Regret-based Sampling of Pareto Fronts for Multi-Objective Robot Planning Problems

Alexander Botros, Nils Wilde, Armin Sadeghi, Javier Alonso-Mora and Stephen L. Smith

***Abstract*—Many problems in robotics seek to simultaneously optimize several competing objectives. A conventional approach is to create a single cost function comprised of the weighted sum of the individual objectives. Solutions to this scalarized optimization problem are Pareto optimal solutions to the original multi-objective problem. However, finding an accurate representation of a Pareto front remains an important challenge. Uniformly spaced weights are often inefficient and do not provide error bounds. We address the problem of computing a finite set of weights whose optimal solutions closely approximate the solution of any other weight vector. To this end, we prove fundamental properties of the optimal cost as a function of the weight vector. We propose an algorithm that greedily adds the weight vector least-represented by the current set, and provide bounds on the regret. We extend our method to include suboptimal solvers for the scalarized optimization, and handle stochastic inputs to the planning problem. Finally, we illustrate that the proposed approach significantly outperforms baseline approaches for different robot planning problems with varying numbers of objective functions.**

I. Introduction

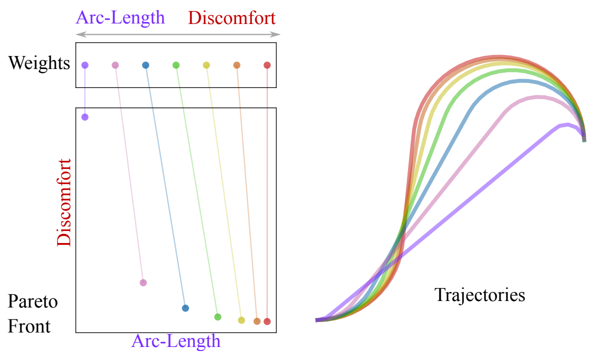
In many robotic planning problems, one seeks to optimize several competing objectives. Examples include motion planning and trajectory generation for autonomous vehicles [1]–[5], human-robot cooperation for task completion [6], warehouse robotics [7], mobility-on-demand servicing [8], and neural networks [9] to name a few.

In Multi-Objective Optimization (MOO) [10], one seeks solutions that achieve an appropriate trade-off between objectives. These problems are often solved through scalarization, which combines multiple objectives into a single cost function. An example is *linear scalarization*, where the cost function is a weighted sum of objectives. Linear scalarization leads to solutions that are Pareto optimal for the MOO problem [11]. That is, the solution to the scalarized single-objective problem cannot be changed to improve the value of one of its objectives without degrading the value of another. However, a challenge in linear scalarization is how to appropriately choose weights on each objective to achieve a desired trade-off.

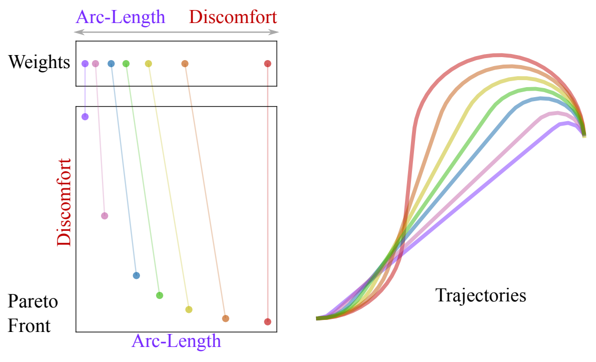
A. Botros and N. Wilde contributed equally.

Acknowledgements: This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), as well as by the European Union's Horizon 2020 research and innovation program under Grant 101017008.

A. Botros, A. Sadeghi and S. L. Smith are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. N. Wilde and J. Alonso-Mora are with the Department for Cognitive Robotics, Delft University of Technology, Delft, Netherlands. {abotros,a6sadegh,stephen.smith}@uwaterloo.ca {n.wilde,j.alonsomora}@tudelft.nl



(a) Uniformly sampled weights.



(b) Proposed min-regret sampling of weights.

Fig. 1: Optimal trade-offs between trajectory length and jerk for a Dubins vehicle, comparing solutions computed for uniformly sampled weights (a) and solutions found with the proposed method (b).

This work is motivated by two classes of robotics problems: 1) obtaining online near-optimal solutions to a linearly scalarized multi-objective optimization problem (LSMOP) for any given weight vectors, and 2) learning the preferred solution behavior of a human user. Applications of the first class include [8] where an adjustable trade-off between the service quality and operating costs for autonomous mobility-on-demand systems are optimized. Conversely, the second class of problems seeks to compute a weight vector representing the relative importance of each objective to a user given some knowledge of that users' preferred solutions [7], [12]–[16].

In either class, it is often beneficial to pre-compute solutions to the LSMOP for a set of weight vectors. If the LSMOP is computationally intensive to solve, requiring online solutions may be impractical. This motivates the problem of finding a set of weight vectors and their corresponding optimal solutions such that for any possible weight vector, there exists an element of the set whose solution is close to optimal. A naive approach would involve densely sampling the set of all possible weight vectors. However, the sensitivity of some objectives to changes in solution may result in a skewed sampling of Pareto optimal solutions. This is illustrated in Figure 1 where we seek to compute a Dubins trajectory between fixed start and goal configurations that minimizes a trade-off between trajectory length and discomfort (measured as the integral of the squared jerk over the trajectory [17]). In the top figure, a set of weight vectors is selected uniformly and the resulting solution trajectories (right) and Pareto front (bottom left) is shown. We observe that uniformly sampling weights generates multiple similar trajectories and does not approximate well all different trade-offs, as shown by the large gap in the Pareto Front.

In this paper, we propose a greedy algorithm that constructs a set of weight vectors Ω by recursively adding weight vectors that are *least represented* by the current set. Only assuming that the given objective functions are bounded the corresponding optimal solutions for weights in Ω provide homogeneous coverage of the Pareto front, as illustrated in Figure 1b.

A. Contributions

The contributions of this work are as follows:

1) Assuming that an exact solver for an LSMOP is available, we propose an algorithm to compute a set of weight vectors Ω whose corresponding solutions provide homogeneous sampling of the Pareto front.
2) We provide a bound on the error incurred from approximating the optimal solution to an LSMOP for any arbitrary weight with a solution corresponding to a weight vector from Ω.
3) We relax the requirement that an exact solver be available and extend the first contribution to include a sub-optimal solver. We extend the second contribution if an approximation factor for this solver is known.
4) If a probability density function over the set of all weights is known (e.g., representing the likelihood that a desired weight lies in a sub-region), we extend the first and second results to reflect the *expected* error.
5) Finally, we showcase the advantages of our sampling algorithm in different robotics applications, namely robot trajectory planning, multi-vehicle traveling salesman problem (mTSP), and learning user preferences.

Our earlier work [18] established the first and second contributions listed above. This paper extends that work by incorporating sub-optimal solvers (third contribution) and stochastic settings (fourth contribution) into the proposed approach. Finally, we extend the simulations to showcase our third and fourth contributions and include the work of [19] as a baseline.

B. Related Work

We review three areas of related work: a) the use of weighted sums to tackle multi-objective planning problems in robotic applications, b) the use of weighted sums to describe user preferences in HRI, and c) techniques for approximating Pareto Fronts.

a) Linear Scalarization in Robotics: The simplicity of linear scalarization has made it one of the most widely used tools in robotics for considering different objectives in a cost or reward function. Though the technique is not able to capture all Pareto-optimal solutions for non-convex fronts, it does guarantee that all LSMOP solutions are Pareto-optimal. For instance, cost functions in autonomous driving often consider objectives such as trajectory length, comfort (measured via jerk), or clearance [1]–[5], [17], [20]. Other examples for motion planners that use weighted sums to balance between objectives include local planners for mobile robots navigating in cluttered environments [21] or social spaces [22], [23], trajectory planners for manipulators [24], [25], and multi-robot planning [26].

In [27], a trajectory smoothing algorithm is proposed based on the weighted sum of competing objectives, namely trajectory length, smoothness, and obstacle distances. The authors of [28] minimize the weighted trade-off between mission completion time and communication outage duration in the navigation of cellular-connected UAVs, while in [29], linear scalarization is used to optimize robotic limitations and observation rewards for use in autonomous human activity tracking. The authors of [30] consider bi-objective path planning. In particular, the goal is to simultaneously optimize for path length and clearance to obstacles in the plane for which they propose a complete and efficient algorithm. Our work does not consider specific objectives as the above-mentioned papers. Instead, we focus on Pareto-optimal trade-offs for any multi-objective planning problem that is formulated as a weighted sum.

b) Weighted sums describing user preferences: In human-robot interaction (HRI), weight vectors are used to represent a user's preference for robot behaviour, *i.e.,* the relative importance of objectives [12], [15], [16], [31]–[33]. In *reward learning* the objective is to learn a user's weight vector using interactions such as demonstrations, corrections, or choice feedback. In order to expedite the learning process, feasible solutions for the multi-objective optimization problem are often pre-computed and shown to the user who then provides feedback. In [12], [16], [34], each pre-computed solution is generated with random action sequences. Thus, the solutions used in the learning process are usually not optimal for any weight. In [15], [35], the authors pre-compute Pareto-optimal solutions enabling an active learning method based on regret leading to significant improvement over randomly generated solutions. However, the work in [15], [35] rely on uniformly sampled weight vectors. Though this approach asymptotically covers the set of all LSMOP-optimal solutions, it can be inefficient as different weight vectors can have very similar, or even identical solutions. The counter intuitive relationship between weights and collected reward was further studied in [36]. Moreover, the authors of [37] observed that the presence of similar solution strongly influences the Boltzmann decision model which is commonly used in HRI. They propose a decision model where a similarity metric corrects the bias induced by a high number of similar trajectories. Similarly, in our work we are interested in finding solutions with dissimilar features. While [37] handles the over-representation of similar solutions in the ground set with their proposed decision model, we instead address the problem at an earlier stage: our algorithm can be used to generate a ground set where similarities are minimized.

c) Approximating Pareto Fronts: Our work is motivated from the following observation: a uniform sampling of weights does not generally provide a uniform sampling of solutions as illustrated in Figure 1. These shortcomings of weighted sum methods are well studied within the optimization literature [38], [39], yet have received less attention in the robotics community despite the wide usage of weighted sums as objective functions. Researchers in optimization developed many techniques for approximating Pareto-fronts, including more complex scalarization methods such as Chebyshev scalarization [38]. Unfortunately, these methods are often not directly applicable to robot planning since they require solving more complex scalar optimization problems. As a consequence, many multi-objective robot planning problems still rely on the simple weighted sum formulation. Finding a set of representative scalarization weights requires solving a series of optimization problems *e.g.,* solving path or motion planning instances, which can be computationally burdensome. Thus, efficient computation of such a set is of particular interest.

Closely related to our work is the *Adapted Weighted Sum* (AWS) method [19], [40]. The AWS iteratively places Pareto samples by partitioning existing samples into subsets and placing new samples into the subset that has the largest *gap*. This requires solving an optimization problem with an additional linear constraint on the objective value, which can result in a harder problem then the original weighted sum optimization. In contrast, our method identifies the most promising weights for additional samples and then solves the weighted sum optimization problem. Further, our method minimizes the regret of the weighted samples and returns an error bound.

The authors of [41] present a Pareto front approximation for trajectory planning using Markov chain random walks. Their goal is to *uniformly* place samples on the Pareto front, while our goal is to *minimize error* in the space of Pareto-optimal costs. Moreover, the random walk technique does not rely on a weighted sum objective but does not generalize to an arbitrary choice of objective functions. Similar to our work, the authors of [42] offer a technique of Pareto-uniform sampling based on equispacing constraints. However, they only consider the case of two competing objectives. Further, they accomplish their goal by solving a nested optimization with the original LSMOP as the inner-most problem, which can be significantly harder. The work in [43] proposes a set of weight vectors that approximately uniformly cover a Pareto front specifically for use in the design of robots. The authors design the set that minimizes the total squared error between the value of the objectives in the set and heuristic objectives. It therefore relies on the approximate optimality of these objectives. The work in [44] proposes a method to cover the set of Pareto-optimal solutions specifically for use in reinforcement learning applications. The authors seek to compute policies that maximize expected returns by computing, storing, and updating a set of samples. In [45] the authors provide a means of exploring a (possibly non-convex) Pareto front in order to obtain a solution that is near-optimal for a user. That work starts with an initial guess solution which moves in a direction according to a user's preference. While the convexity of the Pareto front (a requirement for linear scalarization to obtain all Pareto-optimal solutions) is not assumed, their technique requires solving the LSMOP online as the Pareto front is explored. Moreover, the requirement of a user-preferred direction is not assumed in our work.

In summary, most state-of-the-art methods either address specific problem setups and thus do not generalize across different robot planning problems, or address the problem of finding scalarization weights more generally but pose a complex optimization problem to compute a set of weights. Our work considers the weighted sum formulation for an arbitrary choice of objective functions. We iteratively compute a set of weights, only requiring solving a linear program and the weighted sum objective for different weights, and return a bound on the approximation error for the computed set.

II. Problem Statement

For $n \in \mathbb{N}$, a general multi-objective optimization problem (MOP) is of the form

$$\min_{s\in\mathcal{S}} \Big\{ f_1(s), f_2(s), \ldots, f_n(s) \Big\}. \tag{1}$$

Here, the set of feasible solutions given constraints is denoted $\mathcal{S}$, and it is desired to simultaneously minimize n objectives $f_i(s)$, for $i \in \{1, \ldots, n\}$. However, such a solution typically does not exist. As a result, multi-objective optimization often seeks Pareto-optimal solutions: solutions $s \in S$ such that there does not exist another solution $\bar{s} \in S$ where $f_i(\bar{s}) \leq f_i(s)$ for all i, and with strict inequality for at least one i. The linear scalarization of the MOP above involves the creation of a single cost function by introducing a vector of weights $\boldsymbol{w} = [w_1, w_2, \ldots, w_n] \in \mathbb{R}^n_{\geq 0}$. Let $c(s, \boldsymbol{w})$ denote the cost of the solution s evaluated by the weights $\boldsymbol{w}$, i.e.,

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

where $\boldsymbol{f}(s) = [f_1(s), \ldots, f_n(s)]$, $\forall s \in \mathcal{S}$. The resulting linearly scalarized multi-objective optimization problem (LSMOP) is to solve

$$u(\boldsymbol{w}) = \min_{s\in\mathcal{S}} c(s, \boldsymbol{w}). \tag{3}$$

For any weight $\boldsymbol{w} \in \mathbb{R}^n_{\geq 0}$, solution $s \in \mathcal{S}$, and $\lambda \in \mathbb{R}_{>0}$, it holds that $c(s, \lambda\boldsymbol{w}) = \lambda c(s, \boldsymbol{w})$ implying that a minimizer of $c(s, \boldsymbol{w})$ also minimizes $c(s, \lambda\boldsymbol{w})$. Further, if $\boldsymbol{w} = [0, 0, \ldots, 0]$, then $u(\boldsymbol{w})$ is trivially 0. Thus, given $\boldsymbol{w} = [w_1, \ldots, w_n]$ where not all elements are identically 0, and letting $\lambda = \left(\sum_{i=1}^{n} w_i\right)^{-1}$, we can obtain all non-trivial optimal solutions $u(\boldsymbol{w})$ for all $\boldsymbol{w} \in \mathbb{R}^n_{\geq 0}$ using weights $\boldsymbol{w} \in \mathcal{W}$ where

$$\mathcal{W} = \Big\{ \boldsymbol{w} \in \mathbb{R}^n_{\geq 0}, \sum_{i=1}^{n} w_i = 1 \Big\}. \tag{4}$$

We refer to the set $\mathcal{W}$ as the *weight space*, and define

$$s^*(\boldsymbol{w}) = \arg\min_{s\in\mathcal{S}} \boldsymbol{w} \cdot \boldsymbol{f}(s), \ \forall \boldsymbol{w} \in \mathcal{W}, \tag{5}$$

as an *optimal solution* given weights $\boldsymbol{w}$, implying that $u(\boldsymbol{w}) = c(s^*(\boldsymbol{w}), \boldsymbol{w})$ by (3). We start with the following assumptions:

Assumption 1 (Exact Solution)**.** An exact solver exists for the optimization problem (3).

