Scalarizing Multi-Objective Robot Planning Problems Using Weighted Maximization

Nils Wilde , *Member, IEEE*, Stephen L. Smith , *Senior Member, IEEE*,
and Javier Alonso-Mora , *Senior Member, IEEE*

Abstract—When designing a motion planner for autonomous robots there are usually multiple objectives to be considered. However, a cost function that yields the desired trade-off between objectives is not easily obtainable. A common technique across many applications is to use a weighted sum of relevant objective functions and then carefully adapt the weights. However, this approach may not find all relevant trade-offs even in simple planning problems. Thus, we study an alternative method based on a weighted maximum of objectives. Such a cost function is more expressive than the weighted sum, and we show how it can be deployed in both continuous- and discrete-space motion planning problems. We propose a novel path planning algorithm for the proposed cost function and establish its correctness, and present heuristic adaptations that yield a practical runtime. In extensive simulation experiments, we demonstrate that the proposed cost function and algorithm are able to find a wider range of trade-offs between objectives (i.e., Pareto-optimal solutions) for various planning problems, showcasing its advantages in practice.

Index Terms—Optimization and optimal control, motion and path planning, multi-objective optimization, task and motion planning.

I. INTRODUCTION

AUTOMATED planning and decision making plays a central role in designing intelligent robotic systems. In many real-world settings, autonomous robots are faced with complex scenarios that require them to balance between different objectives simultaneously. For instance, autonomous vehicles need to navigate to a goal, while ensuring safety, passenger comfort and ideally fuel-efficiency [1], [2], [3]. Similarly, mobile robots navigating in human-centered spaces such as offices, hospitals or public areas need to consider task efficiency and conforming to social norms [4], [5].

In multi-objective optimization (MOO) problems – such as finding trajectories trading off different objectives – the optimal solution is usually not unique, but rather there is a set of *Pareto-optimal* solutions. A solution is Pareto-optimal when none of
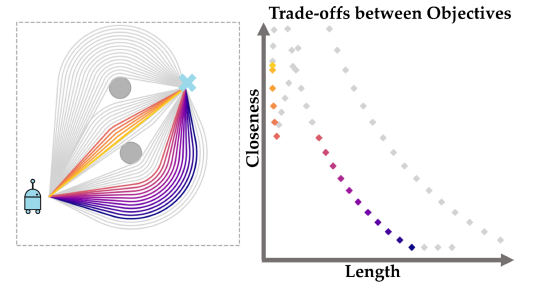
(a) Ground set of feasible trajectories. Pareto-optimal solutions are highlighted in colour.
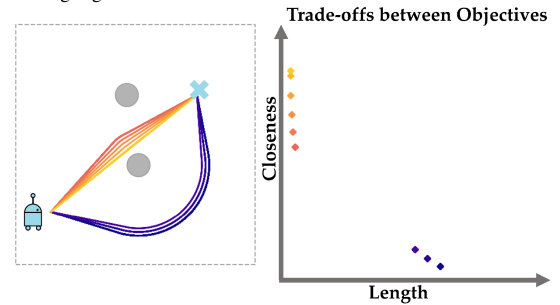
(b) Attainable solutions for weighted sum (WS) optimization.
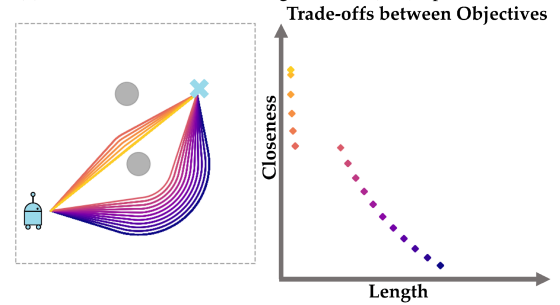
(c) Attainable solutions for the proposed weighted maximum (WM) optimization.

Fig. 1.   Simple planning problem with two objectives: trajectory length and the minimum distance to an obstacle. Shown are the ground set of trajectories (a), the solutions when using a weighted sum objective (b), and the solutions when using the proposed weighted maximum objective (c). In each subfigure the left plot shows the set of trajectories and the right plot shows the corresponding trade-offs. Colored trade-offs are Pareto-optimal, i.e., show the Pareto-front.

the individual objectives can be improved without worsening at least one other objective. Thus, an important challenge in motion planning remains the design of objective functions that balance between several potentially competing objectives and allow for computing Pareto-optimal solutions. A common approach is to formulate a weighted sum of the objective functions [2], [4], [6],

[7]. Often, the weights on the objectives are tuning-parameters, requiring careful calibration. In human-robot interaction (HRI) user preferences for robot behaviour is commonly modelled as a weighted sum of features [4], [8], [9], [10], [11], [12]. The linear structure allows for designing efficient algorithms for both motion planning and learning from human feedback making this approach very popular.

However, it can fail to describe all optimal trade-offs since it is unable to explore non-convex regions of Pareto-fronts. This issue persists even in simple planning problems, as illustrated in Fig. 1. Here we compute trajectories between a fixed start and goal position around two static obstacles. To optimize for trajectory length and the minimal distance to an obstacle we can inflate the obstacles and then plan paths on visibility graphs. Subplot 1a shows the ground set of feasible trajectories together with the corresponding Pareto-front. Subplot 1b shows the trajectories that can be computed with the weighted sum (WS) method for different weights. We observe that the trajectories found by the WS method are only a small subset of the Pareto-optimal solutions from 1a: While there are numerous trajectories available that pass between the two obstacles, there are only few that go around. It is important to notice that this is not due to the choice or resolution of the weights. Rather, *there does not exist any tuning of weights* such that the motion planner returns a more intermediate trade-off, since parts of the Pareto-front are non-convex [13]. This problem occurs when parts of the Pareto-front are non-convex: the solutions of the weighted sum method do only cover the convex hull of the Pareto-front [13].

