost vectors $\mathcal{POC}_{I_2}$, such that (*i*) there exists $C \in \mathcal{POC}_{I_2}$ for which $C \not\preceq C'$ for all $C' \in \mathcal{POC}_{I_1}$, and (*ii*) A also returns $\mathsf{A}(I_1)$ (that contains no solution of cost C), thereby showing that A is not admissible on I_2. $I_2 = \langle G_2 = \langle V, E \cup \{e = \langle s, goal \rangle\}, \mathbf{c_2} \rangle, start, goal \rangle$. G_2 is copied from G_1, with the following changes: (*i*) an edge from s to $goal$ is added (if it didn't exist before) and (*ii*) cost $\mathbf{c_2}$ is identical to $\mathbf{c_1}$ with a single modification, $\mathbf{c_2}(e) = \mathbf{h}(s)$. Edge e from s to $goal$ generates a solution path $U' = [start, \ldots, s, goal]$ whose total cost is $\mathbf{c_2}(U') = \mathbf{c_1}(U) + \mathbf{h}(s)$, which is undominated by any cost in $\mathcal{POC}_{I_1}$ (by assumption (*ii*)). Edge e is thus an essential part of any POF of I_2. Since A is DXBB, it will behave on I_2 similarly to its behavior on I_1. In particular, A will not expand s, will not discover edge e, and will incorrectly return $A(I_1)$ as the POF for I_2. Hence, if $I_2 \in I_{AD}$, A is not admissible.

To complete the proof we now show that $I_2 \in I_{AD}$, i.e., that $\mathbf{h}$ is admissible on G_2 (even after edge e was added). Let x be an arbitrary state in G_2 and let W be any acyclic path in G_2 from x to $goal$. We need to prove that $\mathbf{h}(x) \preceq \mathbf{c_2}(W)$. If W does not contain the new edge e, the claim trivially follows from the admissibility of $\mathbf{h}$ on G_1. Now, suppose W does contain e. In other words, $W = \{X, e\}$ for some path X from x to s (i.e., e is the last edge of W). Now, since X is a path from x to s in G_1 and $\mathbf{h}$ is consistent on G_1 we have:

$$\mathbf{h}(x) \preceq \mathbf{c_1}(X) + \mathbf{h}(s)$$

By definition, $\mathbf{c_2}(e) = \mathbf{h}(s)$ and $\mathbf{c_1}(X) = \mathbf{c_2}(X)$, thus:

$$= \mathbf{c_2}(X) + \mathbf{c_2}(e) = \mathbf{c_2}(W)$$

Therefore, $\mathbf{h}$ is admissible on G_2, i.e., $I_2 \in I_{AD}$. □

We next show that NESs should never be expanded by admissible search algorithms.

Lemma 2. *Given an instance $I \in I_{AD}$, every NES (Def. 2) cannot lead to any path with cost* $\mathbf{p} \in \mathcal{POC}_I$.

Proof. Assume by contradiction that (*i*) there exists a problem instance I such that the costs of all paths from $start$ to some state s in I are dominated by one of the values in $\mathcal{POC}_I$ and that (*ii*) s must be expanded by any admissible MOS Algorithm A in order to find a POF of I (denoted as $\mathsf{A}(I)$). Let $W = XY$ be an acyclic path from $start$ to $goal$ such that X is a path from $start$ to s and Y is a path from s to $goal$. Due to the admissibility of I, we know that $\mathbf{h}(s) \preceq \mathbf{c}(Y)$ for all paths Y from s to $goal$. Thus, we get:

$$\mathbf{c}(W) = \mathbf{c}(X) + \mathbf{c}(Y) \succeq \mathbf{c}(X) + \mathbf{h}(s)$$

and, due to the assumption:

$$\succeq \mathbf{p} \text{ for some } \mathbf{p} \in \mathcal{POC}_I$$

Thus, all paths W that pass through s are not a POF solution, and s should not be expanded. □

4 Moving from States to Nodes

Above, we classified *states*. Nonetheless, search algorithms typically operate in the context of *nodes*, which correspond to specific paths in the graph. In SOS with a consistent heuristic, node f-values are *monotonically non-decreasing* along paths. For states with multiple paths, A^* uses *duplicate detection*, associating each state s with a unique node $n(s)$ representing