Assumption 2 (Bounded Objectives)**.** For any weight $\boldsymbol{w} \in \mathbb{R}^n_{\geq 0}$, and any optimal solution $s^*(\boldsymbol{w})$ in (5), the objectives $\boldsymbol{f}(s^*(\boldsymbol{w}))$ are bounded.

While Assumption 2 persists throughout this paper, Assumption 1 is relaxed for sub-optimal solvers (Section V).

In this work, we propose a method to compute a finite set of weights $\Omega \subset \mathcal{W}$ that will, for any $\boldsymbol{w}^* \in \mathcal{W}$, allow us to approximate $u(\boldsymbol{w}^*)$ with $u(\boldsymbol{w}')$ for an appropriately chosen $\boldsymbol{w}' \in \Omega$. To evaluate the quality of a candidate set Ω, we use the notion of regret from [15], [46], defined formally here:

Definition 1 (Regret)**.** Given two weights $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$, the regret of $\boldsymbol{w}'$ under $\boldsymbol{w}^*$ is defined as

$$r(\boldsymbol{w}'|\boldsymbol{w}^*) = \boldsymbol{w}^* \cdot \boldsymbol{f}\left(s^*(\boldsymbol{w}')\right) - u(\boldsymbol{w}^*). \tag{6}$$

Intuitively, $r(\boldsymbol{w}'|\boldsymbol{w}^*)$ represents the *error* in cost incurred by using an optimal solution given weight $\boldsymbol{w}'$ (given by $s^*(\boldsymbol{w}')$) to approximate a solution given weight $\boldsymbol{w}^*$. We now formally state the main problem addressed in this work.

Problem 1 (Min-Max Regret Sampling)**.** For the LSMOP (3) and an integer $K > 0$, find a set of weights Ω that solves

$$\begin{aligned} &\min_{\Omega} \max_{\boldsymbol{w}^* \in \mathcal{W}} \min_{\boldsymbol{w}' \in \Omega} r(\boldsymbol{w}'|\boldsymbol{w}^*) \\ &s.t.\ |\Omega| \leq K. \end{aligned} \tag{7}$$

Given a weight $\boldsymbol{w}^* \in \mathcal{W}$ and a set $\Omega \subset \mathcal{W}$, the first minimization in (7), $\min_{\boldsymbol{w}' \in \Omega} r(\boldsymbol{w}'|\boldsymbol{w}^*)$ represents the sub-optimality of approximating a solution $s^*(\boldsymbol{w}^*)$ with a solution $s^*(\boldsymbol{w}')$ where $\boldsymbol{w}'$ is the weight in Ω that minimizes this sub-optimality – i.e., $\boldsymbol{w}'$ is a best representative of $\boldsymbol{w}^*$ in Ω. The maximization $\max_{\boldsymbol{w}^* \in \mathcal{W}} \min_{\boldsymbol{w}' \in \Omega} r(\boldsymbol{w}'|\boldsymbol{w}^*)$, represents the regret of the worst represented weight $\boldsymbol{w}^* \in \mathcal{W}$ by elements in Ω. We refer to the solution of this maximization as the *maximum regret given* Ω. In total, (7) seeks a set Ω such that the regret of the worst represented element in $\mathcal{W}$ is minimized.

In this paper, we offer an approximate solution to the optimization in (7) by way of an algorithm that computes a feasible solution Ω such that the maximum regret given Ω is bounded. In the next section, we provide the theoretical groundwork that makes this solution possible.

III. Problem Analysis

We begin with a structural analysis of the cost function $u(\boldsymbol{w})$ from (3) to derive an efficient algorithm for solving Problem 1. First, we make two critical observations.

Observation 1. Given any two weights $\boldsymbol{w}^*, \boldsymbol{w}' \in \mathcal{W}$, we have

$$u(\boldsymbol{w}^*) \leq \boldsymbol{w}^* \cdot \boldsymbol{f}(s^*(\boldsymbol{w}')), \tag{8}$$

That is an optimal solution given weights $\boldsymbol{w}^*$ will incur no higher cost than a solution that is optimal for some different weight vector $\boldsymbol{w}'$. Here, $s^*(\boldsymbol{w}')$ is optimal given weights $\boldsymbol{w}'$ (see (5)) but not necessarily optimal given weights $\boldsymbol{w}^*$. By (6), the inequality in (8) implies that $r(\boldsymbol{w}'|\boldsymbol{w}^*) \geq 0$.

Observation 2 (Optimal Cost Concavity)**.** The optimal cost function $u(\boldsymbol{w})$ is a concave function of $\boldsymbol{w}$. Indeed, for each $s \in \mathcal{S}$, the cost $c(s, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(s)$ is an affine function of $\boldsymbol{w}$ (and is therefore concave). Therefore, $u(\boldsymbol{w}) = \min_{s \in \mathcal{S}} c(s, \boldsymbol{w})$ is concave [47, Section 3.2.3].

Observation 2 motivates the following Lemma:

Lemma 1 (Optimal Cost Continuity)**.** Under Assumptions 1, 2, given any two weights in $\mathcal{W}$, $u(\boldsymbol{w})$ is continuous on the line segment connecting those weights.

Proof. By Observation 2, $u(\boldsymbol{w})$ is concave in $\mathcal{W}$. Noting in addition that $\mathcal{W} \subset \mathbb{R}^n$ is convex, it must hold that $u(\boldsymbol{w})$ is continuous on the interior of $\mathcal{W}$. This is because concave functions are continuous on the interior of convex sets. Therefore, it suffices to prove the result for the case that at least one weight lies on the boundary of $\mathcal{W}$. Consider two weights $\boldsymbol{w}', \boldsymbol{w}'' \in \mathcal{W}$ at least one of which lies on the boundary of $\mathcal{W}$. Suppose that $||\boldsymbol{w}' - \boldsymbol{w}''|| \leq \delta$ for some $\delta > 0$ arbitrarily small, and – without loss of generality – that $u(\boldsymbol{w}') > u(\boldsymbol{w}'')$. Because it is assumed that $u(\boldsymbol{w})$ is bounded on $\mathcal{W}$, if it experiences a discontinuity on the line connecting $\boldsymbol{w}', \boldsymbol{w}''$, it must hold that $u(\boldsymbol{w})$ experiences a jump between $\boldsymbol{w}', \boldsymbol{w}''$. That is, there must exist a $M \in \mathbb{R}_{>0}$ independent of δ such that $u(\boldsymbol{w}'') + M \leq u(\boldsymbol{w}')$. Since $||\boldsymbol{w}'' - \boldsymbol{w}'|| \leq \delta$, and $f(s^*(\boldsymbol{w}''))$ is bounded by Assumption 2, there must exist a value of δ sufficiently small so as to guarantee that $(\boldsymbol{w}' - \boldsymbol{w}'') \cdot f(s^*(\boldsymbol{w}'')) < M/2$. Therefore, by construction,

$$\begin{aligned} \boldsymbol{w}' \cdot f(s^*(\boldsymbol{w}'')) &< \frac{M}{2} + \boldsymbol{w}'' \cdot f(s^*(\boldsymbol{w}'')) = \frac{M}{2} + u(\boldsymbol{w}'') \\ &\leq \frac{M}{2} + u(\boldsymbol{w}') - M = u(\boldsymbol{w}') - \frac{M}{2} < u(\boldsymbol{w}'). \end{aligned}$$

Therefore, $r(\boldsymbol{w}''|\boldsymbol{w}') = \boldsymbol{w}' \cdot f(s(\boldsymbol{w}'')) - u(\boldsymbol{w}') < 0$ which is a contradiction by Observation 1. ☐

Critically, the previous results do not require unique solutions $s^*(\boldsymbol{w})$ or continuous objectives $\boldsymbol{f}(s^*(\boldsymbol{w}))$. Extending these results:

Theorem 1 (Convexity of Regret)**.** For a fixed weight $\boldsymbol{w}' \in \mathcal{W}$, the regret $r(\boldsymbol{w}'|\boldsymbol{w})$ is a convex function of $\boldsymbol{w}$.

Proof. By (6), $r(\boldsymbol{w}'|\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(s^*(\boldsymbol{w}')) - u(\boldsymbol{w})$ where $\boldsymbol{w} \cdot \boldsymbol{f}(s^*(\boldsymbol{w}'))$ is linear in $\boldsymbol{w}$ and $u(\boldsymbol{w})$ is concave (Observation 2). Thus, $r(\boldsymbol{w}'|\boldsymbol{w})$ is the difference of linear and concave functions of $\boldsymbol{w}$ which is convex. ☐

Because $u(\boldsymbol{w})$ is continuous and concave, it must hold that the function lies below any sub-gradient. This motivates the following Corollary which follows directly from the definition of $u(\boldsymbol{w})$ and the concavity of $c(s, \boldsymbol{w})$ for each fixed $s \in \mathcal{S}$ [47, Section 6.5.5].

Corollary 1.1 (Sub-gradient Optimal Cost)**.** For any $\boldsymbol{w} \in \mathcal{W}$ and any minimizing solution $s^*(\boldsymbol{w}) \in \mathcal{S}$, the vector of objectives $\boldsymbol{f}(s^*(\boldsymbol{w}))$ is a sub-gradient, $\partial u(\boldsymbol{w})$, of u at $\boldsymbol{w}$.

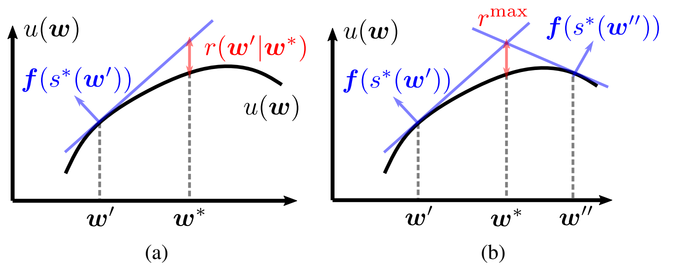


Fig. 2: (a) Regret as the error of a first order approximation. (b) Maximum regret given a set $\Omega = \{\boldsymbol{w}', \boldsymbol{w}''\}$ at the intersection of their tangent lines.

Observe that sub-gradients are defined even for non-differentiable continuous functions. Further, The results above imply that given two weights $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$, the regret $r(\boldsymbol{w}'|\boldsymbol{w}^*)$ — which coincides with the error incurred by approximating a solution $s^*(\boldsymbol{w}^*)$ with the solution $s^*(\boldsymbol{w}')$ — is exactly the error of approximating a concave function via *linear interpolation*. Indeed, the first order approximation of $u(\boldsymbol{w}^*)$ given $u(\boldsymbol{w}')$ is given by $u(\boldsymbol{w}^*) \approx u(\boldsymbol{w}') + \nabla u(\boldsymbol{w}') \cdot (\boldsymbol{w}^* - \boldsymbol{w}')$ assuming u is differentiable at $\boldsymbol{w}'$. However, by Corollary 1.1, $\nabla u(\boldsymbol{w}) = \partial u(\boldsymbol{w}) = \boldsymbol{f}(s^*(\boldsymbol{w}))$. This together with the definition $u(\boldsymbol{w}') = \boldsymbol{f}(s^*(\boldsymbol{w}')) \cdot \boldsymbol{w}'$ allows us to conclude that $u(\boldsymbol{w}^*) \approx u(\boldsymbol{w}') + \boldsymbol{f}(s^*(\boldsymbol{w}')) \cdot \boldsymbol{w}^* - \boldsymbol{f}(s^*(\boldsymbol{w}')) \cdot \boldsymbol{w}' = \boldsymbol{f}(s^*(\boldsymbol{w}')) \cdot \boldsymbol{w}^*$. The error of this first order approximation is $\boldsymbol{f}(s^*(\boldsymbol{w}')) \cdot \boldsymbol{w}^* - u(\boldsymbol{w}^*)$ which is exactly the regret $r(\boldsymbol{w}'|\boldsymbol{w}^*)$. This is illustrated in Figure 2 (a). Further, given any two weights $\boldsymbol{w}', \boldsymbol{w}'' \in \mathcal{W}$, the maximum regret on the line segment L between $\boldsymbol{w}', \boldsymbol{w}''$ given $\Omega = \{\boldsymbol{w}', \boldsymbol{w}''\}$ occurs at the weight on L coinciding with the intersection of the tangent lines to $u(\boldsymbol{w})$ at $\boldsymbol{w}'$ and $\boldsymbol{w}''$ along L (Figure 2 (b)). In light of this analysis, the objective in (7) is solved by a set Ω that provides the best linear interpolation of the concave function $u(\boldsymbol{w})$. These insights are leveraged in the following section.

IV. Algorithm

In this section, we present our solution to Problem 1. The algorithm we propose recursively adds weights to a solution set Ω. A strong candidate weight to add is one that is least represented by the current iteration of Ω. The basic framework for such an approach could be described recursively:

$$\Omega^{k+1} = \Omega^k \cup \{\arg\max_{\boldsymbol{w}^* \in \mathcal{W}} \min_{\boldsymbol{w}' \in \Omega^k} r(\boldsymbol{w}'|\boldsymbol{w}^*)\}, \tag{9}$$

where Ω^k is the solution after k iterations from an initial set. Here, (9) recursively adds the weight with the maximum regret given Ω^k. Obtaining the maximizer $\boldsymbol{w}^*$ is non-trivial due to its nested structure. Instead, our approach replaces $r(\boldsymbol{w}'|\boldsymbol{w}^*)$ in (9) with an upper bound $R(\boldsymbol{w}'|\boldsymbol{w}^*)$ whose maximizer $\boldsymbol{w}^*$ is obtained from a linear program (LP).