We study an alternative form of scalar objective function, the *weighted maximum* (WM) of objectives, also known as Chebyshev scalarization [13]. This allows for finding a richer set of trade-offs between the two objectives as shown in subplot 1c, covering all parts of the Pareto-front. Indeed, this approach is able to find all Pareto-optimal solutions, i.e., is *Pareto-complete* [13]. Despite the theoretical foundations established in the optimization literature, more expressive scalarization methods such as the WM have not found much attention in robot motion planning. To demonstrate the potential of WM optimization in robot motion planning we discuss fundamental shortcomings of widely used WS cost functions, independent of how weights are selected. We revisit established results from optimization to describe theoretical differences between WS and WM cost functions: WM cost is a provably more expressive tool for motion planning, yet only requires the same number of parameters. We show how WM costs can be used in continuous and discrete space planning problems. For discrete planning, we consider a general monotonic utility function to combine objective values of discrete actions (*e.g.,* edges in graphs), allowing for complex planning.

A. Contributions

Our contributions are as follows: First, we consider continuous-space planning problems and show how existing optimization techniques can be used to solve planning problems with WM cost functions. Further, we show NP-hardness of graph based path planning with a WM cost. Second, we present a novel

optimal path planning algorithm for the WM cost and establish its correctness. Further, we show how our algorithm can be enhanced with a cost-to-go heuristic and discuss a budgeted suboptimal version that runs in polynomial time. Third, in a series of simulations, we demonstrate that the proposed WM method finds a substantially richer set of trade-offs in various motion planning problems, and showcase that the proposed graph search finds optimal solutions within a practical runtime.

B. Related Work

Many robot planning problems consider multiple, potentially competing objectives. The most prominent approach is to formulate a weighted sum of a given set of objective functions and then solve the resulting scalar optimization problem. This approach is used in trajectory planning for autonomous driving [1], [2], [3], [14], [15], [16], [17], local planning in cluttered environments [7], [18] or social spaces [19], [20], trajectory generation for manipulators [6], [21], and multi-robot planning [22], [23].

Researcher in HRI use weighted sums of objective functions – usually referred to as features – to model how users evaluate robot behaviour [8], [9], [10], [11], [12], [24], [25], [26]. In human-in-the-loop learning frameworks users provide feedback to a robot in form of demonstrations, choice, labels, critic, and others, allowing the robot to learn weights for the objective functions and thus adapt their behaviour to the user's preferences.

Different approaches that address the shortcomings of WS objectives include alternative scalar functions such as the WM approach, or incremental algorithms that iteratively explore the entire Pareto-front [13]. A popular approach that can explore non-convex Pareto-fronts is the Adapt Weighted Sum Method [27], [28]. Here solutions are sampled iteratively using equality constraints to force new samples to close gaps in the Pareto-front. Yet, this approach is solving a slightly different problem than we are addressing: The method iteratively creates a set of potential solutions that cover the Pareto-front. Thus, it does not have tuning parameters such that carefully choosing them allows for obtaining the desirable solution. Further, the approach does not come with a completeness guarantee and can get stuck when the Pareto-front is discontinuous. Lastly, satisfying the added equality constraints can be infeasible or computationally hard in practice, especially in discrete-space planning. In our work, we propose using a WM cost function which has tunable weights such that any Pareto-optimal solution can be attained for a specific weight vector. Further, we explicitly address the challenges in discrete space planning and propose a novel graph search for minimizing WM costs for different types of individual objective functions.

Overall, the limitations of WS costs find less discussion in the context of robot planning. Recently, [29] compared different scalar objective functions. Our work focuses on the WM cost function only, yet provides a theoretical analysis of its expressiveness and presents a novel algorithm for discrete space planning. Other works directly address MOO for specific problems. For instance, the authors of [30] studied weighted sum and weighted minimum approach for exploring Pareto-fronts of sampling based motion planning problems, while [31]

addresses the problem of simultaneously optimizing for path length and clearance in the plane, proposing a complete and efficient algorithm. WM cost functions also found attention in multi-objective Reinforcement Learning (RL) [32], [33]. The works of [34] and [35] study hierarchical frameworks based on a multi-objective Probabilistic Roadmap (MO-PRM) and explicitly consider two objectives: path length and risk [34], and path length and state-estimation error [35]. The MO-PRM separates objectives in primary and secondary costs, and then plans using a discretization of values for the secondary costs, similar to the ϵ-constrained method [13]. A drawback of that method is that it depends on the resolution of the constraint on the secondary cost and can require solving multiple optimization problems in order to verify Pareto-optimality. In contrast to these works, our paper does not address a specific multi-objective motion planning problem, but rather proposes an alternative to weighted sums for *any* collection of objective functions.

II. PROBLEM STATEMENT

In this section, we revisit some preliminary concepts before introducing our formal problem statement.

A. Preliminaries

1) Pareto-Optimality: Consider a multi-objective optimization problem where the domain is some vector space $\mathcal{X}$. We want to find a solution $\boldsymbol{x} \in \mathcal{X}$ that simultaneously minimizes n different functions, i.e., that solves $\min_{\boldsymbol{x}}\{f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\}$. In general, the solution to a MOO problem is not a unique vector $\boldsymbol{x}$, but a set of *Pareto-optimal* solutions. We briefly review the definitions of *dominated solutions* and the Pareto-front.

Definition 1 (Dominated solution): Given a MOO problem and two solutions $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. Vector $\boldsymbol{x}$ *dominates* $\boldsymbol{x}'$ when $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}')$ holds for all $i = 1, \ldots, n$, and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}')$ holds for at least one j where $1 \leq j \leq n$.

Definition 2 (Pareto-front): Given a MOO problem, the set of *Pareto-optimal* solutions – called *Pareto-front* – is the subset of all solution that are not *dominated* by another solution.

2) Graph Theory: Following [36], a graph is a tuple $G = (V, E)$ where V are vertices and E is a set of edges. In a weighted graph $G = (V, E, d)$ edges are associated with some cost $d : E \to \mathbb{R}$. A walk is a sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ such that $e_i = (v_i, v_{i+1}) \in E$ and $e_i \neq e_j$ for $i, j = 1, \ldots, k$. We define a path P as a sequence of vertices $(v_1, \ldots, v_{k+1})$ with no duplicate entries for which there exists a walk $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ in G.