Algorithm 1: MOS-A*

```
   Input : G = ⟨S, E, c⟩, start, goal, h, OF, TB
1  n_root ← new node at start with g(n_root) = 0,
     h(n_root) = h(start), and parent(n_root) = null
2  initialize OPEN with OF and TB
3  OPEN.insert(n_root); POF ← ∅
4  while OPEN ≠ ∅ do
5      n ← OPEN.pop() // using OF and TB
6      if dom_check_exp(n) then continue
7      if s(n) = goal then
8          add n to POF and continue // line 4
9      for s' ∈ S s.t. (s(n), s') ∈ E do
10         n' ← new node at s' with h(n') = h(s'),
             g(n') = g(n) + c((s, s')), parent(n') = n
11         if dom_check_gen(n') then continue // line 9
12         OPEN.insert(n')
13 return POF
```

the cost-minimal path discovered for s, while pruning the remaining paths. Thus, choosing to expand a node $n(s)$ with minimal f-value in OPEN ensures that $n(s)$ represents an optimal path in the graph from $start$ to s [Dechter and Pearl, 1985]. These properties allow the adaptation of the state classification to node classification in SOS. Given a heuristic function h, A* (*i*) must expand all nodes n with $f(n) < C^*$, (*ii*) should never expand a node with n with $f(n) > C^*$ and (*iii*) might expand some nodes with $f(n) = C^*$.

In A* all nodes in OPEN with the minimal f-value are said to be at the *front* of OPEN and A* must choose for expansion a node from the *front*. Since there could be several nodes that have the same minimal f-value, a tie-breaking policy (TB) is applied to choose a specific node among them.

To make a similar transition from states to nodes in MOS, we first generalize A* to MOS-A* (applicable for MOS).

4.1 Multi-Objective Search A* (MOS-A*)

In MOS-A*, OPEN nodes are associated with **f**-values, which are d-dimensional vectors of values. Thus, in MOS-A*, the *front* of OPEN includes all nodes with *undominated* (but may be equal, $\not\prec$ relation) **f**-vectors. *front* can include multiple **f**-vectors (e.g., $(1, 2)$ and $(2, 1)$), each can be used for choosing nodes for expansion. Thus, to choose a node from the *front* of OPEN, we have two phases. The first phase is unique to MOS (and does not exist in SOS) and is called an *Ordering Function* (OF) [Skyler *et al.*, 2022]. An OF defines a total order between the different **f**-values. The cluster of nodes with the minimal order (according to the OF) is first chosen. A common OF is the *lexicographic order*, used by many algorithms, because it has a significant advantage in its CPU time overhead and in its data structure used [Hernández *et al.*, 2023b]. We discuss and compare this and other OFs below. The second phase uses TB within the chosen cluster (if it has more than one node) to choose the final node to expand.

The pseudocode for the MOS-A* framework is shown in Algorithm 1. It accepts a MOS instance as well as an OF and a TB. Each node n contains a state $s(n)$, and a **g**-value vector $\mathbf{g}(n)$. $\mathbf{f}(n)$ is then calculated as $\mathbf{f}(n) = \mathbf{g}(n) + \mathbf{h}(s(n))$. The *front* of OPEN includes all nodes whose $\mathbf{f}(n)$-values are undominated (but may be equal, $\not\prec$ relation). In each iteration, MOS-A* extracts a node n with the *"best"* **f**-value from the *front* of OPEN, as determined by OF and TB (Line 5). MOS-A* then performs the common expansion cycle and either adds a node to POF if $s(n) = goal$ (Lines 7-8) or expands it (Lines 9-12). When expanding a node n, MOS-A* generates a child node for each successor s' of $s(n)$. When OPEN becomes empty, it terminates and returns the POF.

4.2 Dominance Checks in MOS-A*

MOS-A* performs *dominance checks* for node n to determine whether n has the potential to be included in POF. Otherwise, n is pruned if it does not pass the check. Two types of checks need to be performed on node n.

The first is called *path dominance check*. Here, we compare different occurrences of $s(n)$ and prune those occurrences whose path costs are weakly dominated by costs of other occurrences. After this, only undominated nodes are left in OPEN and CLOSED. This is a generalization of *duplicate detection* in SOS, where only one node is kept for each state with minimal path cost. Similarly to SOS, this step is optional — One may choose not to implement it and risk a large increase in the size of OPEN. This directly translates into CPU time and memory increase. The tradeoff is that path dominance check requires tailored data structures.