Given a set of weights $\Omega \subseteq \mathcal{W}$, we define N as a set of n (recall that n is the number of objectives) linearly independent weights $\boldsymbol{w}^1, \dots, \boldsymbol{w}^n \in \Omega$, and we let $\mathcal{C}(N) \subset \mathbb{R}^n$ denote the convex hull of N. We loosely refer to N as a *neighborhood*, and define a linear lower bound of the objective value $u(\boldsymbol{w})$ from (3) inside a neighbourhood N. Let $P : \mathcal{W} \to \mathbb{R}_{\geq 0}$ be the linear function taking values $P(\boldsymbol{w}^i) = u(\boldsymbol{w}^i)$ for all $\boldsymbol{w}^i \in N$

Algorithm 1:
Minimum-Regret Pareto Sampling (MRPS)

```
Input: An exact solver to find $s^*(\boldsymbol{w}), \boldsymbol{f}(s^*(\boldsymbol{w}))$; a
       budget $K \geq n$
Output: Sampled weights $\Omega$ and maximum regret
$\Omega \leftarrow \{\boldsymbol{e}^i, i = 1, \dots, n\}$ // where $e^i$ is the $i^{th}$ row of
  the $n \times n$ identity matrix
Obtain $s^*(\boldsymbol{e}^i), \boldsymbol{f}(s^*(\boldsymbol{e}^i)), i = 1, \dots, n$ from exact
  solver
$\mathcal{N} \leftarrow \{\Omega\}$
for $k = n$ to $K$ do
    $N$ = neighborhood in $\mathcal{N}$ with maximum $\bar{R}(N)$
    if $\bar{R}(N) = 0$ then
        break // Terminate if maximum upper regret
          bound is 0
    $\Omega \leftarrow \Omega \cup \{\bar{\boldsymbol{w}}(N)\}$
    Obtain $s^*(\bar{\boldsymbol{w}}(N)), \boldsymbol{f}(s^*(\bar{\boldsymbol{w}}(N)))$ from exact solver
    $\mathcal{N} = \mathcal{N} \setminus N$ // Remove max-regret neighborhood
    for $\boldsymbol{w}^i$ in $N$ do
        $N^i \leftarrow N \setminus \{\boldsymbol{w}^i\} \cup \{\bar{\boldsymbol{w}}(N)\}$ // Replace $w^i$ with
          weight of the maximum regret bound
        if $N^i$ is a neighborhood, i.e., its weights are
          lin. independent then
            $\mathcal{N} = \mathcal{N} \cup N^i$
            $\mathcal{F}(N^i) \leftarrow$
              $\mathcal{F}(N) \setminus \{\boldsymbol{f}(s^*(\boldsymbol{w}^i))\} \cup \{\boldsymbol{f}(s^*(\bar{\boldsymbol{w}}(N)))\}$
return $\Omega$ and the maximum value of $\bar{R}(N)$ over all
  $N \in \mathcal{N}$
```

(Figure 3 (a)). We denote the difference between the tangent plane at $\boldsymbol{w}'$ and the P evaluated at $\boldsymbol{w}^*$ with $R(\boldsymbol{w}'|\boldsymbol{w}^*) = \boldsymbol{f}(s^*(\boldsymbol{w}'))\boldsymbol{w}^* - P(\boldsymbol{w}^*),\ \forall \boldsymbol{w}' \in N, \boldsymbol{w}^* \in \mathcal{C}(N)$. Further, let

$$\begin{aligned} \bar{R}(N) &= \max_{\boldsymbol{w}^* \in \mathcal{C}(N)} \min_{\boldsymbol{w}' \in N} R(\boldsymbol{w}'|\boldsymbol{w}^*), \\ \bar{\boldsymbol{w}}(N) &= \arg\max_{\boldsymbol{w}^* \in \mathcal{C}(N)} \min_{\boldsymbol{w}' \in N} R(\boldsymbol{w}'|\boldsymbol{w}^*). \end{aligned} \tag{10}$$

Finally, we let $\mathcal{F}(N)$ represent the set of objective vectors of the neighborhood:

$$\mathcal{F}(N) = \{\boldsymbol{f}(s^*(\boldsymbol{w}^i)), \boldsymbol{w}^i \in N\}. \tag{11}$$

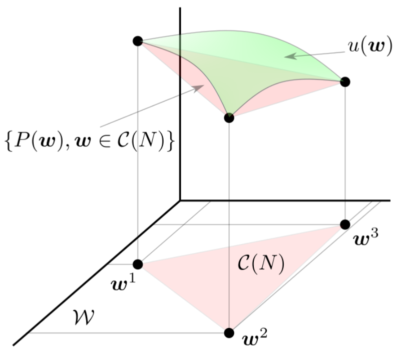
Thus, $R(\boldsymbol{w}'|\boldsymbol{w}^*)$ is similar to $r(\boldsymbol{w}'|\boldsymbol{w}^*)$ from (6), but with $u(\boldsymbol{w}^*)$ replaced with $P(\boldsymbol{w}^*)$. These definitions, illustrated in Figure 3, motivate the Theorem:

Theorem 2 (Upper Bound of Maximum Regret in a Neighborhood)**.** Given a neighborhood N of weights, it holds that

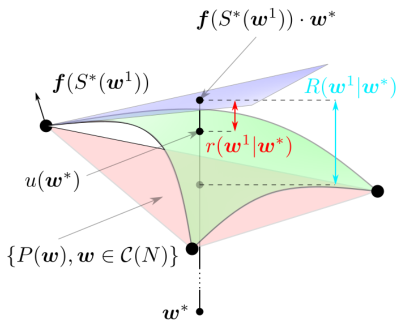
$$\max_{\boldsymbol{w}^* \in \mathcal{C}(N)} \min_{\boldsymbol{w}' \in N} r(\boldsymbol{w}'|\boldsymbol{w}^*) \leq \bar{R}(N).$$

Proof. For $\boldsymbol{w}^* \in \mathcal{C}(N)$, let $\boldsymbol{w}'_1 = \arg\min_{\boldsymbol{w}' \in N} r(\boldsymbol{w}'|\boldsymbol{w}^*)$, $\boldsymbol{w}'_2 = \arg\min_{\boldsymbol{w}' \in N} R(\boldsymbol{w}'|\boldsymbol{w}^*)$. Note that $\boldsymbol{w}'_1 = \boldsymbol{w}'_2$. Indeed, we have $R(\boldsymbol{w}'_2|\boldsymbol{w}^*) \leq R(\hat{\boldsymbol{w}}|\boldsymbol{w}^*)$ for all $\hat{\boldsymbol{w}} \in N$ if and only if $\boldsymbol{w}^* \cdot \boldsymbol{f}(s^*(\boldsymbol{w}'_2)) \leq \boldsymbol{w}^* \cdot \boldsymbol{f}(s^*(\hat{\boldsymbol{w}}))$ which is equivalent to $r(\boldsymbol{w}'_2|\boldsymbol{w}^*) \leq r(\hat{\boldsymbol{w}}|\boldsymbol{w}^*)$.

By Observation 2, $u(\boldsymbol{w})$ is concave on $\mathcal{C}(N)$ implying that for $\boldsymbol{w} \in \mathcal{C}(N)$, $u(\boldsymbol{w}) \geq P(\boldsymbol{w})$ (see Figure 3). Thus, by (6), $r(\boldsymbol{w}'|\boldsymbol{w}^*) \leq R(\boldsymbol{w}'|\boldsymbol{w}^*)$ for $\boldsymbol{w}^* \in \mathcal{C}(N)$. The result follows.



(a) $P(\boldsymbol{w}), \mathcal{C}(N)$ for $N = \{\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{w}^3\}$


$\boldsymbol{f}(S^*(\boldsymbol{w}^1)) \cdot \boldsymbol{w}^*$
$\boldsymbol{f}(S^*(\boldsymbol{w}^1))$
$R(\boldsymbol{w}^1|\boldsymbol{w}^*)$
$r(\boldsymbol{w}^1|\boldsymbol{w}^*)$
$u(\boldsymbol{w}^*)$
$\{P(\boldsymbol{w}), \boldsymbol{w} \in \mathcal{C}(N)\}$
$\boldsymbol{w}^*$

(b) Regret r, and regret bound R

Fig. 3: Illustrative example of a neighborhood and its properties.

□

The value of $\bar{R}(N)$ with corresponding weight $\bar{\boldsymbol{w}}(N)$ from (10) can be obtained by solving the following LP:

$$\max_{x \in \mathbb{R}, \boldsymbol{w} \in \mathbb{R}^n} x - P(\boldsymbol{w})$$
$$s.t. \begin{bmatrix} f_1(s^*(\boldsymbol{w}^1)) & \dots & f_n(s^*(\boldsymbol{w}^1)) & -1 \\ \vdots & \ddots & \vdots & \vdots \\ f_1(s^*(\boldsymbol{w}^n)) & \dots & f_n(s^*(\boldsymbol{w}^n)) & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ x \end{bmatrix} \succcurlyeq 0, \quad (12)$$
$$\boldsymbol{w} \in \mathcal{C}(N).$$

If $(x^*, \boldsymbol{w}^*)$ solves (12), the optimal cost is given by $x^* - P(\boldsymbol{w}^*) = \bar{R}(N)$, and $\boldsymbol{w}^* = \bar{\boldsymbol{w}}(N)$. Indeed, for any feasible $x, \boldsymbol{w}$, it holds that $x \leq \min_{\boldsymbol{w}^i \in N} \boldsymbol{f}(s(\boldsymbol{w}^i)) \cdot \boldsymbol{w}$. Since x is maximized, this will hold with equality for $x^*, \boldsymbol{w}^*$. Therefore, the cost of (12) is equivalent to $\max_{\boldsymbol{w} \in \mathcal{C}(N)} \min_{\boldsymbol{w}^i \in N} R(\boldsymbol{w}^i|\boldsymbol{w}) = \bar{R}(N)$. A detailed explanation of the implementation for Equation (12) is provided in the supplementary materials.

In (12), if $P(\boldsymbol{w})$ is replaced with $u(\boldsymbol{w})$, then the resulting problem is solved by $x^*, \boldsymbol{w}^*$ if and only if $\boldsymbol{w}^*$ maximizes the regret in $\mathcal{C}(N)$ given the neighborhood N. This problem is not linear and would require solving the LSMOP in (3) potentially many times. Using the LP in (12) our method is summarized in Algorithm 1 described in the next section. We iteratively partition $\mathcal{W}$ into smaller neighborhoods, adding weights that result in the largest upper bound of regret.

A. Algorithm Description

Algorithm 1 creates and maintains a set $\mathcal{N}$ of neighborhoods $N \subset \mathcal{W}$. Each $N \in \mathcal{N}$ is a set of weights $N = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^n\}$ where $\boldsymbol{w}^i \in \Omega, i = 1, \dots, n$. We compute $\bar{R}(N)$ and $\bar{\boldsymbol{w}}(N)$ with the LP in (12) for N using the set of objective vectors $\mathcal{F}(N)$. The algorithm begins with a single neighborhood whose weights are the n canonical basis elements of $\mathbb{R}^n$ (Line 1). The solutions for these basis weights correspond to the single-objective solutions for all n objective functions. We assume that the budget K is at least the number of objective functions n. The algorithm then iteratively selects the neighborhood N in $\mathcal{N}$ with the largest upper bound of regret (Line 5), and adds its regret weight $\bar{\boldsymbol{w}}(N)$ to Ω (Line 8). It then splits and replaces N with at most n smaller neighborhoods (Lines 11-15) formed by iteratively replacing elements in N with $\bar{\boldsymbol{w}}(N)$ (Line 12). Finally, the algorithm returns the set Ω as well as an upper bound on the regret given Ω (Line 16). Two steps of the algorithm are illustrated in Figure 4, starting with a single neighborhood N_1 in (a) which is then split around $\boldsymbol{w}^3 = \bar{\boldsymbol{w}}(N_1)$ into two new neighborhoods $\mathcal{N} = \{N_2, N_3\}$ in (b). Since $\bar{R}(N_3) > \bar{R}(N_2)$, N_3 is split around $\boldsymbol{w}^5 = \bar{\boldsymbol{w}}(N_3)$ in (c). Finally, in (d), the red area shows the regret given Ω.

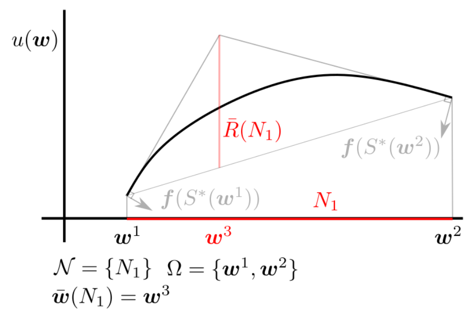
Observe that Algorithm 1 may be modified to compute a set Ω given a desired maximum regret $r_{\max} > 0$. This could be accomplished by replacing the input K with $r_{\max}$, and the stopping criteria in Line 4 with a while loop that runs until $\bar{R} \leq r_{\max}$. Here, $\bar{R}$ represents the maximum regret over all neighborhoods $\bar{R} = \max_{N \in \mathcal{N}} \bar{R}(N)$ and can be maintained in the body of the loop. Since $\mathcal{N}$ forms a partition of $\mathcal{W}$, we are guaranteed that the regret of any weight given Ω is no more than $\bar{R}$ by Theorem 2. Therefore, if Algorithm 1 terminates when $\bar{R} \leq r_{\max}$, the desired maximum regret is achieved.

B. Algorithm Properties

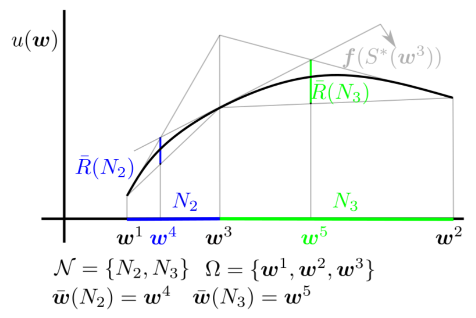
We observe several beneficial properties to the approach outlined above.

Observation 3 (Runtime)**.** For a budget of K, Algorithm 1 will require that the LSMOP in (3) be solved at most K times, once per element of Ω. Let $t_{\texttt{LSMOP}}$ be the runtime to solve (3). Using the interior-point method allows for solving LPs in $O(a^2 b^{3/2})$ time with a being the number of variables, and b the number of constraints. The LP in (12) has $2n$ variables and $2n$ constraints with n being the number of objective functions (see Equation (21) in the appendix for details). Thus, the runtime of Algorithm 1 is $O(K(t_{\texttt{LSMOP}} + n^3))$.