B. Problem Formulation

We consider a planning problem described by a robot's state and action space $(\mathcal{X}, \mathcal{A})$, a start state x_s and a set of goal states $X_g \subset \mathcal{X}$. Let $\mathcal{T}$ be the set of all feasible trajectories starting at x_s and ending at some state $x_g \in X_g$. Note that the set $\mathcal{T}$ is typically defined implicitly as set of constraints on the robot's state and actions, such as kinodynamic constraints on motion, or spatial constraints for obstacle avoidance. We keep this set abstract at this point, but give specific examples in Section V.

To define the desired robot behaviour the designer of a motion planner considers a set objectives to be minimized. Let these objectivess be denoted by $f_1, \ldots, f_n$ where $f_i : \mathcal{T} \to \mathbb{R}_{\geq 0}$ for $i = 1, \ldots, n$. The optimal solution to the motion planning problem is some trajectory $T^* \in \mathcal{T}$. Assuming that the objectives $f_1, \ldots, f_n$ contain all aspects under consideration, T^* is a Pareto-optimal solution to the problem

$$\min_{T \in \mathcal{T}} \{f_1(T), \ldots, f_n(T)\}. \tag{1}$$

Let $\mathcal{T}' \subseteq \mathcal{T}$ denote the set of all Pareto-optimal solutions. Given above definitions, we can pose our main problem.

Problem 1 (Parametric single objective planning): Given state and action space $(\mathcal{X}, \mathcal{A})$, initial state x_s and goal states X_g, and objectives $f_1, \ldots, f_n$ find an algorithm such that, for any Pareto-optimal solution $T^* \in \mathcal{T}'$, there exists algorithm parameters for which the algorithm returns T^*.

Our approach to Problem 1 is writing (1) as a scalar function where tuning weights $\boldsymbol{w}$ define the balance between objectives. The scalar function needs to be solvable, and for any Pareto-optimal trajectory $T^* \in \mathcal{T}$ there exist a choice of weights such that T^* is the solution to the scalar optimization problem.

III. APPROACH

A common approach to tackle Problem 1 is solving the MOO problem from (1) via means of linear scalarization, also referred to as the weighted sum (WS) method or cost [13]. This yields the following cost function

$$c^{\text{sum}}(T) = \sum_{i=1}^{n} f_i(T) w_i = \boldsymbol{f}(T) \cdot \boldsymbol{w}, \tag{2}$$

where $\boldsymbol{w} \in [0, 1]^n$ is a vector of tunable weights. While this approach has been widely used and been proven to be effective, its simplicity limits the expressiveness. In this paper, we offer an alternative model-based approach. We propose a weighted maximum approach for a scalar cost functions, where the summation is replaced by taking the maximum:

$$c'(T) = \max_{i=1,\ldots,n} f_i(T) w_i. \tag{3}$$

The cost of a trajectory is now given by the objective that attains the largest value when multiplied by its weight. That is, a trajectory is evaluated only based on the most prominent weighted objective value. We notice that when the solution to (3) is unique, it is Pareto-optimal. However, if there are multiple solutions, then one is Pareto-optimal, while all others are only *weakly* Pareto-optimal [13].

In order to only attain Pareto-optimal solutions, we add $\rho \sum_{i=1}^{n} f_i(T)$ as a tie-break in the cost function, where $\rho > 0$ is a sufficiently small constant:

$$c^{\text{max}}(T) = \max_{i=1,\ldots,n} f_i(T) w_i + \rho \sum_{i=1}^{n} f_i(T). \tag{4}$$

We refer to this as the weighted maximum (WM), or *augmented Chebyshev problem* [13]. Next, we characterize its expressiveness compared to the WS. Given a planning problem

with the ground set of feasible trajectories $\mathcal{T}$, let $\mathcal{T}' \subseteq \mathcal{T}$ be the set of all Pareto-optimal trajectories. Further, let $\mathcal{T}^{\mathtt{sum}} \subseteq \mathcal{T}$ be the set of trajectories that are optimal for *some* weight in (2) and let $\mathcal{T}^{\mathtt{max}} \subseteq \mathcal{T}$ be the set of trajectories that are optimal for *some* weights in (4). In detail, we have $\mathcal{T}^{\mathtt{sum}} = \{T' \in \mathcal{T} \mid T' = \arg\min_T c^{\mathtt{sum}}(T), \, \boldsymbol{w} \in [0,1]^n\}$, and $\mathcal{T}^{\mathtt{max}} = \{T' \in \mathcal{T} \mid T' = \arg\min_T c^{\mathtt{max}}(T), \, \boldsymbol{w} \in [0,1]^n\}$. We revisit a known result:

Lemma 1 (Pareto-optimality of scalarization): For any planning problem $\mathcal{T}^{\mathtt{sum}} \subseteq \mathcal{T}^{\mathtt{max}} \subseteq \mathcal{T}'$.

A proof is omitted since this is a well established result in multi-objective optimization [13]. The lemma ensures that any solution to the two scalarized optimization methods is always a Pareto-optimal solution. However, a lesser known result is while all solutions to (2) are Pareto-optimal, there can exist Pareto-optimal solutions that are not a solution to (2) for any $\boldsymbol{w}$. Thus, the WS is less expressive than the WM.

Proposition 1 (Expressiveness): Given a planning problem where trajectories T are parametrized in $\mathbb{R}^m$, and auxiliary cost functions $\boldsymbol{f}(T) = [f_1(T) \, \ldots \, f_n(T)]$. If at least one of the cost functions $f_i(T)$ is not a proper convex[1] and continuous function over $\mathbb{R}^m$ then $\mathcal{T}^{\mathtt{sum}} \subset \mathcal{T}^{\mathtt{max}}$, i.e., optimizing (2) is strictly less expressive than optimizing (4).