The second type is called *solution dominance check*. Here, we prune node n if $\mathbf{f}(n)$ is dominated by a solution that was already found. This dominance check is mandatory. Without it, the algorithm can run forever and continue to produce solutions. This check is not performed in SOS because the search usually halts after finding the first solution. But, it is equivalent to pruning nodes with $f > C^*$, for example, if the task of the search is to return all optimal solutions.

In MOS-A* *dominance checks* can be performed in two locations: after a node was chosen for expansion (Line 6) or after a node was generated (Line 11) but before it is inserted into OPEN. Many algorithms use this framework (e.g., MOA* [Stewart and White III, 1991], NAMOA* [Mandow and De La Cruz, 2005], NAMOA*-dr [Pulido *et al.*, 2015], and BOA* [Hernández *et al.*, 2023a]). While they all use lexicographic OF, they mainly differ in whether, when, and how dominance checks are implemented as well as what data structures are used for maintaining OPEN. These differences significantly impact the constant time per node expansion, leading to substantial speedup possibilities in CPU time. In our analysis, we focus solely on understanding the impact of various OFs and TBs on the number of node expansions, excluding considerations of CPU runtime.

We now show how to adapt the classification of states to the classification of nodes for this family of MOS-A* algorithms.

4.3 Node Classification in MOS-A*

In MOS-A*, any state s is represented by a set of nodes $N(s)$, where each node $n \in N(s)$ corresponds to an undominated path from $start$ to s. Similarly to SOS, it was shown that given a consistent heuristic, the **f**-value of nodes is monotonically non-decreasing [Stewart and White III, 1991] along any given path. Thus, only expanding a node $n \in N(s)$ for which $\mathbf{f}(n)$ is undominated by any other **f**-values in OPEN ensures

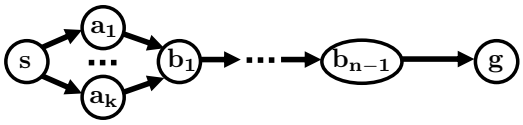


Figure 2: Information propagation back from the goal.

that n represents a Pareto-optimal path from $start$ to s. Consequently, the classification of states can be adapted to nodes, as we now show for MOS in general, but only illustrate for the special case of BOS in Figure 1.

Definition 4 (Must-Expand Nodes (MENs))**.** *Given a problem instance $I \in I_{CON}$, a node n is a MEN if $\boldsymbol{f}(n)$ is undominated by any cost in any POF of instance I, $\mathcal{POC}_I$, i.e., $\forall \mathbf{p} \in \mathcal{POC}_I$ it holds that $\mathbf{p} \not\preceq \boldsymbol{f}(n)$.*

Here too, the definition is split into two subcategories:

Definition 4.1 (Optimality Nodes). *Optimality nodes n dominate one solution (or more) from any POF, i.e., $\exists \mathbf{p} \in \mathcal{POC}$ such that $\boldsymbol{f}(n) \prec \mathbf{p}$.*

Optimality nodes (Zone A in Figure 1) are analogous to the MENs in SOS (with $f(n) < C^*$) and must be expanded to ensure the optimality of solutions in any POF.

Definition 4.2 (Completeness Nodes). *Completeness nodes n are mutually undominated by any solution in any POF, i.e., $\forall \mathbf{p} \in \mathcal{POC}$ it holds that $\boldsymbol{f}(n) \not\preceq \mathbf{p}$ and $\mathbf{p} \not\preceq \boldsymbol{f}(n)$.*

Completeness nodes (Zone B in Figure 1) have no analogy in SOS and have to be expanded to ensure there are no more solutions in any POF (i.e., that the POF is complete).

Definition 5 (Never-Expand Nodes (NENs))**.** *NENs n are dominated by at least one path in any POF, i.e., $\exists \mathbf{p} \in \mathcal{POC}$, such that $\mathbf{p} \prec \boldsymbol{f}(n)$.*

NENs are analogous to nodes with $f(n) > C^*$ in SOS and belong to Zone D which also includes the (dashed) border with Zone B in Figure 1. We note that the definition of MENs, albeit without the subdivision into two categories, and NENs is similar to the definitions presented by MPC10.

Definition 6 (Maybe-Expand Nodes (MBENs))**.** *MBENs are nodes n that are neither MENs nor NENs, i.e, $\exists \mathbf{p} \in \mathcal{POC}$ such that $\boldsymbol{f}(n) = \mathbf{p}$.*

All nodes in C points in Figure 1 are MBENs, which are analogous to nodes with $f(n) = C^*$ in SOS. The proofs of Lemmas 1 (for MES) and 2 (for NES) can be directly adapted for nodes when using MOS-A* and the above definitions.