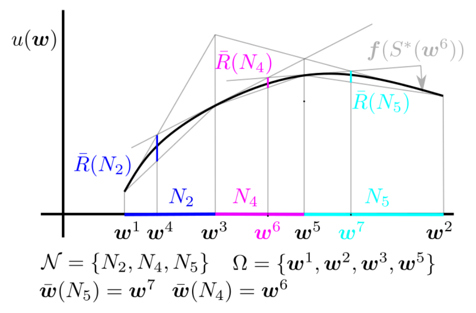
Observation 4 (Regret bound)**.** The value of $\bar{R}(N)$ returned by the algorithm is an upper-bound on the value of regret in the original problem (7). Indeed, initially $\mathcal{N} = \{\Omega\}$ and $\mathcal{C}(\Omega) = \mathcal{W}$. At every iteration, a neighborhood $N \in \mathcal{N}$ is split into at most n sub-neighborhoods such whose convex hulls are disjoint and collectively form $\mathcal{C}(N)$. Then, by Theorem 2, it holds that $\max_{\boldsymbol{w}^* \in \mathcal{W}} \min_{\boldsymbol{w}' \in \Omega} r(\boldsymbol{w}'|\boldsymbol{w}^*) \leq \max_{N \in \mathcal{N}} \bar{R}(N)$.


$u(\boldsymbol{w})$
$\bar{R}(N_1)$
$f(S^*(w^2))$
$f(S^*(w^1))$
N_1
$\boldsymbol{w}^1$ $\boldsymbol{w}^3$ $\boldsymbol{w}^2$
$\mathcal{N} = \{N_1\}$ $\Omega = \{\boldsymbol{w}^1, \boldsymbol{w}^2\}$
$\bar{\boldsymbol{w}}(N_1) = \boldsymbol{w}^3$

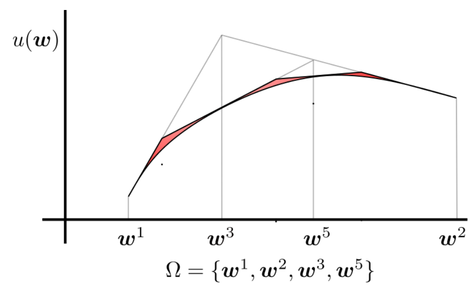
(a) First iteration


$u(\boldsymbol{w})$
$f(S^*(w^3))$
$\bar{R}(N_3)$
$\bar{R}(N_2)$
N_2 N_3
$\boldsymbol{w}^1$ $\boldsymbol{w}^4$ $\boldsymbol{w}^3$ $\boldsymbol{w}^5$ $\boldsymbol{w}^2$
$\mathcal{N} = \{N_2, N_3\}$ $\Omega = \{\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{w}^3\}$
$\bar{\boldsymbol{w}}(N_2) = \boldsymbol{w}^4$ $\bar{\boldsymbol{w}}(N_3) = \boldsymbol{w}^5$

(b) Second iteration


$u(\boldsymbol{w})$
$\bar{R}(N_4)$
$f(S^*(w^6))$
$\bar{R}(N_5)$
$\bar{R}(N_2)$
N_2 N_4 N_5
$\boldsymbol{w}^1$ $\boldsymbol{w}^4$ $\boldsymbol{w}^3$ $\boldsymbol{w}^6$ $\boldsymbol{w}^5$ $\boldsymbol{w}^7$ $\boldsymbol{w}^2$
$\mathcal{N} = \{N_2, N_4, N_5\}$ $\Omega = \{\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{w}^3, \boldsymbol{w}^5\}$
$\bar{\boldsymbol{w}}(N_5) = \boldsymbol{w}^7$ $\bar{\boldsymbol{w}}(N_4) = \boldsymbol{w}^6$

(c) Third iteration


$u(\boldsymbol{w})$
$\boldsymbol{w}^1$ $\boldsymbol{w}^3$ $\boldsymbol{w}^5$ $\boldsymbol{w}^2$
$\Omega = \{\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{w}^3, \boldsymbol{w}^5\}$

(d) Regret after three iterations

Fig. 4: Illustration of the first two iterations of Algorithm 1.

Lemma 2 (Algorithm Completeness)**.** The set $\Omega(K)$ returned by Algorithm 1 on input K asymptotically and monotonically approaches a set with zero regret as $K \to \infty$.

Proof. The proof follows by contradiction: If the result did not hold, then a neighborhood $N \in \mathcal{N}$ would fail to decrease in size when split (Lines 11-15). This in turn requires that there is a $\boldsymbol{w}^i \in N \subseteq \Omega(K)$ such that $||\bar{\boldsymbol{w}}(N) - \boldsymbol{w}^i||_2$ decreases to 0. Since $\bar{\boldsymbol{w}}(N)$ is chosen to maximize $\bar{R}(N)$, this can only occur if $\bar{R}(N)$ is unbounded at $\boldsymbol{w}^i$ implying that the objectives are unbounded at $\boldsymbol{w}^i$ in violation of Assumption 2. □

Observation 5 (Optimality for Discrete Solution Spaces)**.** In the case where the solution space $\mathcal{S}$ of the LSMOP in (3) is discrete, the function $u(\boldsymbol{w})$ will be piece-wise linear by Lemma 1. If K is at least the number of linear pieces of $u(\boldsymbol{w})$, the solution set $\Omega(K)$ has zero maximum regret and is the smallest set that accomplishes this. Indeed $\Omega(K)$ is comprised of exactly one weight in each linear piece of $u(\boldsymbol{w})$. Since the regret is defined by the error of a first order approximation, this value is exactly zero on each linear piece.

In the next section, we extend Algorithm 1 to the case where Assumption 1 does not hold. That is, where no exact solver for the optimization problem 3. This is followed by an extension to Algorithm 1 for the case when a prior belief about the optimal weights of a user are known (See Section VI). It should be noted that both extensions reduce to Algorithm 1 when the sub-optimal solver is optimal, or the prior belief is uniform (respectively). Moreover, these extensions can be used in tandem for the case where Assumption 1 is violated *and* a prior belief about the optimal user weights is known. This last is not formally stated, but is trivially obtained from the descriptions of each extension to follow.

V. Extension to Suboptimal Solvers

In this section, we illustrate how Algorithm 1 can be adapted to accept sub-optimal solvers for the underlying LSMOP. Instead of $s^*(\boldsymbol{w})$, an optimal solution to an LSMOP for weights $\boldsymbol{w}$ (see (5)), consider a feasible solution $\hat{s}(\boldsymbol{w}) \in \mathcal{S}$ obtained from a sub-optimal solver. Following Section II and given any weight vector $\boldsymbol{w} \in \mathcal{W}$, we let $\hat{u}(\boldsymbol{w}) = c(\hat{s}(\boldsymbol{w}), \boldsymbol{w})$ (see (2)) be the cost of the solution $\hat{s}(\boldsymbol{w})$ given weights $\boldsymbol{w}$. Observe that $\hat{u}$ is identical to the function u (see (3)) except that a sub-optimal solution $\hat{s}(\boldsymbol{w})$ is now in place of $s^*(\boldsymbol{w})$.

A. Look-up Table Solutions

Algorithm 1 relies heavily on the concavity of $u(\boldsymbol{w})$ as a function of $\boldsymbol{w}$, which is guaranteed from the optimality of $s^*(\boldsymbol{w})$. Therefore, the function $\hat{u}(\boldsymbol{w})$ — the cost of a solution computed using a sub-optimal solver — is not guaranteed to be concave. In this section, we begin with a technique to replace $\hat{u}(\boldsymbol{w})$ with a concave function. The high level idea is to maintain a look-up table, *i.e.,* a set of discovered solutions $\mathcal{T}$. Given any weight $\boldsymbol{w} \in \mathcal{W}$, we define a new solver $s_{\mathcal{T}}$ to select the best solution in $\mathcal{T}$. In detail, given a set of sampled weights $\Omega = \{\boldsymbol{w}_1, \dots, \boldsymbol{w}_r\} \subset \mathcal{W}, r \geq 1$, we denote by $\mathcal{T}(\Omega) = \{\hat{s}(\boldsymbol{w}), \boldsymbol{w} \in \Omega\}$ the set of all solutions to weights $\boldsymbol{w} \in \Omega$ obtained via the sub-optimal solver $\hat{s}$.

For any subset of weights $\Omega \subset \mathcal{W}$ and its associated set of discovered solutions $\mathcal{T}(\Omega)$, we define the *best discovered solution* to any weight $\boldsymbol{w} \in \mathcal{W}$ as

$$s_{\mathcal{T}}(\boldsymbol{w}) = \underset{s \in \mathcal{T}(\Omega)}{\text{argmin}}\ c(s, \boldsymbol{w}). \tag{13}$$

Further, we define a *best discovered cost* function $u_{\mathcal{T}}(\boldsymbol{w}) = c(s_{\mathcal{T}}(\boldsymbol{w}), \boldsymbol{w})$. Finally, we define the *best discovered regret* function for any weights $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$:

$$r_{\mathcal{T}}(\boldsymbol{w}'|\boldsymbol{w}^*) = \boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}')) - u_{\mathcal{T}}(\boldsymbol{w}^*)$$

We now prove that for any subset Ω, the best discovered cost function $u_{\mathcal{T}}(\boldsymbol{w})$ has the same properties as the optimal cost function $u(\boldsymbol{w})$ from (3) that are leveraged by Algorithm 1. For the remainder of this section, we assume that Ω is a non-empty countable subset of $\mathcal{W}$, and we let $\mathcal{T}(\Omega)$ denote the associated set of discovered solutions.

Lemma 3 (Best Discovered Cost Properties)**.** For any set of weights Ω, it holds that $u_{\mathcal{T}}(\boldsymbol{w})$ is a concave continuous function of $\boldsymbol{w} \in \mathcal{W}$ for which $\partial u_{\mathcal{T}}(\boldsymbol{w})$ is a sub-gradient given by $\boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}))$ for all $\boldsymbol{w} \in \mathcal{W}$. Further, for all $\boldsymbol{w} \in \Omega$, $u_{\mathcal{T}}(\boldsymbol{w}) \leq \hat{u}(\boldsymbol{w})$.

Proof. Begin by observing that for any weights $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathcal{W}$, it must hold that $u_{\mathcal{T}}(\boldsymbol{w}_1) \leq c(s_{\mathcal{T}}(\boldsymbol{w}_2), \boldsymbol{w}_1)$. Indeed, by the definition of $u_{\mathcal{T}}$, if the observation does not hold, then

$$u_{\mathcal{T}}(\boldsymbol{w}_1) = c(s_{\mathcal{T}}(\boldsymbol{w}_1), \boldsymbol{w}_1) > c(s_{\mathcal{T}}(\boldsymbol{w}_2), \boldsymbol{w}_1),$$

implying that $s_{\mathcal{T}}(\boldsymbol{w}_1)$ is not a minimizer of $c(s, \boldsymbol{w}_1)$ over $\mathcal{T}(\Omega)$ (since $s_{\mathcal{T}}(\boldsymbol{w}_2) \in \mathcal{T}(\Omega)$ by (13)), which is a contradiction of the definition of $s_{\mathcal{T}}(\boldsymbol{w}_1)$ from (13). Therefore, $u_{\mathcal{T}}(\boldsymbol{w}_1) \leq c(s_{\mathcal{T}}(\boldsymbol{w}_2), \boldsymbol{w}_1)$ for all $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathcal{W}$. Next, we establish the concavity of $u_{\mathcal{T}}(\boldsymbol{w})$.

For any $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathcal{W}$ and $\lambda_1 \in [0, 1], \lambda_2 = 1 - \lambda_1$, let $\boldsymbol{w} = \lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2$. From (13), and the definition of $u_{\mathcal{T}}$,

$$\begin{aligned} u_{\mathcal{T}}(\boldsymbol{w}) &= c(s_{\mathcal{T}}(\boldsymbol{w}), \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w})) \\ &= \lambda_1 \boldsymbol{w}_1 \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w})) + \lambda_2 \boldsymbol{w}_2 \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w})) \\ &= \lambda_1 c(s_{\mathcal{T}}(\boldsymbol{w}), \boldsymbol{w}_1) + \lambda_2 c(s_{\mathcal{T}}(\boldsymbol{w}), \boldsymbol{w}_2) \\ &\geq \lambda_1 u_{\mathcal{T}}(\boldsymbol{w}_1) + \lambda_2 u_{\mathcal{T}}(\boldsymbol{w}_2), \end{aligned}$$

where the inequality holds by the observation made at the top of the proof. This establishes the concavity of $u_{\mathcal{T}}(\boldsymbol{w})$. The continuity of $u_{\mathcal{T}}(\boldsymbol{w})$ therefore follows from the proof of Lemma 1. Next we show $\partial u_{\mathcal{T}}(\boldsymbol{w}^*) = \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*)),\ \forall \boldsymbol{w}^* \in \mathcal{W}$. Let $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$, then from our first observation,

$$\begin{aligned} &u_{\mathcal{T}}(\boldsymbol{w}') - u_{\mathcal{T}}(\boldsymbol{w}^*) = \boldsymbol{w}' \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}')) - \boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*)) \\ &\leq \boldsymbol{w}' \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*)) - \boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*)) \\ &= \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*)) \cdot (\boldsymbol{w}' - \boldsymbol{w}^*) \end{aligned}$$

Which establishes $\boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*))$ as a sub-gradient of $u_{\mathcal{T}}(\boldsymbol{w})$ at $\boldsymbol{w}^*$. Finally, we show that for all $\boldsymbol{w}' \in \mathcal{T}(\Omega)$, $u_{\mathcal{T}}(\boldsymbol{w}') \leq \hat{u}(\boldsymbol{w}')$. Since $\boldsymbol{w}' \in \Omega$, it must hold that $\hat{s}(\boldsymbol{w}') \in \mathcal{T}(\Omega)$ by the definition of $\mathcal{T}(\Omega)$. Therefore, from (13) and the definition of $u_{\mathcal{T}}(\boldsymbol{w})$, it must hold that $u_{\mathcal{T}}(\boldsymbol{w}') = c(s_{\mathcal{T}}(\boldsymbol{w}'), \boldsymbol{w}') \leq c(\hat{s}(\boldsymbol{w}'), \boldsymbol{w}') = \hat{u}(\boldsymbol{w}')$ which completes the proof. ☐

Lemma 3 implies that the cost function $u_{\mathcal{T}}(\boldsymbol{w})$ has the same properties as the optimal cost function $u(\boldsymbol{w})$ (established in Section III) that made Algorithm 1 possible.