The proof follows directly from Theorem 6.3 and Remark 6.4 in [37]: Linear scalarization is only Pareto-complete when the costs are proper convex and continuous. Thus, when that condition is violated we have $\mathcal{T}^{\mathtt{sum}} \subset \mathcal{T}'$. In contrast, the WM is Pareto-complete [13], i.e., $\mathcal{T}^{\mathtt{max}} = \mathcal{T}'$ always holds.

The effect of Proposition 1 becomes apparent in Fig. 1. When there is more than one homotopy class for navigating around obstacles, the objective for minimizing closeness to obstacles becomes non-convex. As a consequence, the Pareto-front is non-convex. The WS method is then only able to find solutions lying on the convex hull of the Pareto-front, while the WM method finds solutions in all parts of the Pareto-front.

IV. MOTION PLANNING WITH WEIGHTED MAXIMUM COST

We now consider the problem of finding an optimal trajectory for the proposed WM cost for given weights $\boldsymbol{w}$. Thus, we study how the WM cost can be used in continuous space motion planners such as Model-Predictive Control (MPC), and in discrete, graph-based planners such as state-lattices.

A. Continuous Space Planning

We consider a discrete time, continuous space planning problem to find an optimal trajectory T, subject to kinodynamical constraints $g(T) \leq 0$. When minimizing the proposed WM cost, the problem may be written as

$$\min_T \, \max_i f_i(T)w_i + \rho \sum_{i=1}^n f_i(T)$$
$$s.t. \, g(T) \leq 0. \tag{5}$$

[1]A convex function $f(x)$ is proper convex over its domain $\mathcal{X}$ if it never attains $-\infty$, and if there exists at least some x_0 where $f(x_0) < \infty$.

Following [38], this can be reformulated as follows

$$\min_T \, t + \rho \sum_{i=1}^n f_i(T)$$
$$s.t. \, \max_i \, w_i f_i(T) \leq t \text{ for } i = 1, \ldots, n,$$
$$g(T) \leq 0. \tag{6}$$

The newly introduced max constraint can be written out as n constraints of the form $w_i f_i(T) \leq t$ for $i = 1, \ldots, n$. We observe that this removes the maximization from the problem, such that the objective and constraints are linear compositions of the individual objective f_i. In case constraints $g(T)$ are non-convex, solving for the weighted maximum does not make the problem fundamentally harder than optimizing for the weighted sum. However, the same does not hold for graph-based planners as we will show next.

B. Graph Based Planning

We now consider the WM cost for discrete space motion planners such as graph or lattice based methods and characterize the hardness of the problem. Let $G = (V, E)$ be a graph where we associate each edge $e \in E$ with non-negative and bounded trajectory costs $f_1'(e), \ldots, f_n'(e)$.

1) LP Formulation: We briefly consider the simple case where the costs of a path are the sum of the edge costs $f_i(P) = \sum_{e \in E(P)} f_i(e)$. We recall the linear program (LP) formulation of a shortest path problem, i.e., a path minimizing (2). Thus, the cost of an edge e is given by $\boldsymbol{f}(e) \cdot \boldsymbol{w}$, and the network flow constraints are summarized as $F(\boldsymbol{x}) \leq \boldsymbol{b}$. The well-known LP-formulation [39] then is

$$\min_{\boldsymbol{x}} \, \sum_{(v,u)} x_{vu} \cdot \boldsymbol{f}(e_{vu}) \cdot \boldsymbol{w}$$
$$s.t. \, F(\boldsymbol{x}) \leq \boldsymbol{b}. \tag{7}$$

Here $\boldsymbol{x}$ is a binary vector, where $x_{vu} = 1$ indicates that edge (v, u) is contained in the path. When now considering the WM cost, the objective is $\min_{\boldsymbol{x}} \max_i \, w_i \sum_{(v,u)} x_{vu} \cdot f_i'(e_{vu}) + \rho \sum_{i=1}^n f_i(e_{vu})$. In principle, we can apply the same reformulation technique as in (6) and still obtain an LP. However, the solution will not have integer values since the constraints are no longer totally-unimodular. Hence, the LP solution will not solve the shortest path problem [39].

2) Formal Problem Analysis: Given that the LP-formulation for shortest paths does not work for the WM cost, we study the problem of finding a path that minimizes (4) in more detail. We consider the general case where the costs of a path P are not necessarily the sum of the edge costs in the path. Instead, the cost for a path P with edges $E(P)$ is

$$f_i(P) = \beta \left(f_i'(e_1), f_i'(e_2), \ldots \right), \text{ where } e_1, e_2, \cdots \in E(P). \tag{8}$$

We refer to β as the composition function and assume that β is monotonically increasing. This captures two widely used concepts of defining costs over a robot's trajectory: i) summation

or integration over the trajectory to compute its length, time, integral square jerk, accumulated risk or similar costs, and ii) taking the maximum value over a trajectory such as the maximum jerk or maximum risk. Thus, we can state the problem of finding a path of minimal maximum weighted cost.

Problem 2 (Min-max cost path (MMCP)): Given a strongly connected graph $G = (V, E)$ with start and goal vertices s, g in V, edge cost functions $f'_1(e), \ldots, f'_n(e)$, a composition function β, and weights $w_1, \ldots, w_n$, find a path that solves

$$\min_P \max_i w_i f_i(P) + \rho \sum_{i=1}^n f_i(P). \tag{9}$$

The problem is closely related to the multi-objective shortest path (MOSP) problem, which is NP-hard for two or more objectives [40]. The main difference is that MOSP considers that β is taking the sum over different edge cost, which makes it a special case of Problem 2. We formally establish hardness of our problem.

Proposition 2 (Hardness of MMCP): The MMCP is NP-hard.