Lemma 3. *Any MOS-A* algorithm expands all MENs and does not expand NENs.*

Proof. Since MOS-A* algorithms expand only nodes with undominated **f**-values and prune only dominated nodes, they are guaranteed to expand all MENs. We thus need to prove that MOS-A* algorithms do not expand NENs. Assume by contradiction that a MOS-A* Algorithm A expands a NEN n. Since n is a NEN, there exists a Pareto-optimal solution π such that $\pi \prec \mathbf{f}(n)$. Since MOS-A* algorithms expand only nodes with undominated **f**-values, π would be discovered before n is expanded. Thus, n will be pruned during dominance check, in contradiction to n being expanded by A. □

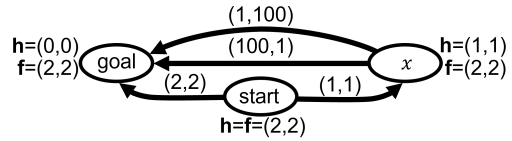


Figure 3: Tie-breaking example in MOS-BF*.

4.4 State-Based vs. Node-Based Perspectives

In the context of MOS, the definitions of Must-Expand (ME), Maybe-Expand (MBE), and Never-Expand (NE) are inherently different for nodes and states. With a consistent heuristic, the number of states expanded is bounded by the size of the graph. By contrast, the number of nodes representing cost-unique paths can be exponential in the number of states. Generally, the size of the Pareto-optimal frontier can be exponential in the number of states and thus counting states expanded may significantly *underestimate* the time and memory required by the algorithm. By contrast, considering node expansions can be too restrictive, potentially *overestimating* the resources needed by the search as we now demonstrate using Figure 2. There are k states, $a_1, \ldots, a_k$, connected to s (start). Each of these k states is further linked to a series of states labeled $b_1, \ldots, b_n = g$, forming k Pareto-optimal paths: $s, a_i, b_1, \ldots, n_{n-1}, g$, for $1 \leq i \leq k$. In currently existing MOS-A* algorithms, the search is carried out through distinct search nodes (paths) independently. Thus, each node a_i ($1 \leq i \leq k$) will entail a distinct expansion of state $b_1, \ldots, g$ (forming a distinct node for each of them). In this context, the definitions of MENs and NENs hold significance.

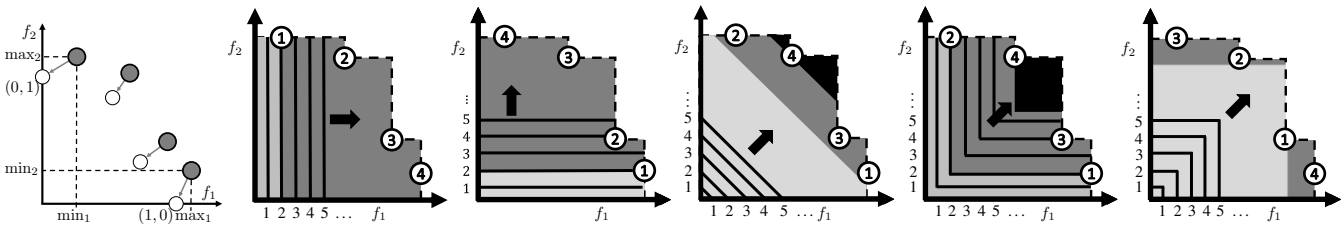
However, under the DXBB assumption, an algorithm A may first expand $s, a_1, b_1, \ldots, g$ and discover the first Pareto-optimal path. A can then propagate information, e.g., the path and its costs backward from g to b_1. Since the expansion function is defined on states (and not paths), when a_2 is expanded, the path from b_1 to the goal is already known. As a result, $b_1, \ldots, b_n$ need not be expanded again. This also applies to the expansion of $a_3, \ldots, a_k$. In these scenarios, where certain states need not be re-expanded, MENs and NENs may overestimate the number of expanded nodes where as MESs and NESs remain valid.

We note that algorithms with such backpropagation and information sharing do not yet exist and are the focus of an active research effort. We thus next examine the influence of TBs and OFs in the context of MEN, MBEN, and NEN.

4.5 Role of Tie-Breaking Policies in MOS-A*

Since MOS-A* must expand all MENs and never expands any NEN, OFs and TBs can only affect which of the MBENs are expanded during the search. We discuss these next.