Algorithm 2: MRPS WITH HEURISTIC SOLVER

```
Input: A suboptimal $\hat{s}$ solver to approximately
       compute $s(\boldsymbol{w})$; a budget $K \geq n$
Output: Sampled weights $\Omega$ and maximum regret
1  $\Omega \leftarrow \{\boldsymbol{e}^i, i = 1, \ldots, n\}$
2  $\mathcal{T}(\Omega) \leftarrow \{\hat{s}(\boldsymbol{w}), \boldsymbol{w} \in \Omega\}$
3  $\mathcal{N} \leftarrow \{\Omega\}$
4  for $k = n$ to $K$ do
5      $N$ = neighborhood in $\mathcal{N}$ with maximum $\bar{R}(N)$ //
         Here, $\bar{R}(N)$ is computed from (10) with $s^*$
         replaced with $s_{\mathcal{T}}$ defined in (13)
6      if $\bar{R}(N) = 0$ then
7          $N \leftarrow$ Largest_Neighbourhood
8          $\bar{\boldsymbol{w}} \leftarrow$ Mean_Weight($N$)
9      $\Omega \leftarrow \Omega \cup \{\bar{\boldsymbol{w}}(N)\}$
10     $\mathcal{T}(\Omega) \leftarrow \mathcal{T}(\Omega) \cup \hat{s}(\bar{\boldsymbol{w}}(N))$
11     Lines 10-15 of Algorithm 1 with $s_{\mathcal{T}}$ replacing $s^*$
12 return $\Omega$ and the maximum value of $\bar{R}(N)$ over all
       $N \in \mathcal{N}$
```

B. Adapted Algorithm

Leveraging Lemma 3, we now propose minor modifications to Algorithm 1 that will allow it to accept sub-optimal solvers. The procedure is outlined in Algorithm 2 and closely follows Algorithm 1. The primary modifications are as follows: In line 2 of Algorithm 2, we initialize the set of discovered solutions $\mathcal{T}(\Omega)$. In Line 5, we replace the definition of $\bar{R}(N)$ for a neighborhood $N \in \mathcal{N}$ that is given in (10) with a version where $s^*(\boldsymbol{w})$ is replaced with $s_{\mathcal{T}}(\boldsymbol{w})$ given in (13) for any weight $\boldsymbol{w} \in \mathcal{W}$. Following the definitions in Section IV, recall that given a set of weights $\Omega \subseteq \mathcal{W}$ and a neighborhood N associated with weights in Ω, the function $\bar{R}(N)$, and its associated weight vector $\bar{\boldsymbol{w}}(N)$ are defined in (10) relative to a function $R(\boldsymbol{w}'|\boldsymbol{w}^*)$ for weights $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$. This latter function is defined as $R(\boldsymbol{w}'|\boldsymbol{w}^*) = \boldsymbol{w}^* \cdot \boldsymbol{f}(s^*(\boldsymbol{w}')) - P(\boldsymbol{w}^*)$ for a specific plane P defined in that section. It was shown in Theorem 2 that $\bar{R}(N)$ is an upper bound for the maximum regret in the neighborhood N given Ω. In Algorithm 2, we simply replace $s^*(\boldsymbol{w})$ with $s_{\mathcal{T}}(\boldsymbol{w})$ in all preceding function definitions. In a manner identical to the proof of Theorem 2, it is easily verified from the results of Lemma 3 that the modified function $\bar{R}(N)$ is an upper bound on the maximum best known regret $r_{\mathcal{T}}$ in the neighborhood N. Further, the values $\bar{R}(N), \bar{\boldsymbol{w}}(N)$ may still be obtained by solving the linear program (12) with s^* replaced with $s_{\mathcal{T}}$.

Next, Lines 6-8 replace the conditional stopping criteria in Lines 6-7 of Algorithm 1. In practice, we have noticed that if the sub-optimal solver $\hat{s}$ is a poor estimate of the optimal solution s^*, then using a conditional stopping criteria $\bar{R}(N) = 0$ for all $N \in \mathcal{N}$ tends to cause the Algorithm to terminate prematurely. This is because there may exist neighborhoods N and weights $\boldsymbol{w}$ in the convex hull of N such that $c(\hat{s}(\boldsymbol{w}), \boldsymbol{w}) < \min_{\boldsymbol{w}' \in N} u_{\mathcal{T}}(\boldsymbol{w}')$ which cannot happen when using an optimal solver. Finally, in Line 10, when a weight $\bar{\boldsymbol{w}}$ is added to Ω, its corresponding sub-optimal solution $\hat{s}(\bar{\boldsymbol{w}})$ is added to $\mathcal{T}(\Omega)$. The remainder of the Algorithm is unchanged from Algorithm 1. Observe that if $\hat{s}(\boldsymbol{w}) = s^*(\boldsymbol{w})$ for all $\boldsymbol{w} \in \mathcal{W}$, that is, if the 'sub-optimal' solver is in fact optimal, then Algorithm 2 reduces to Algorithm 1 by construction.

C. Extended Error Bound

We conclude with an error bound guarantee in the case that solutions $\hat{s}(\boldsymbol{w})$ are derived from a sub-optimal solver with a known approximation factor. Let Ω be a set of weight vectors computed by Algorithm 2, and let $\mathcal{N}$ be the set of partitioning neighborhoods associated with Ω. Finally, let

$$\beta = \max_{N \in \mathcal{N}} \max_{\boldsymbol{w} \in N} \frac{\bar{R}(N)}{u_{\mathcal{T}}(\boldsymbol{w})}. \tag{14}$$

Intuitively, β represents the maximum relative regret over all neighborhoods $N \in \mathcal{N}$. We write β as a ratio opposed to a difference as in (6) to derive an approximation factor. Observe that the value of β is easily computed since each neighborhood N is a discrete set of weights in Ω. Thus, for each $\boldsymbol{w} \in N$ and each $N \in \mathcal{N}$, $u_{\mathcal{T}}(\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}))$. We offer the following result.

Theorem 3 (Approximation factor)**.** Let Ω be the set of weights computed by Algorithm 2 with input solutions $\hat{s}(\boldsymbol{w})$ and β be given by (14). If the solutions $\hat{s}(\boldsymbol{w})$ are derived from a solver with known approximation factor α, then, for every $\boldsymbol{w}^* \in \mathcal{W}$, there exists a $\boldsymbol{w}' \in \Omega$ such that

$$c(s_{\mathcal{T}}(\boldsymbol{w}'), \boldsymbol{w}^*) \leq \alpha(\beta + 1)u(\boldsymbol{w}^*).$$

Proof. For any $\boldsymbol{w}^* \in \mathcal{W}$, there must exist a neighborhood N such that $\boldsymbol{w}^* \in \mathcal{C}(N)$ since $\mathcal{C}(N), N \in \mathcal{N}$ forms a partition of $\mathcal{W}$. Since $N \subseteq \Omega$ it must hold by Lemma 3 and the assumption that $\hat{s}(\boldsymbol{w})$ is derived from a solver that approximates $s^*(\boldsymbol{w})$ to within a factor of α of the resulting cost, that for all $\boldsymbol{w} \in N$, $u_{\mathcal{T}}(\boldsymbol{w}) \leq \hat{u}(\boldsymbol{w}) \leq \alpha u(\boldsymbol{w})$. Let P denote the hyperplane passing through $\{u(\boldsymbol{w}), \boldsymbol{w} \in N\}$ and $P_{\mathcal{T}}$ the hyperplane passing through $\{u_{\mathcal{T}}(\boldsymbol{w}), \boldsymbol{w} \in N\}$. Observe that since $u(\boldsymbol{w}) \geq \alpha^{-1} u_{\mathcal{T}}(\boldsymbol{w})$ for all $\boldsymbol{w} \in N$ and $P, P_{\mathcal{T}}$ are planes, it must hold that $\alpha P(\boldsymbol{w}) \geq P_{\mathcal{T}}(\boldsymbol{w}), \forall \boldsymbol{w} \in \mathcal{C}(N)$. From the definition of $\bar{R}(N)$, there must exist a weight $\boldsymbol{w}' \in N$ with $\boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}')) - P_{\mathcal{T}}(\boldsymbol{w}^*) \leq \bar{R}(N)$ implying that

$$\begin{aligned} \boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}')) &\leq \bar{R}(N) + P_{\mathcal{T}}(\boldsymbol{w}^*) \\ &\leq \bar{R}(N) + \alpha P(\boldsymbol{w}^*) \\ &\leq \bar{R}(N) + \alpha u(\boldsymbol{w}^*), \end{aligned} \tag{15}$$

where the final inequality holds from the concavity of $u(\boldsymbol{w})$ (see Observation 2) implying that $u(\boldsymbol{w})$ lies below the plane $P(\boldsymbol{w})$ for $\boldsymbol{w} \in \mathcal{C}(N)$. Letting $\boldsymbol{w}'' = \arg\min_{\boldsymbol{w} \in N} u(\boldsymbol{w})$, we observe by the concavity of $u(\boldsymbol{w}), u_{\mathcal{T}}(\boldsymbol{w})$ (see Lemma 3) that $u(\boldsymbol{w}) \geq u(\boldsymbol{w}'')$ and $u_{\mathcal{T}}(\boldsymbol{w}) \geq \min_{\boldsymbol{w} \in N} u_{\mathcal{T}}(\boldsymbol{w})$. Since $u_{\mathcal{T}}(\boldsymbol{w})$ is within a factor of α of $u(\boldsymbol{w})$ for all $\boldsymbol{w} \in N$ and $\boldsymbol{w}'' \in N$ by definition, it must hold that $u(\boldsymbol{w}^*) \geq u(\boldsymbol{w}'') \geq \alpha^{-1} u_{\mathcal{T}}(\boldsymbol{w}'') \geq \alpha^{-1} \min_{\boldsymbol{w} \in N} u_{\mathcal{T}}(\boldsymbol{w})$. Thus,

$$\frac{\bar{R}(N)}{u(\boldsymbol{w}^*)} \leq \alpha \frac{\bar{R}(N)}{\min\limits_{\boldsymbol{w} \in N} u_{\mathcal{T}}(\boldsymbol{w})} = \alpha \max_{\boldsymbol{w} \in N} \frac{\bar{R}(N)}{u_{\mathcal{T}}(\boldsymbol{w})} \leq \alpha\beta. \tag{16}$$

Thus, from (15), (16),

$$\frac{\boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}'))}{u(\boldsymbol{w}^*)} \leq \alpha(\beta + 1).$$

Finally, noting that $c(s_{\mathcal{T}}(\boldsymbol{w}'), \boldsymbol{w}^*) = \boldsymbol{w}^* \cdot \boldsymbol{f}(s_{\mathcal{T}}(\boldsymbol{w}^*))$ completes the proof. □

In this section, we extended the algorithm introduced in Section IV to include sub-optimal solvers. Moreover, when the suboptimal solver is an approximation algorithm we retain a bound on the error of a solution set. In the following section, we consider another extension to include stochastic settings.

VI. Extension to Stochastic Settings

In this section, we consider two cases of stochastic inputs to the problem: first, there is a bias representing the likely relevance of some weights over others. Second, the constraints defining the planning problem are random, *i.e.,* different instances have to be considered.

A. Weights as a Random Variable

In practice, some regions of the weight space $\mathcal{W}$ may correspond to Pareto-optimal solutions that are less relevant, *e.g.,* there is prior information available on what weights may be likely to represent of a user's preference. In the example in Figure 1, feasible solutions were trajectories between fixed start and goal configurations, while the objectives were trajectory length and comfort. It may be very unlikely for any user to *only* care about comfort.

In this section we consider that a probability density function (PDF) $g(\boldsymbol{w})$ over $\mathcal{W}$ is given. The problem then becomes one of computing a set of weights Ω that approximates solutions corresponding to *probable* weights, *i.e.,* that minimizes the maximum regret *discounted* by the prior $g(\boldsymbol{w})$.

First, we introduce a notation shorthand: Given a set of sampled weights Ω and a weight $\boldsymbol{w}$ in $\mathcal{W}$, we define the regret of $\boldsymbol{w}$ using Ω as

$$r^{\Omega}(\boldsymbol{w}) = \min_{\boldsymbol{w}' \in \Omega} r(\boldsymbol{w}' | \boldsymbol{w}) \tag{17}$$

The goal of this section is to compute a set of weights that address Problem 1 while considering the prior belief expressed by $g(\boldsymbol{w})$. To this end, consider two optimization problems:

$$\min_{\substack{\Omega \\ |\Omega| \leq K}} \int_{\mathcal{W}} r^{\Omega}(\boldsymbol{w}^*) g(\boldsymbol{w}^*) d\boldsymbol{w}^*, \tag{18a}$$

$$\min_{\substack{\Omega \\ |\Omega| \leq K}} \operatorname*{ess\,sup}_{\boldsymbol{w}^* \in \mathcal{W}} r^{\Omega}(\boldsymbol{w}^*) g(\boldsymbol{w}^*). \tag{18b}$$

We refer to the term $r^{\Omega}(\boldsymbol{w}^*) g(\boldsymbol{w}^*)$ as the *discounted regret*. The problem in (18a) seeks to compute a set Ω that minimizes the expected regret $\mathbb{E}[r^{\Omega}(\boldsymbol{w}^*)]$, while the second (18b) minimizes the *essential supremum* (supremum up to a set of measure zero) of the discounted regret.

While both problems have their applications, we focus on solving (18b). The main shortcoming of (18a) is that it can lead to placing samples to reduce the regret for a large set of weights with low probability. However, as we have seen in Figure 1, there can be a large subset of samples that yields similar – or even identical – corresponding optimal solutions. Minimizing the expected regret can then overly favour minimizing the regret for a large subset of weights, regardless of the incurred regret.