Proof: We consider the special case that $f_i(P) = \sum_{e \in E(P)} f_i(e)$. The decision version of MMCP decides if there exists a path such that the $\max_i w_i f_i(P) \le \alpha$ for some constant α. We can reduce MOSP to MMCP. MOSP takes as an input a graph $G = (V, E)$ and some cost functions $\gamma_1, \ldots, \gamma_n$ assigning costs to edges. The decision version answers if there exists a path P such that $\sum_{e \in E(P)} \gamma_i(e) \le \alpha$ for all i and some constant α. Given an instance of MOSP, we use the same graph as an input for MMCP, choose costs $f_i(e) = \gamma_i(e)$, and set $w_i = 1$ for $i = 1, \ldots, n$. The solution to MMCP then indicates if $\max_i \sum_{e \in E(P)} \gamma_i(e) \le \alpha$, which trivially also decides the MOSP instance. ∎

3) Algorithm Description: We now present a complete algorithm for MMCP, detailed in Algorithm 1. Our approach is a modification to Dijkstra's algorithm where we record all paths to a vertex, similar to the *Martin algorithm* for MOSP [41]. To that end, the elements in our open_list are tuples consisting of a cost, a vertex, and a path (i.e., a sequence of preceding vertices) from the start to this vertex. Similar to Dijkstra's our algorithm retrieves the lowest cost element from the open_list (line 3). We then expand the neighbouring vertices u (line 7) and ensure that the path to the neighbour is not in the open_list, is not dominated by another path to u, and does not contain cycles (lines 8-10). We then add the path to u with its WM cost to the open_list (line 12-13). Algorithm 1 is able to handle any monotonically increasing composition function β (see (8)) as opposed to the sum of edge costs considered in MOSP. Opposed to MOSP, we are only interested in finding one solution for a given weight instead of the set of all Pareto-optimal paths. Thus, Algorithm 1 terminates once the goal is reached (lines 4-5).

4) Theoretical Properties: First, we characterize the runtime. In the worst case, Algorithm 1 explores all paths from s to any vertex u, leading to $2^{|V|}$ (the size of the power set for all sequences of vertices) executions of the while loop. For each subpath, we compute its cost only once in line 12, which requires evaluating $f_i(e)$ for all its edges and all n objective functions. The number of edges is upper bounded by $|V|^2$.

Algorithm 1: Min-Max Cost Path.

Input: Graph $G = (V, E)$, start and goal s, g in V, cost functions on edges $\boldsymbol{f} = [f_1, \ldots, f_n]$, composition function β, weight $\boldsymbol{w}$

Output: Path from s to g with minimal WM cost

1   open_list $= \{(0, s, (s))\}$
2   **while** |open_list| > 0 **do**
3     $(cost, v, P) \leftarrow$ open_list.pop() // get by min cost
4   **if** $v = g$ **then**
5   **return** P
6   **for** u *in* Neighbours(v) **do**
7   Create tentative path $P^u = P \cup \{u\}$
8       $\mathcal{P} \leftarrow$ all paths in open_list that end at u
9   **if** *Any $P' \in \mathcal{P}$ dominates P^u or P^u contains cycles* **then**
10   **continue**
11   Delete tuples from open_list where P' is dominated by P^u
12   Comp. WM cost $c^{\texttt{max}}(P^u, \boldsymbol{w})$ // Using (4), (8)
13   Add $(c^{\texttt{max}}(P^u, \boldsymbol{w}), u, P^u)$ to open_list

14   **return** *Failure*

Assuming that the evaluation of the costs $f_i(e)$ takes constant time, the total runtime is $O(2^{|V|} \cdot |V|^2 \cdot n)$. While the runtime only grows linearly with the number of objective functions, it can scale exponentially with the number of vertices. However, due to the stopping criteria in line 4, the algorithm does not enumerate all $2^{|V|}$ solutions in practice. In our simulations we show that it is able to solve instances with $|V| = 2000$.

Next we will establish correctness of the algorithm. We begin by considering the subpath elimination in line 9.

Lemma 2 (Subpath elimination): Let P' and P^u be two subpaths from s to u such that P' dominates P^u. If the composition function β is monotone, then P^u cannot be part of an optimal path from s to the goal g.

Proof: If P' dominates P^u then $f_i(P') \le f_i(P^u)$ for all i. Now consider any possible path Q from u to the goal g. Since f_i is *monotone* we then have $f_i(P' \cup Q) \le f_i(P^u \cup Q)$. Finally, $c(P' \cup Q) \le c(P^u \cup Q)$ follows directly from (4). Hence, P^u cannot lead to a path to the goal of lower cost than P' and thus may be disregarded from the search. ∎

The result is similar to the analysis in [40], yet extends to the case of any monotone composition function β instead of only sums. Based on Lemma 2, we can ensure that our algorithm finds the optimal solution.

Proposition 3 (Correctness): For any given weight $\boldsymbol{w}$, Algorithm 1 returns the optimal solution $P^* = \arg\min_P \max_i f_i(P) \cdot w_i$.

Proof: The proof follows three steps: (i) Lemma 2 ensures that we never eliminate the optimal path during the search. (ii) Eventually, a tuple $(cost, v, P)$ where $v = g$ will be pulled from the open list and we thus find a path to the goal. (iii) the first time such a tuple is retrieved from the open list, it must have the minimal cost in the open list, and since it is the element of

minimal cost in the open list and the cost is monotone, there cannot be another subpath in the open list that, when extended until g, achieves a smaller cost. ∎

5) Cost-to-Go Heuristic: While Algorithm 1 is optimal, its runtime scales exponentially with the size of the graph. The runtime can be improved with a cost-to-go heuristic as in an A^* or D^* algorithms [42]. To use a heuristic, we augment the path P^u with a virtual edge to the goal. This virtual edge allows for including an estimate for the cost-to-go. An A^* algorithm simply adds a heuristic value for the cost-to-go to the current cost. In contrast, our problem considers a maximization in the cost as well as a potentially non-linear composition function. Thus, we explicitly add an edge and then calculate the WM cost in line 12 for the augmented path. The objective values of the virtual edge must be chosen such that the WM cost of the augmented path is an underestimate of the optimal path to the goal. For instance, if one objective is length, we can set the length of virtual edge to the Euclidean distance while other objective values are zero.