TB policies determine which node to expand, among all nodes with the same **f**-value in OPEN. Consequently, different TBs can result in different nodes being expanded, even when the same OF is used. An example of the TBs effect on the search is shown in Figure 3. There are three states, $start$, x, and $goal$. Any MOS-A* must expand $start$ first (with any OF), generating nodes x and $goal$ (both have the same $\mathbf{f} = (2, 2)$). Thus, the order by which they are expended is solely determined by the TB. Consider the following two



(a) Normalization (b) *Lex1* (c) *Lex2* (d) *Avg* (e) *Min* (f) *Max*

Figure 4: The progress of different ordering functions. The three phases are colored differently.

TBs: $p_{h\uparrow}$ which expand nodes in increasing order of their sum of **h**-values, and $p_{h\downarrow}$ which expand nodes in decreasing order of their sum of **h**-values. Using $p_{h\uparrow}$, $goal$ is expanded first (after $start$), finding a solution of cost $(2, 2)$. In this case, the algorithm can terminate without expanding the node of x, as a solution that weakly dominates this node has been found (using the solution dominance check in Line 7 of Algorithm 1). By contrast, $p_{h\downarrow}$ expands x and only then expands $goal$, resulting in three node expansions instead of two.

4.6 Role of Ordering Functions in MOS-A*

Different TBs can result in different node expansions. We now show that using different ordering functions (OFs) with the same TB results in the *same* set of node expansions.

Lemma 4. *Let O_1 and O_2 be two OFs and let tb be a deterministic TB. A MOS-A* Algorithm A that expands nodes according to O_1 and tb will expand the same set of nodes on problem instance $I \in I_{CON}$ when using O_2 and tb.*

Proof. We prove the contrapositive. Let n be the first node expanded by A using O_1 and not expanded using O_2. Since A is admissible, then n is not a MEN. Otherwise, A must expand n even when using O_2. In addition, n is not a NEN (Lemma 3). Thus, n is a MBEN and $\mathbf{f}(n) \in \mathcal{POC}$. A could avoid expanding n only by finding a solution of cost $\mathbf{f}(n)$ that does not pass through n. That solution must be discovered by O_2 by expanding some other node n' that is not expanded by O_1, with $\mathbf{f}(n') \preceq \mathbf{f}(n)$ (due to admissibility). However, if $\mathbf{f}(n') \prec \mathbf{f}(n)$, n' would have been expanded before n when using O_1 and n would have been pruned due to the solution dominance check, contradicting the fact that n was expanded using O_1. Thus, $\mathbf{f}(n') = \mathbf{f}(n)$. Since n is the first node not expanded by A using O_2, both n and n' must have been in OPEN before either is expanded. The same deterministic tie breaking tb is used in both runs. Thus, the order by which n and n' are expanded is identical in the two runs, in contradiction to n' being expanded before n when using O_1. □

5 Ordering Functions

While ordering functions (OFs) do not change the final set of nodes expanded, they can affect the *order* by which these nodes are expanded and thus the order and rate by which solutions are discovered. This may be crucial in *anytime scenarios*, where one needs many solutions as fast as possible.

Frequently, the objectives are measured on different scales, e.g., time in seconds and distance in miles. Thus, to compare them meaningfully by OFs, we normalize them to be on the same $[0, 1]$ scale. Skyler *et al.* (2022) introduced a normalization scheme for BOS, which can also be used for MOS as we now describe. Let $\min_i = \min_{\mathbf{c} \in \mathcal{POC}} c_i$ and $\max_i = \max_{\mathbf{c} \in \mathcal{POC}} c_i$. All the values in range $[\min_i, \max_i]$ are then linearly projected into range $[0, 1]$. This is achieved by the following projection function applied on any value x_i: $\bar{x}_i = \frac{x_i - \min_i}{\max_i - \min_i}$ This normalization is illustrated for BOS in Figure 4a, where the gray and white circles are the solutions before and after the normalization, respectively.