B. Probabilistic Sampling Algorithm

We now adapt Algorithm 1 to solve the probabilistic problem posed in (18b). To this end, for any neighborhood N of weights with convex hull $\mathcal{C}(N)$, let

$$p(N) = \int_{\mathcal{C}(N)} g(\boldsymbol{w}) d\boldsymbol{w},$$

denote the probability that the random variable $\boldsymbol{w}^*$ lies in $\mathcal{C}(N)$. We propose a simple, yet effective change to Algorithm 1 in order to handle probabilities over weights, and denote the new algorithm as MRPS-P. To solve for the objective in (18b), we replace $\bar{R}(N)$ in Lines 5, 16 of Algorithm 1 with $\bar{R}^p(N) = p(N)\bar{R}(N)$. That is, the neighborhood N we select to place an additional sampled weight is the one with the largest regret bound $\bar{R}(N)$, *discounted* by the probability $p(N)$ that $\mathcal{C}(N)$ contains $\boldsymbol{w}^*$. In this way, we avoid adding weights to Ω that lie in regions that are unlikely to contain $\boldsymbol{w}^*$ unless the regret of not including such a weight is sufficiently large. Letting $\Omega, \bar{R}^p$ denote the outputs of the modified version of Algorithm 1, we offer the following results with regard to the objective (18b):

Lemma 4 (Worst Case Discounted Regret)**.** The maximum discounted regret $\bar{R}^p$ returned by Algorithm MRPS-P is an upper bound on the objective in (18b):

$$\operatorname*{ess\,sup}_{\boldsymbol{w}^* \in \mathcal{W}} r^{\Omega}(\boldsymbol{w}^*) g(\boldsymbol{w}^*) \leq \bar{R}^p. \tag{19}$$

Proof. Since $\mathcal{N}$ forms a partition of $\mathcal{W}$ (Section IV-B), there exists a neighborhood $N \in \mathcal{N}$ with $\boldsymbol{w}^* \in \mathcal{C}(N)$. Therefore,

$$\begin{aligned} r^{\Omega}(\boldsymbol{w}^*) g(\boldsymbol{w}^*) &\leq \int_{\mathcal{C}(N)} r^{\Omega}(\boldsymbol{w}) g(\boldsymbol{w}) d\boldsymbol{w} \\ &\leq \bar{R}(N) \int_{\mathcal{C}(N)} g(\boldsymbol{w}) d\boldsymbol{w} = \bar{R}^p(N) \leq \bar{R}^p. \end{aligned} \tag{20}$$

The first inequality holds by Theorem 2, while the final inequality holds since modified Algorithm 1 returns the maximum value $\bar{R}^p$ over all neighborhoods in $\mathcal{N}$. □

In addition to providing a bound on the objective of (18b), we observe that $\bar{R}^p$ also provides a bound on the objective of (18a). Indeed, by Lemma 4,

$$\mathbb{E}[r^{\Omega}(\boldsymbol{w}^*)] = \int_{\mathcal{W}} r^{\Omega}(\boldsymbol{w}) g(\boldsymbol{w}) d\boldsymbol{w} \leq \bar{R}^p \int_{\mathcal{W}} d\boldsymbol{w} = \bar{R}^p.$$

Thus, with a simple modification, we can adapt the MRPS algorithm to incorporate prior beliefs over weights. The algorithm then greedily minimizes an upper bound of the discounted regret. Observe that if the PDF is uniform, that is, if all weights are equally likely to represent a user, then the proposed extension to the MRPS algorithm, reduces to the original MRPS algorithm. In the next section, we extend Algorithm 1 to account for multiple problem instances.

C. Constraints as a Random Variable

As a final extension of Algorithm 1, we treat the constraints $\mathcal{S}$ in (1) as random variables. This encompasses the case where multiple instances of (3) are possible. Consider, for example, the problem detailed in Figure 1. Here, we compute trajectories between a *fixed* start and goal to minimize a tradeoff between travel time and discomfort. However, changing the positions of the start and goal would result in a different instance of problem (3) with a (potentially) different set of weights Ω computed by Algorithm 1.

This may seem to limit the applicability of the approach presented here. However, there is a simple extension. We define a random variable I existing in a sample space $\mathcal{I}$ which represents a possible instance of the constraints $\mathcal{S}$. Defining $s_I^*(\boldsymbol{w})$ as an optimal solution (5) with $\mathcal{S} = I$ given fixed weights $\boldsymbol{w}$, we may define a vector of hyper-objectives $\boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w}) = \mathbb{E}_{\mathcal{I}}[\boldsymbol{f}(s_I^*(\boldsymbol{w}))]$. Similarly, we define $u^{\mathrm{E}}(\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w})$. It is easily verified using an identical strategy to the proofs presented in Section III that $u^{\mathrm{E}}(\boldsymbol{w})$ possesses all of the properties (continuity, concavity, $\partial u(\boldsymbol{w}) = \boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w})$ for all $\boldsymbol{w} \in \mathcal{W}$) that make Algorithm 1 possible. Further, the function $u^{\mathrm{E}}(\boldsymbol{w})$ is precisely the expected optimal cost $\mathbb{E}_{\mathcal{I}}[u_I(\boldsymbol{w})]$ for $u_I(\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(s_I^*(\boldsymbol{w}))$. Indeed for any weight $\boldsymbol{w} \in \mathcal{W}$

$$\mathbb{E}_{\mathcal{I}}[u_I(\boldsymbol{w})] = \boldsymbol{w} \cdot \mathbb{E}_{\mathcal{I}}[\boldsymbol{f}(s_I^*(\boldsymbol{w}))] = \boldsymbol{w} \cdot \boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w}) = u^{\mathrm{E}}(\boldsymbol{w}).$$

This implies that if $r^{\mathrm{E}}(\boldsymbol{w}'|\boldsymbol{w}^*) = \boldsymbol{w}^* \cdot \boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w}') - u^{\mathrm{E}}(\boldsymbol{w}^*)$ for $\boldsymbol{w}', \boldsymbol{w}^* \in \mathcal{W}$ then $r^{\mathrm{E}}(\boldsymbol{w}'|\boldsymbol{w}^*)$ is precisely the expected regret $\mathbb{E}_{\mathcal{I}}[\boldsymbol{w}^* \cdot \boldsymbol{f}(s_I^*(\boldsymbol{w}')) - u_I(\boldsymbol{w}^*)]$. Thus, by replacing $\boldsymbol{f}(s^*(\boldsymbol{w}))$ with $\boldsymbol{f}^{\mathrm{E}}(\boldsymbol{w})$ in Algorithm 1, we can obtain a set Ω with bounded maximum *expected* regret over all constraints $\mathcal{I}$.

VII. Numerical Results

We demonstrate our algorithm in simulations for two different applications: trajectory planning and multi-robot coordination. We consider three variations of our problem: i) optimal solvers are available (Section IV), ii) only a suboptimal solver is available (Section V), and iii) a prior belief over relevant weights is given (Section VI). These experiments illustrate how the proposed methods allow for computing pre-sampled sets of weights and their solutions that closely approximate solutions to any weight requested online. In particular, we report the worst-case approximation error. In an additional experiment, we investigate how the proposed method can be used for presampling solution in order to learn user preferences.

A. Simulation Setup

a) Planning problems: The first planning problem involves computing Dubins trajectories for different objectives, similar to the example shown in Figure 1. In the two-objective case we consider the competing objectives of trajectory length and integral square jerk (a common metric of comfort [2], [17]). For higher dimensions we additionally consider the maximum jerk, as well as avoiding high-risk areas of the environment. Given a weight vector, we compute the Dubins' trajectory that optimizes the resulting cost function numerically. In detail, because Dubins' trajectories are easily computed for a fixed minimum turning radius, we iterate over small increments in turning radius within given bounds selecting the one with minimum cost. Though the result is sub-optimal, it can be made arbitrarily close to optimal by increasing the resolution; our experiments used $10,000$ steps.

Thus, we consider this approach to satisfy Assumption 1. The second planning problem is the multi-vehicle traveling salesman problem (mTSP) with deadlines [48]. We consider two objectives: the number of vertices visited before their deadline and the total distance travelled by all robots. Since the problem is NP-hard, we use it to highlight the extension of our work to suboptimal solvers in Section V.

b) Evaluation measures: For evaluation we rely on a large (1000) set of uniformly randomly sampled weights. We evaluate algorithm performance using the regret as defined in (6), as well as the relative regret where the difference in (6) is replaced with a ratio. For the experiments with prior distributions over weights, we consider the essential supremum of the discounted regret and the expected regret from (18b) and (18a), respectively.

c) Baseline algorithms: We compare our proposed algorithm against three baselines: i) Uniform sampling in the weight space [15], [49], [50], denoted by `Uniform`, ii) sampling weights from a prior distribution for the stochastic problem settings, denoted by `Prior` and iii) the adaptive weight sampling [19], [40], denoted as `AWS`. We refer to our proposed approach from Algorithm 1 as `MRPS`, and its modification to suboptimal solvers as `MRPS-S` and considering probabilistic weights as `MRPS-P`.

d) Sampling budgets: We test the different algorithms with different budgets K for the number of weight samples. Since our algorithm initially computes the n single-objective solutions, we only consider $K \geq n$.

B. Experiments with Optimal Solvers

We begin with planning Dubins trajectories as shown in Figure 1. To find solutions $s^*(\boldsymbol{w})$, the motion planner can sample a large set of Dubins trajectories using different turn radia and pick the optimum among these.

a) Illustrative Example: First, we present more insight into the example from Figure 1 with $K = n + 5 = 7$ samples. We notice that `MRPS` produces a larger variety of sample trajectories, especially those with shorter length. This is also visualized in the approximations of the Pareto front: `Uniform` exhibits a large gap, while the proposed method places samples more homogeneously. In Figure 5 we show the optimal cost $u(\boldsymbol{w})$ (ground truth computed with 10,000 uniform weights), together with the tangent planes of the approximating samples of both baselines `Uniform` and `AWS` and our proposed approach. Here, `Uniform` places weights – and thus tangents – in an equal distance along the x-axis from one another. This results in numerous samples with similar tangents on the left side of the plot where the function $u(\boldsymbol{w})$ is almost linear. At the right end where $u(\boldsymbol{w})$ changes more rapidly, uniform does not have sufficient samples for a tight approximation. In contrast, `MRPS` places more samples at the right end resulting in a smaller gap between the best approximating tangent and the $u(\boldsymbol{w})$, such that the maximum regret is $\approx {}^1\!/_{10}$ compared to `Uniform`. While `AWS` improves over `Uniform`, the approximation is not as tight as `MRPS`, resulting in a maximum regret that is still twice as large.

Solver	Budget K				
	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
`Uniform`	0.59/1.08	0.25/0.44	0.14/0.2	0.05/0.07	0.02/0.03
`AWS`	0.37/0.91	0.10/0.14	0.10/0.14	0.06/0.14	0.05/0.14
`MRPS`	0.26/0.45	0.06/0.11	0.03/0.07	0.01/0.02	0.00/0.01

(a) Maximum regret for $n = 2$ objectives.

Solver	Budget K				
	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
`Uniform`	1.14/1.7	0.67/1.03	0.66/1.03	0.65/1.03	0.51/0.88
`AWS`	0.83/1.41	0.39/0.63	0.25/0.58	0.22/0.58	0.22/0.58
`MRPS`	0.76/1.61	0.15/0.25	0.13/0.22	0.07/0.11	0.03/0.05

(b) Maximum regret for $n = 3$ objectives.

Solver	Budget K				
	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
`Uniform`	0.81/1.33	0.8/1.31	0.71/1.31	0.48/1.04	0.34/0.85
`AWS`	0.78/1.31	0.65/1.31	0.62/1.31	0.46/1.17	0.44/1.17
`MRPS`	0.58/1.09	0.2/0.34	0.15/0.28	0.1/0.2	0.05/0.12

(c) Maximum regret for $n = 4$ objectives.

TABLE I: Mean and 95^{th} percentile of the maximum regret for the Dubins planning problem with an optimal solver.

b) Quantitative Analysis: We repeat the above Dubins planning experiment with randomized goal locations and various sampling budgets K. Results are shown in Figure 6. When $K = n$ only the basis solutions $e^1, \dots, e^n$, *i.e.*, the single objective solutions, are available. We observe that `MRPS` achieves substantially smaller absolute and relative regret values for all $K > n$ compared to `Uniform`. With just 3 samples, `MRPS` achieves a performance comparable to `Uniform` using 10 samples, and for 20 samples `MRPS` provides approximations with effectively no regret. The `AWS` approach performs similar to `MRPS` for few samples ($K = n + 1$ and $K = n + 3$), yet makes only very little progress for larger K. In summary, the experiment shows that the proposed `MRPS` method is able to efficiently place samples that allow for minimum regret approximations for any scalarization weight.

c) Varying number of objectives: We also considered problem variances with 3 or 4 objectives with detailed results shown in Table I. As expected, with increasing number of objectives the regret increases (with exception of `MRPS` for $K = n+1$). Nonetheless, `MRPS` achieves the best performance under all settings. In particular, `MRPS` continues to decrease the regret for larger K almost converging to an optimal set of samples with zero regret. In contrast, `Uniform` and `AWS` make only little progress for $K > n+5$ in the three objective setup.

In summary, this experiment showed that, given an optimal solver, Algorithm 1 solves Problem 1 for varying number of objectives, strongly outperforming baseline methods.

C. Experiments with Suboptimal Solvers

Next we conduct experiments when no optimal solver is available, as discussed in Section V. The planning problem is a multi-vehicle Traveling Salesman Problem (mTSP) with deadlines. Suboptimal solutions are computed using a Large Neighbourhood Search (LNS) heuristic [51] running for with

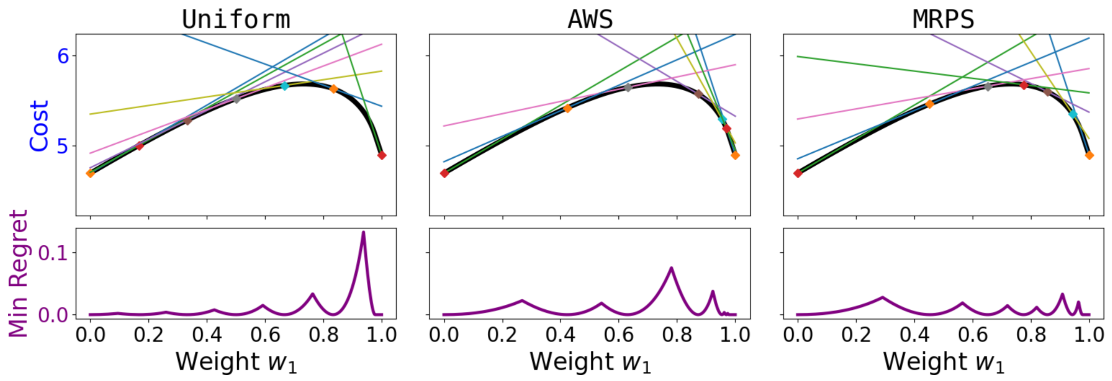


Fig. 5: Example results for the Dubins planning problem with two objectives. Shown are the approximations of $u(\boldsymbol{w})$ and resulting regret with `Uniform`, adaptive sampling (`AWS`), and the proposed approach `MRPS`.