6) Runtime Budgeting: Finally, we can modify Algorithm 1 to find potentially suboptimal solutions in polynomial runtime, similar to anytime algorithms such as ARA* [43]. To that end, we introduce a budget b for the number of predecessor paths leading to every vertex that we can store. We then only add a new tuple in line 13 when the number of tuples with a path ending at u in the open list is below b. This prevents the `open_list` to grow exponentially, yet might prevent the algorithm from finding an optimal solution.

In summary, we have shown how the WM cost can be incorporated in continuous- and discrete-space planning problems. For graph based planning we provided hardness results together with a complete algorithm.

V. NUMERICAL RESULTS

To illustrate the advantages of the WM method, we consider several motion planning problems with multiple objectives and compare the attainable solutions when using either WS and WM. Further, we investigate the runtime of Algorithm 1.

A. Comparison of WS and WM Cost Functions

For given objective functions, we compare how expressive WM ans WS approaches are. We approximate the sets of attainable solutions $\mathcal{T}^{\mathtt{sum}}$ and $\mathcal{T}^{\mathtt{max}}$ for the WS and WM cost functions as follows: We randomly sample a large set of weights $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^k$ and then compute the respective sets of optimal solutions $\mathcal{S}^{\mathtt{sum}} = \{T^{\mathtt{sum}}(\boldsymbol{w}^1), \ldots, T^{\mathtt{sum}}(\boldsymbol{w}^k)\}$ solving (2) and $\mathcal{S}^{\mathtt{max}} = \{T^{\mathtt{max}}(\boldsymbol{w}^1), \ldots, T^{\mathtt{max}}(\boldsymbol{w}^k)\}$ solving (4). Thus, $\mathcal{S}^{\mathtt{sum}}$ and $\mathcal{S}^{\mathtt{max}}$ are both subsets of the Pareto-front of the MOO described by the given objective functions, i.e., consist of Pareto-optimal solutions.

We use three quantitative measures to compare $\mathcal{S}^{\mathtt{sum}}$ and $\mathcal{S}^{\mathtt{max}}$: dispersion, coverage and number of unique solutions on the Pareto-fronts. The dispersion captures *gaps* in the approximation of the Pareto-front, and is defined as follows:

Definition 3 (Dispersion): Given solutions $\mathcal{S} = \{T^1, \ldots, T^k\}$, the dispersion of $\mathcal{S}$ is the maximum distance between

TABLE I
NUMERICAL RESULTS FOR DIFFERENT PLANNING PROBLEMS

Problem	Cost	↓ Dispersion	↑ Coverage	↑ # Solutions
Obstacles	SUM	0.49	0.49	9
	MAX	0.15	0.65	17
Driver	SUM	0.52	0.95	22
	MAX	0.28	0.97	29
Graph-1	SUM	0.26	0.71	11
	MAX	0.19	0.75	40
Graph-2	SUM	0.42	0.73	6
	MAX	0.20	0.80	10

a point $\boldsymbol{p}$ on the Pareto-front and the closest $\boldsymbol{f}(T^i)$ for $i = 1, \ldots, k$.

In principle, this distance should be defined as a measure along the Pareto-front, yet for practical purposes we use the Euclidean distance. Note that for a useful interpretation of dispersion measure, we require objectives to be normalized. Coverage captures the volume of the set of points that is dominated by the solutions $\mathcal{S}$ [44]. In a minimization problem with normalized features we sample over the set $[0, 1]^n$ to estimate coverage. Lastly, sampling k different weights does often not lead to k different solutions. Thus, given a set $\mathcal{S} = \{T^1, \ldots, T^k\}$, we compute the number of trajectories T^i where the Euclidean distance between $\boldsymbol{f}(T^i)$ and $\boldsymbol{f}(T^j)$ is above some threshold δ for all $i, j = 1, \ldots, k$. We refer to this measure as the number of unique solutions. All experiments run with $k = 200$ samples and distance threshold $\delta = 0.01$.

1) Simple Obstacles: First, we revisit the example from Fig. 1 and provide numerical results in Table I (labelled *Obstacles*). We observe that WM outperforms WS on all three metrics, and with a large margin on dispersion on and number of solutions. This highlights the significant shortcomings of the WS even in very simple planning problems.

2) Continuous space Motion Planning: The second experiments considers a continuous space motion planner. We use the driver experiment that is popular in numerous studies on reward learning in HRI, for instance [10], [11], [25]. An autonomous car navigates on a three lane road in the presence of a human-driven vehicle. The problem considers four objectives: heading, position in the lane, speed and distance to the other car. We solve the problem numerically using a numerical solver for constrained non-convex optimization. The min-max objective is implemented as in (6).

Qualitative results are shown in Fig. 2. Since the numerical solver may return suboptimal solutions, we filtered all trajectories that were dominated by another trajectory. Overall, the WM yields a larger variety of solutions. In particular, the solutions for the WS method are only variations of few types of trajectories, while WM offers more nuanced solutions. On the evaluation measures, the WM clearly outperforms the WS with respect to dispersion and the number of unique solutions, yet by a smaller margin than in the *Obstacles* experiment. For coverage WM has only a small benefit.

3) Graph-Based Motion Planning: In the third setup we consider a probabilistic roadmap (PRM) with 1000 vertices, shown in Fig. 3. Similar to the first experiment, the objectives are
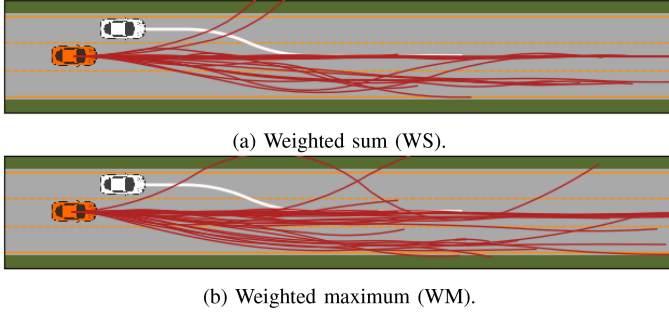
(a) Weighted sum (WS).