For BOS, $\min_i$ and $\max_i$ values are calculated by running a SOS algorithm for objective i until *all* optimal solutions are found. $\min_i$ is set to the cost of the optimal solution. $\max_j$ (for the other dimension $i \neq j$) is set to be the maximal value of objective j in all optimal solution costs for objective i. This procedure can only be done for BOS (with $d = 2$). In MOS with $d > 2$, this is not possible. For example, consider an instance where $\mathcal{POC} = \{(2, 4, 10), (3, 3, 100), (4, 2, 10)\}$. Applying this method (i.e., running A* for each objective) would only find $\{(2, 4, 10), (4, 2, 10)\}$. The max_i vector will be $(4, 4, 10)$ instead of $(4, 4, 100)$. So, in practice, one can take any upper (lower) bound on the maximal (minimal) solution for $\max_i$ ($\min_i$). Then, the normalized value $\bar{\mathbf{x}}$ can be calculated based on these bounds using the normalization.

We next describe several OFs and analyze their behavior. Figure 4 illustrates the progress of the different OFs on the search in the bi-objective search (BOS) setting. The search is divided into three phases: (1) The First phase – nodes that are expanded before the first solution is found (the light grey area); (2) The Mid phase – nodes that are expanded after finding the first solution but before finding the last solution (the dark grey area); and (3) The Last phase – nodes that are expended after the last solution was found (the black area). The arrows indicate the order by which nodes are expanded.

Lexicographical Ordering (*Lex*). Given a MOS with d objectives, there are $d!$ *Lex* orderings as follows. Let $\mathbf{p} = p_1, \ldots, p_d$ be a permutation of $[1, \ldots, d]$. Then, the total order of the **f**-values is based on the *lexicographical order* induced by $\mathbf{p}$ (i.e., first compare $\mathbf{f}_{p_1}$, in the case of a tie, compare $\mathbf{f}_{p_2}$, and so on). Figures 4b and 4c show *Lex1* and *Lex2* (resp.) for BOS. *Lex1* finds the top-left solution first while *Lex2* finds the bottom right solution first.

(Weighted) Average Ordering (*Avg*). Assume two nodes n and m with f-values $\mathbf{f}(n) = [f_{n_1}, f_{n_2}, \ldots, f_{n_d}]$ and $\mathbf{f}(m) = [f_{m_1}, f_{m_2}, \ldots, f_{m_d}]$, respectively. Let $\mathbf{w}$ be a weight vector. *Avg* is defined as follows:

	$d=2$			$d=3$			$d=4$		
Phase	First	Mid	Last	First	Mid	Last	First	Mid	Last
Lex1	17.2	82.8	0.0	12.7	87.3	0.0	9.9	90.1	0.0
Lex2	17.1	82.9	0.0	12.1	87.9	0.0	9.5	90.1	0.4
Avg	47.1	51.4	1.5	25.1	74.6	0.3	19.6	80.1	0.3
Min	27.9	68.6	3.5	14.6	82.3	3.1	11.0	86.5	2.5
Max	43.1	56.9	0.0	28.5	71.5	0.0	20.2	79.8	0.0

Table 1: Percentage of expansions in each search stage.

$$Avg(n,m) = \arg\min\left(\sum_{i=1}^{d} w_i \cdot f_{n_i}\ ,\ \sum_{i=1}^{d} w_i \cdot f_{m_i}\right)$$

If $\mathbf{w} = [1,\ldots,1]$, then it is considered to be the *sum OF*. A similar order would result from $\mathbf{w} = [1/d,\ldots,1/d]$, which is the mathematical *average* of the different f_i-values. Figure 4d illustrates *Avg*. The **f**-values of expanded nodes form a diagonal line that proceeds along the direction of the black arrow. Assuming the **f**-values are normalized, the slope of this diagonal line is defined by **w**.

Maximum (Minimum) Ordering (*Max*, Resp. *Min*). First, *Max* (*Min*) orders the (normalized) objectives of each node in decreasing (increasing) order. Then, the ordered objectives are compared lexicographically. Figures 4f and 4e present *Max* and *Min* OFs, respectively.

6 Empirical Evaluation

We evaluated the expansions of each OF in the different expansion phases defined above on 200 random instances of the BAY road-map [DIMACS, 2006] with 2–4 objectives: time, distance, money, and uniform cost (respectively).