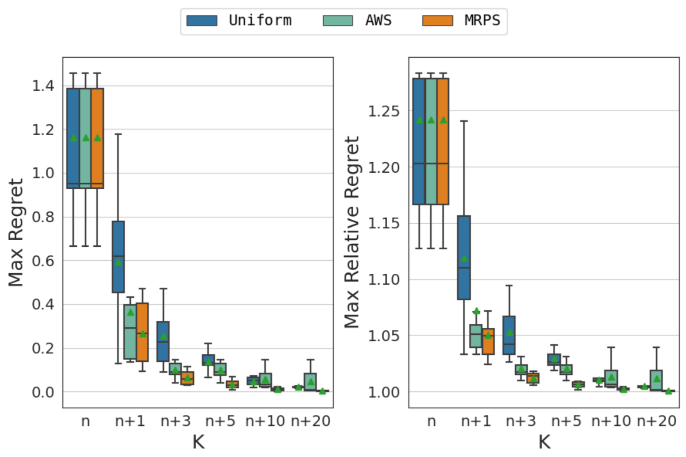
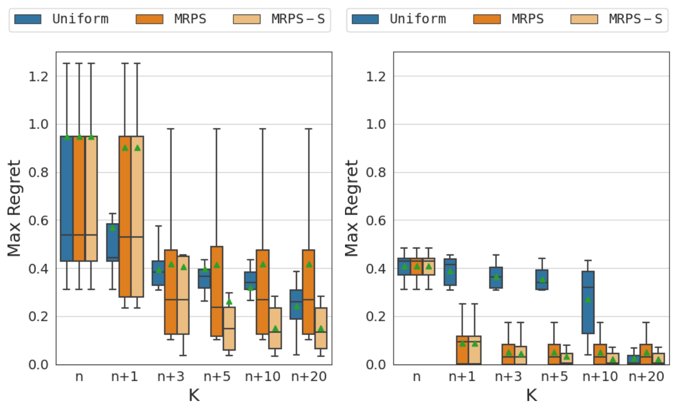


Fig. 6: Results for the Dubins experiment with $n = 2$ objectives.



(a) Cheap heuristic (10 iterations). (b) Strong heuristic (1000 iterations).

Fig. 7: Results for the mTSP experiment with $n = 2$ objectives, 20 vertices and 20 robots with different suboptimal LNS heuristics.

varying computation budget, *i.e.,* number of iterations. We randomly generate 10 different problem instances with 20 vertices and 20 robots, all starting at a central depot. Due to the poor scalability of exact solvers for the problem, we do not have access to the ground truth $u(\boldsymbol{w})$. Instead, we compute the regret with respect to solutions found by the LNS heuristic with $10,000$ iterations.

Figure 7 shows the results, comparing `Uniform` with `MRPS` and the modification `MRPS-S` proposed in Section V. Given the suboptimal solver for mTSP, the assumption made in `MRPS` are not satisfied. The algorithm can still be executed, yet it may prematurely determine that it is unable to make further progress and thus terminate. This experiment thus highlights the benefit of `MRPS-S` when no exact solver is available. The `AWS` approach cannot be directly incorporated into the LNS solver since it poses an equality constraint on the solution vector. Such a constraint raises issues of feasibility particularly in discrete optimization problems like mTSP. Thus, `AWS` is omitted from this experiment. Further, we compare algorithm performance when different solvers for the mTSP are available: We consider a *strong heuristic*, where we run LNS with $1,000$ iterations, and a *cheap heuristic*, using LNS with only 10 iterations.

First, we notice that the different version of the heuristic directly influence the solutions: The regret is substantially smaller for the stronger heuristic using 1000 iterations. Yet, under both settings we observe that `MRPS` does not make any more progress after three iterations. Since Algorithm 1 depends on having access to an optimal solver, it terminates prematurely. For both variants of the heuristic, `Uniform` eventually outperforms `MRPS`. However, we observe that `MRPS-S` avoids the pitfalls of `MRPS` and strongly outperforms both, `MRPS` and `Uniform`. We notice that due to the suboptimal solver, none of the methods converge to a regret of zero. While `Uniform` eventually ties with `MRPS-S`, our methods remains much more efficient: For the strong heuristic, `Uniform` requires $n + 20$ samples to achieve the same regret as `MRPS-S` already has with $n + 5$ samples. Lastly, the performance gap of `MRPS-S` (and `MRPS`) compared to `Uniform` is larger for the stronger heuristic. Since our method relies on the solution vectors of previous solutions, it is misguided when the solver returns poor solutions. Thus, for unreliable solvers, `Uniform` can be more robust.

Overall, the experiment has shown that i) the original approach `MRPS` is challenged when a suboptimal solver is used, eventually performing poorer than uniform sampling. This is particularly apparent when the solver is a cheap heuristic. ii) The extension `MRPS-S` addresses this shortcoming and is able to find strong solutions within few iterations, strongly outperforming greedy for cheap and strong heuristics.

D. Experiments with Stochastic Problem Inputs

We now consider the setup from Section VI-A where a prior belief over weights is available. The objective is to minimize the essential supremum of the discounted regret formulated in (18b). We consider priors $g(\boldsymbol{w})$ in the form of a normal distribution, with uniformly random mean from the weight space $\mathcal{W}$, and uniformly random variance within $[.01, .1]^n$.

a) Illustrative Example: Figure 8 showcases the difference between the different approaches when planning Dubins trajectories with two objectives. We observe that for all approaches, the maximum regret and the maximum discounted regret occurs at different weights. While MRPS achieves the lowest regret, its discounted regret is largest since it neglect to sample around weights with a high prior belief. In contrast, the modified approach MRPS-P places fewer samples at the right end, trading-off a higher regret for a smaller discounted regret. In comparison, directly sampling from the prior belief (labelled Prior) does not yield the desired result: Here samples are too concentrated around the distribution mean and does incur a higher maximum discounted regret.

b) Quantitative Results: Figure 9 shows detailed results for varying numbers of samples K. for Dubins trajectories with three objectives. We include the maximum discounted regret and expected regret corresponding to the problems formulated in (18a) and (18b), respectively. Table II additionally contains the discounted regret for 2 and 4 objectives.

In Figure 9, we observe that MRPS-P does not perform well with only $K = n + 1$ sample. Indeed, in the first iteration it is equivalent to MRPS and does not consider the prior belief since there is only one neighbourhood to explore. Yet, for $K = n + 3$ MRPS-P shows already the lowest essential supremum of the discounted regret, and approaches 0 for $K = n + 10$. The original method MRPS achieves a similarly strong performance for only $K = n + 20$. Thus, when the sampling budget is limited, MRPS-P performs better. Among the baselines Uniform performs best. As highlighted in the example in Figure 8, only sampling from the prior belief (Prior) does not explore different weights efficiently and thus only makes little progress after the first few iterations. In the right plot, we illustrate the expected regret. Overall, we observe a similar trend as for the maximum discounted regret, yet Uniform, MRPS also approach 0 for $K = n + 20$. Yet, MRPS-P shows the strongest benefit for $K = n + 3$ and $K = n + 5$, highlighting that while MRPS-P is designed to solve (18a), it also bounds the error for (18b) and thus is suitable for tackling both problems. For completeness, we include results for AWS. Since this approach does not take the prior into account, it is unable to effectively progress after a few iterations and thus not suitable for this problem without further adaptation.

Overall, the third experiment showed that MRPS-P effectively adapts the original algorithm to the setting where a prior belief over weights is given. For the two objectives – the maximum discounted regret and the expected regret – MRPS-P proved to be more sample efficient than the baseline methods and the original MRPS algorithm. In summary, throughout the three different experiments we demonstrated that our proposed algorithm and its two extensions provide are able to approximate the set of Pareto-optimal solutions for different problem variants.

	Budget K				
Solver	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
Uni.	0.36/1.25	0.20/0.74	0.10/0.33	0.04/0.12	0.01/0.05
AWS	0.86/3.64	0.13/0.30	0.09/0.25	0.05/0.22	0.03/0.22
MRPS	0.32/0.87	0.12/0.31	0.05/0.16	0.01/0.03	0.00/0.01
Prior	0.35/1.31	0.24/0.85	0.24/0.85	0.17/0.53	0.17/0.53
MRPS-P	0.32/0.87	0.06/0.17	0.03/0.08	0.01/0.02	0.00/0.01

(a) Maximum discounted regret for $n = 2$ Objectives.

	Budget K				
Solver	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
Uni.	0.55/1.70	0.19/0.47	0.14/0.39	0.11/0.30	0.07/0.20
AWS	0.49/1.67	0.39/1.40	0.39/1.40	0.39/1.40	0.39/1.40
MRPS	1.13/3.29	0.31/0.85	0.18/0.57	0.10/0.24	0.04/0.12
Prior	0.26/0.66	0.25/0.66	0.23/0.55	0.23/0.55	0.21/0.55
MRPS-P	1.13/3.29	0.20/0.58	0.10/0.24	0.05/0.17	0.03/0.08

(b) Maximum discounted regret for $n = 3$ Objectives.

	Budget K				
Solver	$n+1$	$n+3$	$n+5$	$n+10$	$n+20$
Uni.	3.71/7.04	2.39/7.04	2.09/4.38	0.80/2.02	0.49/1.07
AWS	4.83/13.96	2.68/13.96	2.38/13.96	0.94/2.90	0.93/2.90
MRPS	4.85/13.96	1.56/4.93	0.84/3.21	0.56/1.73	0.34/1.52
Prior	2.26/7.00	1.90/7.00	1.71/6.00	1.65/6.00	1.49/5.89
MRPS-P	4.85/13.96	1.42/1.92	0.74/1.52	0.39/1.13	0.21/0.58

(c) Maximum discounted regret for $n = 4$ Objectives.

TABLE II: Mean and 95^{th} percentile of the maximum discounted regret for the Dubins planning problem with an optimal solver and prior distributions over weights.

E. Computation times

We briefly report the practical computation time for a Python implementation run on a I7-10750H 2.6GHz with 32GB ram. Computing $K = n + 10$ samples with MRPS or MRPS-P for the Dubins problem runs within $\approx .4s$ for two objectives, and within $\approx .5s$ for four objectives, on average. For the mTSP problem the computation time of MRPS-S for $K = n + 10$ samples is $\approx 1.1s$ for the cheap heuristic, and $\approx 17s$ for the strong heuristic.

F. Reward Learning

To illustrate the practical impact of the proposed method, we consider the problem of learning user preferences, i.e., learning a user specific, but hidden weight vector $\boldsymbol{w}^*$. Our algorithm serves a pre-processing step to generate a ground set of potential robot behaviours, from which we then elicit the solution that best fits a user's preferences. As mode of user interaction, we consider learning from choice [7], [12], [15], [16], [50] where the user is iteratively queried with two potential robot trajectories and indicates the preferred one. Repeating this over multiple iterations allows the robot infer about the user weights $\boldsymbol{w}^*$. Most algorithms for this problem require a set of presampled trajectories from which the best query is selected using some heuristic [12], [15], [16], [50].

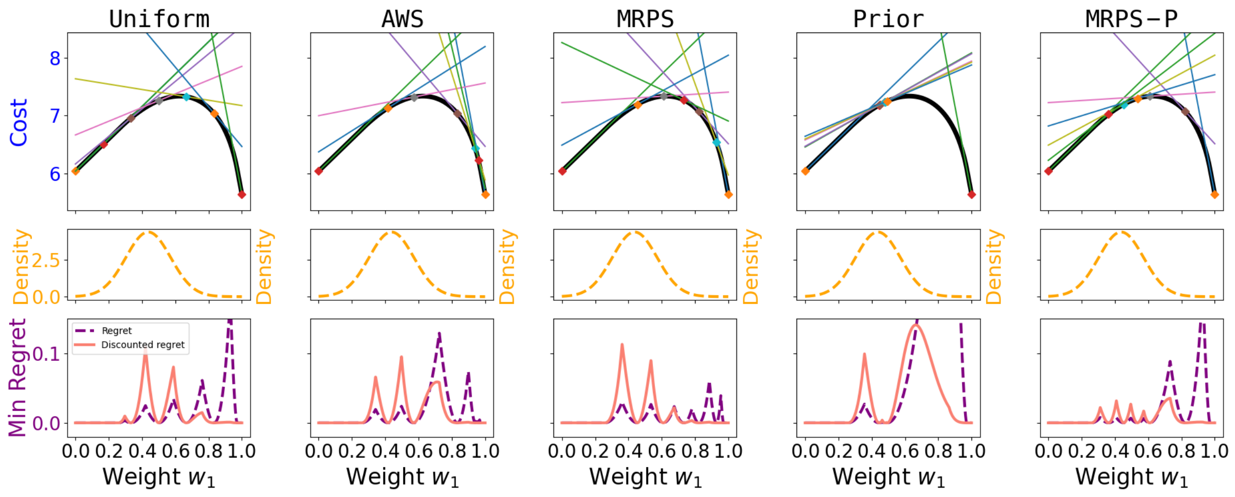


Fig. 8: Pareto sampling with prior distributions over weights. We compare the original algorithm MRPS with the adaptation MRPS-P proposed in Section VI-A. *Top row:* Objective function $u(\boldsymbol{w})$ and the sample points placed by the respective algorithms. *Middle row:* Prior belief. *Bottom row:* Regret and discounted regret incurred by the solutions.

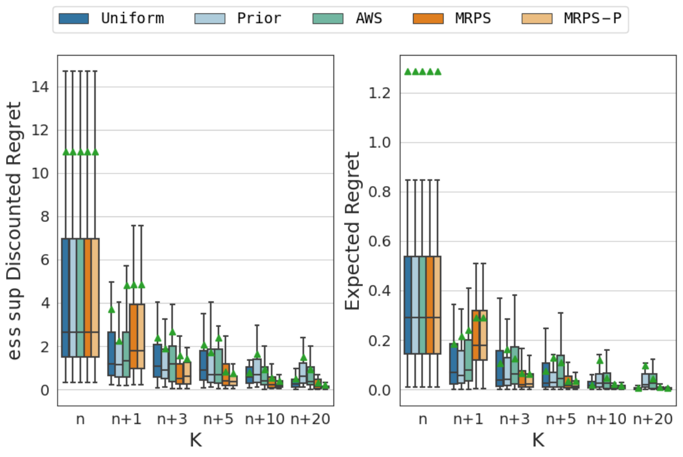


Fig. 9: Results for the Dubins experiment with $n = 3$ objectives and prior distributions over weights.

These presamples are either random trajectories [12], [16], or optimal solutions for uniformly random weights for the LSMOP [15], [35], [50].

Our proposed algorithm MRPS can be used to generate presamples for these learning problems. Thus, we compare the learning progress over 10 iterations when either using Uniform or MRPS samples. For a clear comparison, we use a simple, deterministic user model: presented with trajectories A and B, they choose A if and only if $f(A)\boldsymbol{w}^* \leq f(B)\boldsymbol{w}^*$. Similar to [7], [46] the trajectory preferred in the previous iteration constitutes one of the two trajectories presented in the next. The robot can *actively* choose the second trajectory to be presented in the next iteration. We employ a random selection from the presampled set (Random), or the minmax regret approach from [15] (Regret). Finally, generating random user weights $\boldsymbol{w}^*$ is not trivial: When $\boldsymbol{w}^*$ is drawn uniformly random, the sample set Uniform comes from the same distribution, biasing the experiment. Thus, we randomly select users from the union of two sample sets Ω(Uniform) and Ω(MRPS) each of size $K = 20$. We evaluate learning performance using the relative error $f(S^*(\boldsymbol{w}'))\cdot\boldsymbol{w}^*/u(\boldsymbol{w}^*)$ where $\boldsymbol{w}'$ is the expected user weight. This measure is widely used in reward learning problems [36] and captures how well the learned cost function approximates the unknown user cost function (parameterized by $\boldsymbol{w}^*$) in terms of the quality of the corresponding solutions.