(b) Weighted maximum (WM).

Fig. 2. Results for the driver experiments. White shows the human driven car, red trajectories show solutions for the autonomous car.

Fig. 4. Performance of heuristic variants of Algorithm 1 with different computation budgets b. Shown are the cost ratio over an optimal solution and the computation time ratio over Dijkstra's algorithm.

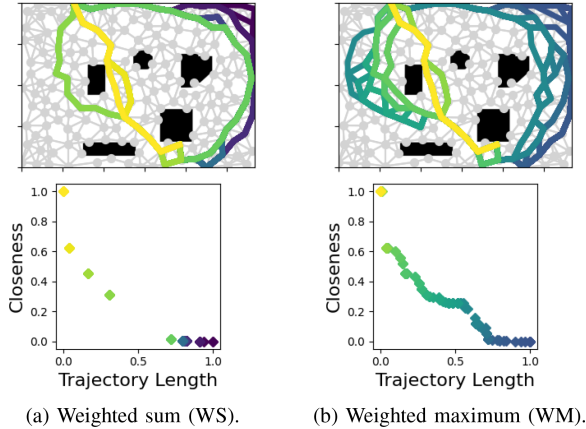(a) Weighted sum (WS).   (b) Weighted maximum (WM).

Fig. 3. Optimal paths for a graph based motion planning problem with two objectives: Path length, and the summed distance to an obstacle. Upper: colored trajectories are solutions found by the weighted sum (a) and weighted maximum (b), respectively. Lower: Objective values of the trajectories of the upper plot.

path length and closeness to obstacles. We consider two problem variations for closeness: the summed closeness, labelled as Graph-1 in Table I with an example shown in Fig. 3, and minimum closeness, labelled as Graph-2.

In Fig. 3 we observe that the WM finds a larger variety of paths, some falling into a homotopy class for which the WS method does not find any path. In the Pareto-fronts WS exhibits several large gaps, while the WM covers the Pareto-front more densely. The gaps of the WS correspond to non-convex parts of the Pareto-front, implying that these parts cannot be covered by the WS for any choice of weights. The measures in Table I show again a substantially smaller dispersion, slightly higher coverage and higher number of solutions for WM compared to WS in both graph problems.

In summary, in all three planning problems the proposed method is able to find better sets of Pareto-optimal trade-offs compared to the weighted sum method.

B. Performance of WM Planning on Graphs

In a second experiment, we investigate the performance of Algorithm 1 when using the computation budget and the cost-to-go heuristic. As a heuristic, we add a virtual edge where the length equals the Euclidean distance to the goal and the closeness is zero. We use a PRM with 2000 vertices, similar to the one in
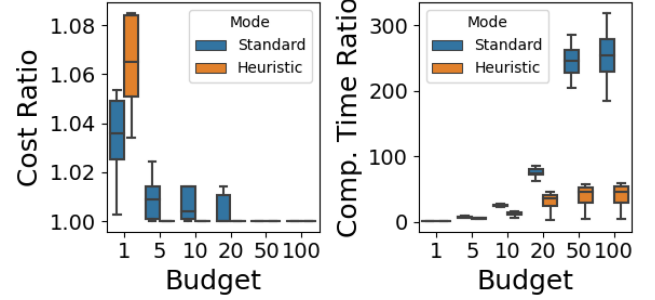
Fig. 3. In 1000 trials, we randomise start and goal locations, as well as the weights in the cost function. Fig. 4 shows the cost ratio compared to optimal and the computation time for various computation budgets b. For the standard implementation without heuristic, we observe that for $b = 50$, all returned solutions are almost optimal (ratio < 1.001). This comes with an increase in computation time by a factor of 250 on average, but still remains below 3 seconds (Hardware specification: Intel i7-11800H @2.3 GHz, 32 GB RAM.). Using the cost-to-go heuristic keeps average the runtime increase below a factor of 35 (or $< .5$ seconds). Moreover, the heuristic also allows for finding close-to-optimal solutions with a very small budget of $b = 5$, where the runtime increase is negligible. Only for $b = 1$ the heuristic can misguide the search and yield suboptimal solutions. Yet, this does not invalidate the admissibility of the chosen heuristic: For small b the algorithm has no guarantee for finding an optimal solution, independent of the heuristic. In conclusion, the cost-to-go heuristic and computation budget allow for finding paths with minimal WM cost within a practical runtime.

VI. DISCUSSION

We studied WM cost functions as an alternative to commonly used WS costs in motion planning problems with multiple objectives. We showed that while the WS method is widely used, it might only represent a small subset of all optimal trade-offs when at least one of the objectives is non-convex. We proposed a WM approach as an alternative cost function, which is Pareto-complete. Further, we showed how the WM cost can be used in continuous-space planning, characterized the hardness for graph-based planning and presented a novel path planning algorithm. Our simulations showed that the proposed WM cost is substantially more expressive than the WS across different motion planning problems, and that our proposed path planning algorithm can efficiently find close-to-optimal solutions. While the WM formulation makes path planning on graphs NP-hard, our simulation results show that it allows for finding substantially richer sets of solutions, recovering all parts of the Pareto-front. Further, using runtime budgeting and the cost-to-go heuristic allows for computing close-to-optimal solutions within a practical computation time.

Future work should consider how WM costs can used for learning user preferences in human-in-the-loop learning

problems. Given its advantageous expressiveness the WM allows for designing user models that represent a wider variety of user preferences with the same number of parameters. Another research direction is investigating how robot complex multi-robot routing problems can be solved for a WM cost. Further, for discrete space planning we assumed a monotonic composition function. Future work could include non-monotonic cost functions to broaden the range of applications. Lastly, finding suitable parameters for the WM cost remains a challenge. Thus, we plan to adapt our earlier work [45] to find sets of representative weights for the WM cost.