6.1 Expansions in the Three Phases

Table 1 shows the ratio (in %) of expansions in each of the three phases compared to the entire run. For $d = 2$, in the First phase, as expected, *Lex1* and *Lex2* have a small percentage. This is because they find the (lexicographically) first solution with a perfect heuristic in the first dimension. *Min* alternates between *Lex1* and *Lex2*, and finds the first solution after expanding slightly less than the sum of the two *Lex* orderings as it expands their intersection only once. Finally, *Avg* and *Max* expanded almost 50% of the nodes in the First phase. The Mid phase is the heaviest phase as it finds *all* solutions except the first one. In the Last phase, the nodes are expanded to prove that there are no more solutions in the POF. *Min* expanded 3.5% of the nodes after the last solution was found (black zone in Figure 4). This means that *Min* found the entire POF the fastest, as all functions expanded the same set of nodes in each instance. Importantly, *Lex* and *Max* do not expand any node in the Last phase, and the last solution found is the last node expanded for these OFs. The rest columns present these measurements for $d = 3$ and $d = 4$. The same trends are shown, but the relative size of the Mid phase becomes larger because there are more solutions to find.

6.2 Ratio of Solution Found

Figure 5 presents another view of this experiment by showing the average ratio of solutions found from the POF (y-axis) as a function of the progress of the expansions (x-axis), measured in percentiles for BOS problems ($d = 2$). There is a curve for each of the OFs described above (*Lex1*, *Lex2*, *Min*, *Avg*, *Max*). Being close to the top left corner is desirable because this indicates that many solutions are found early during the search. The results show that on average, *Min* was earliest in finding larger portion of the POF, then *Lex1* and *Lex2* while *Max* and *Avg* were the slowest. This is reasonable as many completeness nodes (Zone B) are often located in the center of the POF (can also be seen in Figure 1), which is the last area *Min* explores. Additionally, the dots at the top of the figure present the maximal value (among the different runs) that the OF achieved for each percentile ($Lex_i(max)$, *Min*(max), $Avg(max)$, $Max(max)$). The maximum values correlate with the average results. As mentioned, the last node explored by *Lex* and *Max* is a solution. Therefore, these functions reach 100% of the solutions in the end, after expanding all 100% of the nodes. By contrast, there was an instance in which *Avg* found all POF after 73% of the expansions, and *Min* found the entire POF after only 43% of the expansions. To summarize, while *Lex* reaches the first solution the fastest, other functions (*Avg* and *Min*) are able to find the entire POF faster, with fewer expansions. The same trends were seen for $d = 3$ and $d = 4$. These results are reported in the supplementary material.

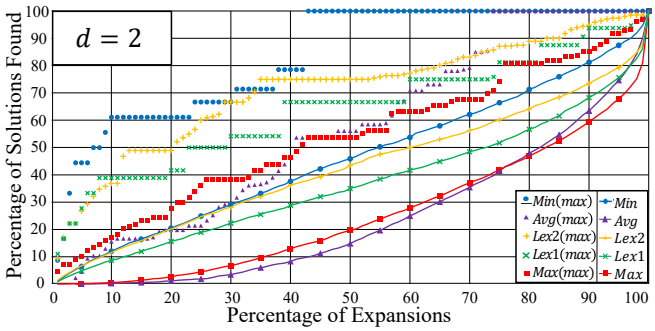


Figure 5: Solutions found vs. percentage of expansions.

Note that the different ordering, together with the compatible implementation of dominance checks, may significantly reduce the CPU time per node expansion and the overall running time. Improvements of up to a few orders of magnitude were observed by the method that use *Lex* and the dimensionality reduction that is connected to it [Hernández *et al.*, 2023a]. These dimensionality reductions are also possible for *Min*, *Max*, and *Avg*, and are the matter of current ongoing but orthogonal work. Nevertheless, in this paper we solely focus the number of nodes expanded. We leave the discussion on actual CPU time which is a function of the constant time per node of the different implementations to a different paper.

7 Conclusions

We analyzed expansions in MOS, classifying them to Must-Expand, Maybe-Expand, and Never-Expand, both in the context of states and nodes. In addition, we considered the issue of *OFs*, which are used by MOS-A* algorithms to decide which node to expand next based on their **f**-values. We presented several OFs and compared them experimentally, showing the rate OFs find solutions. As all OFs must expand the same set of nodes, we showed the benefit of some OFs with regard to the order in which these nodes are explored.

Acknowledgements

The research was supported by the United States-Israel Binational Science Foundation (BSF) under grant number 2021643 and by Israel Science Foundation (ISF) under grant number 909/23. The research at the University of Southern California was also supported by the National Science Foundation (NSF) under grant numbers 1817189, 1837779, 1935712, 2121028, 2112533, and 2321786. Finaly, the research was also supported by the National Center for Artificial Intelligence (CENIA), Financiamiento Basal ANID, and the Centro Ciencia & Vida under grant numbers FB210017, FB210008. 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 any government.

References

[Ahmadi *et al.*, 2021] Saman Ahmadi, Guido Tack, Daniel Harabor, and Philip Kilby. Bi-Objective Search with Bi-Directional A^*. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs*, pages 3:1–3:15, 2021.

[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.

[Clímaco and Pascoal, 2012] J. Clímaco and M. Pascoal. Multicriteria Path and Tree Problems: Discussion on Exact Algorithms and Applications. *ITOR*, 19(1-2):63–98, 2012.

[Current and Marsh, 1993] J. Current and M. Marsh. Multiobjective Transportation Network Design and Routing Problems: Taxonomy and Annotation. *EJOR*, 65(1):4–19, 1993.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized Best-First Search Strategies and the Optimality of A^*. *J. ACM*, 32(3):505–536, 1985.

[DIMACS, 2006] DIMACS. The 9th DIMACS Implementation Challenge: Shortest Path, 2006.

[Eckerle *et al.*, 2017] Jürgen Eckerle, Jingwei Chen, Nathan R. Sturtevant, Sandra Zilles, and Robert C. Holte. Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search. In *ICAPS*, pages 79–87, 2017.

[Fu *et al.*, 2019] Mengyu Fu, Alan Kuntz, Oren Salzman, and Ron Alterovitz. Toward Asymptotically-Optimal Inspection Planning Via Efficient Near-Optimal Graph Search. In *Robotics: Science and Systems*, 2019.

[Fu *et al.*, 2021] Mengyu Fu, Oren Salzman, and Ron Alterovitz. Computationally-Efficient Roadmap-based Inspection Planning via Incremental Lazy Search. In *ICRA*, pages 7449–7456, 2021.

[Goldin and Salzman, 2021] Boris Goldin and Oren Salzman. Approximate Bi-Criteria Search by Efficient Representation of Subsets of the Pareto-Optimal Frontier. In *ICAPS*, pages 149–158, 2021.

[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, 1968.

[Hernández *et al.*, 2023a] Carlos Hernández, William Yeoh, Jorge Baier, Ariel Felner, Oren Salzman, Han Zhang, Shao-Hung Chan, and Sven Koenig. Multi-objective search via lazy and efficient dominance checks. In *IJCAI*, 2023.

[Hernández *et al.*, 2023b] Carlos Hernández, William Yeoh, Jorge A. Baier, Han Zhang, Luis Suazo, Sven Koenig, and Oren Salzman. Simple and efficient bi-objective search algorithms via fast dominance checks. *Artif. Intell.*, 314:103807, 2023.

[Mandow and De La Cruz, 2005] Lawrence Mandow and José Luis Pérez De La Cruz. A New Approach to Multi-Objective A^* Search. In *IJCAI*, pages 218–223, 2005.

[Mandow and Pérez-de-la-Cruz, 2010] Lawrence Mandow and José-Luis Pérez-de-la-Cruz. Multiobjective A^* search with consistent heuristics. *J. ACM*, 57(5):27:1–27:25, 2010.

[Pulido *et al.*, 2015] Francisco-Javier Pulido, Lawrence Mandow, and José-Luis Pérez-de-la Cruz. Dimensionality Reduction in Multiobjective Shortest Path Search. *CAR*, 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 *IJCAI*, pages 6759–6768, 2023.

[Skriver, 2000] A. J.V. Skriver. A Classification of Bicriterion Shortest Path (BSP) Algorithms. *Asia Pacific J. of Oper. Res.*, 17(2):199–212, 2000.

[Skyler *et al.*, 2022] S. Skyler, D. Atzmon, A. Felner, O. Salzman, H. Zhang, S. Koenig, W. Yeoh, and C. Hernández. Bounded-Cost Bi-Objective Heuristic Search. In *SoCS22*, pages 239–243, 2022.

[Stewart and White III, 1991] Bradley S Stewart and Chelsea C White III. Multiobjective A^*. *Journal of the ACM*, 38(4):775–814, 1991.

[Tarapata, 2007] Z. M. Tarapata. Selected Multicriteria Shortest Path Problems: An Analysis of Complexity, Models and Adaptation of Standard Algorithms. *IJAM*, 17(2), 2007.

[Ulungu and Teghem, 1991] EL Ulungu and J Teghem. Multi-Objective Shortest Path Problem: A Survey. In *Workshop on Multicriteria Decision Making*, pages 176–188, 1991.