REFERENCES

[1] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2011, pp. 163–168.

[2] F. Christianos, P. Karkus, B. Ivanovic, S. V. Albrecht, and M. Pavone, "Planning with occluded traffic agents using bi-level variational occlusion models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5558–5565.

[3] A. Botros and S. L. Smith, "Tunable trajectory planner using G^3 curves," *IEEE Trans. Intell. Veh.*, vol. 7, no. 2, pp. 273–285, Jun. 2022.

[4] N. Wilde, A. Blidaru, S. L. Smith, and D. Kulić, "Improving user specifications for robot behavior through active preference learning: Framework and evaluation," *Int. J. Robot. Res.*, vol. 39, no. 6, pp. 651–667, 2020.

[5] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "Soc-NavBench: A grounded simulation testing framework for evaluating social navigation," *ACM Trans. Hum.-Robot Interact.*, vol. 11, no. 3, pp. 1–24, 2022.

[6] M. Zucker et al., "Chomp: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9/10, pp. 1164–1193, 2013.

[7] Z. Lu, Z. Liu, G. J. Correa, and K. Karydis, "Motion planning for collision-resilient mobile robots in obstacle-cluttered unknown environments with risk reward trade-offs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7064–7070.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, Art. no. 1.

[9] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29.

[10] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proc. Robot.: Sci. Syst.*, 2017.

[11] E. Biyik, M. Palan, N. C. Landolfi, D. P. Losey, and D. Sadigh, "Asking easy questions: A user-friendly approach to active reward learning," in *Proc. 3rd Conf. Robot Learn.*, 2019, pp. 1177–1190.

[12] S. Habibian, A. Jonnavittula, and D. P. Losey, "Here's what I've learned: Asking questions that reveal reward learning," *ACM Trans. Hum.-Robot Interact.*, vol. 11, no. 4, pp. 1–28, 2022.

[13] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, vol. 5252. Berlin, Germany: Springer, 2008.

[14] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 250–256.

[15] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "DiffStack: A differentiable and modular control stack for autonomous vehicles," in *Proc. Conf. Robot Learn.*, 2023, pp. 2170–2180.

[16] Z. Zuo et al., "MPC-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 3, pp. 513–522, Sep. 2021.

[17] P. Hang, C. Lv, C. Huang, J. Cai, Z. Hu, and Y. Xing, "An integrated framework of decision making and motion planning for autonomous vehicles considering social behaviors," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14458–14469, Dec. 2020.

[18] P. De Petris, M. Dharmadhikari, H. Nguyen, and K. Alexis, "Risk-aware motion planning for collision-tolerant aerial robots subject to localization uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 4561–4568.

[19] Y. Che, A. M. Okamura, and D. Sadigh, "Efficient and trustworthy social navigation via explicit and implicit robot–human communication," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 692–707, Jun. 2020.

[20] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[21] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Sci. Robot.*, vol. 8, no. 84, 2023, Art. no. eadf7843

[22] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.

[23] M. Cáp and J. Alonso-Mora, "Multi-objective analysis of ridesharing in automated mobility-on-demand," in *Proc. Robot.: Sci. Syst.*, 2018.

[24] N. Wilde, A. Sadeghi, and S. L. Smith, "Learning submodular objectives for team environmental monitoring," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 960–967, Apr. 2022.

[25] N. Wilde, D. Kulić, and S. L. Smith, "Active preference learning using maximum regret," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10952–10959.

[26] N. Wilde, E. Biyik, D. Sadigh, and S. L. Smith, "Learning reward functions from scale feedback," in *Proc. Conf. Robot Learn.*, 2022, pp. 353–362.

[27] I. Y. Kim and O. L. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Struct. Multidisciplinary Optim.*, vol. 29, pp. 149–158, 2005.

[28] I. Y. Kim and O. De Weck, "Adaptive weighted sum method for multiobjective optimization: A new method for pareto front generation," *Struct. Multidisciplinary Optim.*, vol. 31, no. 2, pp. 105–116, 2006.

[29] A. Thoma, K. Thomessen, A. Gardi, A. Fisher, and C. Braun, "Prioritizing paths: An improved cost function for local path planning for UAV in medical applications," in *Proc. 20th Aust. Int. Aerosp. Congr.: 20th Australian Int. Aerosp. Congr. Eng.*, 2023, pp. 324–329.

[30] D. Yi, M. A. Goodrich, and K. D. Seppi, "MORRF*: Sampling-based multi-objective motion planning," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1733–1739.

[31] B. Sakcak and S. M. LaValle, "Complete path planning that simultaneously optimizes length and clearance," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 10100–10106.

[32] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, 2013, pp. 191–199.

[33] X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 977–983.

[34] X. D. Ding, B. Englot, A. Pinto, A. Speranzon, and A. Surana, "Hierarchical multi-objective planning: From mission specifications to contingency management," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 3735–3742.

[35] S. D. Bopardikar, B. Englot, and A. Speranzon, "Multiobjective path planning: Localization constraints and collision probability," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 562–577, Jun. 2015.

[36] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial Optimization*, vol. 1. Berlin, Germany: Springer, 2011.

[37] Y. Censor, "Pareto optimality in multiobjective problems," *Appl. Math. Optim.*, vol. 4, no. 1, pp. 41–59, 1977.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

[39] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6. Belmont, MA, USA: Athena Sci., 1997.

[40] M. Ehrgott, *Multicriteria Optimization*. vol. 491. Berlin, Germany: Springer, 2005.

[41] X. Gandibleux, F. Beugnies, and S. Randriamasy, "Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function," in *4OR*, vol. 4, no. 1, pp. 47–59, 2006.

[42] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artif. Intell.*, vol. 155, no. 1/2, pp. 93–146, 2004.

[43] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, vol. 16.

[44] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[45] A. Botros, A. Sadeghi, N. Wilde, J. Alonso-Mora, and S. L. Smith, "Error-bounded approximation of pareto fronts in robot planning problems," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2022, pp. 506–522.