We test this learning framework using the four feature Dubins planning problem from earlier, with one fixed goal location. Figure 10 shows the result for learning with presampled sets of various sizes K. We observe that the MRPS samples lead to a smaller learning error than Uniform samples, regardless of the query method (Random or Regret). Indeed, the Uniform sets do not always include a close-to-optimal sample such that the learning is eventually unable to make progress. That is the case when $\boldsymbol{w}^*$ was drawn from the MRPS samples. However, the opposite effect is negligible: Learning with the MRPS samples finds close-to-optimal solutions, implying that the error is very small even when $\boldsymbol{w}^*$ comes from Uniform.

When comparing different sizes K of the sample sets, we observe that all approaches learn slightly slower for larger K - the increased number of available trajectories seems to rather distract the learning algorithm than offering more informative queries. More surprisingly, when using Uniform samples, the learning still stops making progress. This indicates that the larger set still does not contain close-to-optimal solutions. This further supports our earlier findings that while MRPS only needs small K to find a close-to-optimal sample for any $\boldsymbol{w}^*$, while Uniform is unable to achieve the same even for large K.

In summary, the experiment shows that using MRPS to generate pre-sampled solutions in reward learning allows for learning close-to-optimal solutions with significantly fewer samples than when relying on randomly generated pre-samples.

VIII. Summary

In this paper, we studied the problem of computing different trade-offs between competing objectives in robot planning

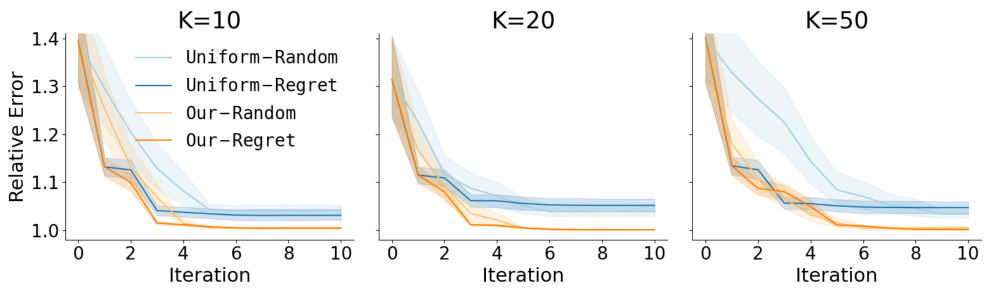


Fig. 10: Learning from choice with presampled sets of different size K.

problems. Commonly, such problems are approached via linear scalarization where multiple objectives are combined into a single objective taking the form of a weighted sum. Thus, we focused on computing a finite set of weights and corresponding solutions such that any other solution can be approximated with minimal regret. We studied fundamental properties of this linear objective function and its relation to regret. Assuming that we have access to an optimal solver for the scalarized objective, we presented an iterative sampling algorithm that repeatedly adds weights where the regret of the current set is largest and return a tight error bound. Next, we extended the algorithm to accept suboptimal solvers, and retained the error bound when approximation factors are provided. In a second extension, we also considered prior beliefs over practically relevant scalarization weights and adapted our algorithm to minimize the discounted maximum regret. In a series of simulations, we showcased the proposed methods for the three different problem variants, demonstrating their higher sampling efficiency compared to baselines. In a further experiment, we highlighted the practical effect of well-designed sample sets when learning user preferences for robot behavior.

IX. Discussion and Future Work

We extended our earlier results [2] eliminating the restrictive assumption that an optimal solver is available and also extended the framework to stochastic settings. This highly increases the practical value of the proposed method. However, in this paper we focused on establishing theoretical results and limited the evaluation to a series of simulation experiments. Future work should investigate the practical benefits of our method in two different settings: Many planners require careful parameter tuning. For any objective using a weighted sum, the presented method can be used to efficiently explore different parameter settings, especially when the the tuning process is sensitive. The presented technique may also be adapted to tuning loss functions for machine learning systems. The other application is learning user preferences for robot behaviour. We have shown that our method yields samples that allow for learning user preferences more accurately and more efficiently. The practical benefits should be further investigate considering different modes of user interaction and need practical verification in user studies. Finally, our analysis is limited to linear scalarization. If the Pareto front of the underlying MOP is non-convex, linear scalarization fails to capture all Pareto-optimal solutions, *i.e.,* is not Pareto-complete. However, many of the theoretical results presented here can be extended to any scalarization technique that is concave in the weight space. Thus, future work should consider how our proposed algorithm may be adapted to techniques such as Chebyshev-scalarization to ensure Pareto-completeness.

References

[1] 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 *2015 IEEE/RSJ IROS*. IEEE, 2015, pp. 250–256.

[2] A. Botros and S. L. Smith, “Tunable trajectory planner using g 3 curves,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 273–285, 2022.

[3] F. Christianos, P. Karkus, B. Ivanovic, S. V. Albrecht, and M. Pavone, “Planning with occluded traffic agents using bi-level variational occlusion models,” *arXiv preprint arXiv:2210.14584*, 2022.

[4] Z. Zuo, X. Yang, Z. Li, Y. Wang, Q. Han, L. Wang, and X. Luo, “Mpc-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 3, pp. 513–522, 2020.

[5] 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 transactions on vehicular technology*, vol. 69, no. 12, pp. 14 458–14 469, 2020.

[6] A. Bajcsy, D. P. Losey, M. K. O’Malley, and A. D. Dragan, “Learning robot objectives from physical human interaction,” in *CoRL*. PMLR, 2017, pp. 217–226.

[7] N. Wilde, A. Blidaru, S. L. Smith, and D. Kulić, “Improving user specifications for robot behavior through active preference learning: Framework and evaluation,” *IJRR*, vol. 39, no. 6, pp. 651–667, 2020.

[8] M. Čáp and J. A. Mora, “Multi-objective analysis of ridesharing in automated mobility-on-demand,” in *RSS*, 2018.

[9] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, “Diffstack: A differentiable and modular control stack for autonomous vehicles,” *arXiv preprint arXiv:2212.06437*, 2022.

[10] R. T. Marler and J. S. Arora, “The weighted sum method for multi-objective optimization: new insights,” *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[11] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Springer Science & Business Media, 2012, vol. 164.

[12] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, “Active preference-based learning of reward functions,” in *RSS*, Jul. 2017.

[13] J. Y. Zhang and A. D. Dragan, “Learning from extrapolated corrections,” in *IEEE ICRA*, 2019, pp. 7034–7040.

[14] R. Holladay, S. Javdani, A. Dragan, and S. Srinivasa, “Active comparison based learning incorporating user uncertainty and noise,” in *RSS Workshop on Model Learning for Human-Robot Communication*, 2016.

[15] N. Wilde, D. Kulić, and S. L. Smith, “Active preference learning using maximum regret,” in *2020 IEEE/RSJ IROS*, 2020, pp. 10 952–10 959.

[16] 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 *CoRL*, 2019.

[17] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 163–168.

[18] A. Botros, A. Sadeghi, N. Wilde, J. Alonso-Mora, and S. L. Smith, “Error-bounded approximation of pareto fronts in robot planning problems,” in *15th Workshop on the Algorithmic Foundations of Robotics, WAFR 2022*. Springer, 2023, pp. 506–522.

[19] I. Y. Kim and O. L. de Weck, “Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation,” *Structural and multidisciplinary optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[20] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, “Diffstack: A differentiable and modular control stack for autonomous vehicles,” in *Conference on Robot Learning*. PMLR, 2023, pp. 2170–2180.

[21] 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 *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7064–7070.

[22] Y. Che, A. M. Okamura, and D. Sadigh, “Efficient and trustworthy social navigation via explicit and implicit robot–human communication,” *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 692–707, 2020.

[23] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, “Model predictive contouring control for collision avoidance in unstructured dynamic environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[24] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “Chomp: Covariant hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[25] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, “Motion planning around obstacles with convex optimization,” *arXiv preprint arXiv:2205.04422*, 2022.

[26] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, “Online trajectory generation with distributed model predictive control for multi-robot motion planning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[27] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments,” *IJRR*, vol. 29, no. 5, pp. 485–501, 2010.

[28] Y. Zeng, X. Xu, S. Jin, and R. Zhang, “Simultaneous navigation and radio mapping for cellular-connected uav with deep reinforcement learning,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 7, pp. 4205–4220, 2021.

[29] D. Kent and S. Chernova, “Human-centric active perception for autonomous observation,” in *2020 IEEE ICRA*, 2020, pp. 1785–1791.

[30] B. Sakcak and S. M. LaValle, “Complete path planning that simultaneously optimizes length and clearance,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 100–10 106.

[31] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, “Human preferences for robot-human hand-over configurations,” in *IEEE/RSJ IROS*, 2011, pp. 1986–1993.

[32] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, “Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences,” *The International Journal of Robotics Research*, vol. 41, no. 1, pp. 45–67, 2022.

[33] S. Habibian, A. Jonnavittula, and D. P. Losey, “Here’s what i’ve learned: Asking questions that reveal reward learning,” *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 4, pp. 1–28, 2022.

[34] C. Basu, M. Singhal, and A. D. Dragan, “Learning from richer human guidance: Augmenting comparison-based learning with feature queries,” in *Proceedings of the ACM/IEEE HRI*, 2018, pp. 132–140.

[35] N. Wilde, E. Biyik, D. Sadigh, and S. L. Smith, “Learning reward functions from scale feedback,” in *CoRL*. PMLR, 2022, pp. 353–362.

[36] N. Wilde and J. Alonso-Mora, “Do we use the right measure? challenges in evaluating reward learning algorithms,” in *Conference on Robot Learning*. PMLR, 2023, pp. 1553–1562.

[37] A. Bobu, D. R. Scobee, J. F. Fisac, S. S. Sastry, and A. D. Dragan, “Less is more: Rethinking probabilistic models of human behavior,” in *Proceedings of the ACM/IEEE HRI*, 2020, pp. 429–437.

[38] J. Branke, J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective optimization: Interactive and evolutionary approaches*. Springer Science & Business Media, 2008, vol. 5252.

[39] I. Das and J. E. Dennis Jr, “A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems,” Tech. Rep., 1996.

[40] I. Y. Kim and O. L. De Weck, “Adaptive weighted-sum method for bi-objective optimization: Pareto front generation,” *Structural and multidisciplinary optimization*, vol. 29, pp. 149–158, 2005.

[41] J. Lee, D. Yi, and S. S. Srinivasa, “Sampling of pareto-optimal trajectories using progressive objective evaluation in multi-objective motion planning,” in *IEEE/RSJ IROS*, 2018, pp. 1–9.

[42] V. Pereyra, M. Saunders, and J. Castillo, “Equispaced pareto front construction for constrained bi-objective optimization,” *Mathematical and Computer Modelling*, vol. 57, no. 9-10, pp. 2122–2131, 2013.

[43] J. Xu, A. Spielberg, A. Zhao, D. Rus, and W. Matusik, “Multi-objective graph heuristic search for terrestrial robot design,” in *IEEE ICRA*, 2021, pp. 9863–9869.

[44] S. Parisi, M. Pirotta, and J. Peters, “Manifold-based multi-objective policy search with sample reuse,” *Neurocomputing*, vol. 263, pp. 3–14, 2017.

[45] O. Schütze, O. Cuate, A. Martín, S. Peitz, and M. Dellnitz, “Pareto explorer: a global/local exploration tool for many-objective optimization problems,” *Engineering Optimization*, vol. 52, no. 5, pp. 832–855, 2020.

[46] N. Wilde, A. Sadeghi, and S. L. Smith, “Learning submodular objectives for team environmental monitoring,” *IEEE RA-L*, vol. 7, no. 2, pp. 960–967, 2021.

[47] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[48] H. N. Psaraftis, M. Wen, and C. A. Kontovas, “Dynamic vehicle routing problems: Three decades and counting,” *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[49] Q. Zhang and H. Li, “Moea/d: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[50] N. Wilde, D. Kulić, and S. L. Smith, “Bayesian active learning for collaborative task specification using equivalence regions,” *IEEE RA-L*, vol. 4, no. 2, pp. 1691–1698, Apr. 2019.

[51] A. Sadeghi and S. L. Smith, “Heterogeneous task allocation and sequencing via decentralized large neighborhood search,” *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.

Appendix

A. Implementation of Max-Min Neighbourhood Regret

We detail the implementation of the linear program in Equation (12). We formulate the convex hull constraint $\boldsymbol{w} \in \mathcal{C}(N)$ using a scalars $\lambda^1, \dots, \lambda^n \in [0,1]$ to write $\boldsymbol{w}$ as a convex combination of the neighboughood weights $\boldsymbol{w} = \lambda^1 \boldsymbol{w}^1 + \dots + \lambda^n \boldsymbol{w}^n$. The equality constraints are given by

$$\begin{bmatrix} 1 & & \dots & 1 & 0 & \dots & 0 \\ 0 & & \dots & 0 & 1 & \dots & 1 \\ -1 & 0 & \dots & 0 & w_1^1 & \dots & w_1^n \\ 0 & -1 & \dots & 0 & w_2^1 & \dots & w_2^n \\ \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & \dots & -1 & w_n^1 & \dots & w_n^n \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ \lambda^1 \\ \vdots \\ \lambda^n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (21)$$

The first row ensures that $\boldsymbol{w}$ lies in $\mathcal{W}$ (*i.e.,* all its components sum to 1). The second ensures the same for $\lambda^1, \dots, \lambda^n$. The other rows ensure that the i-th element of the vecotr $\boldsymbol{w}$ is a convex combination of the i-th component of all neighbourhood vectors $\boldsymbol{w}^1, \dots, \boldsymbol{w}^n$. In the objective function, we write $P(\boldsymbol{w})$ using the same convex combination as

$$x - \sum_{i=1}^{n} \lambda^i u(\boldsymbol{w}^i). \quad (22)$$

Finally, we require that $w_i \geq 0$ and $\lambda^i \geq 0$ for all $i = 1, \dots, n$.

Biography Section

Alexander Botros (Member, IEEE) is the director of Machine Learning & Artificial Intelligence at Integrus Solutions, a Hong Kong-based due diligence research company. He completed this work during a postdoctoral fellowship at the Autonomous Systems Lab at the University of Waterloo where he received his PhD in Electrical & Computer Engineering (2021). His current research includes natural language processing, language and logic modelling, and machine learning more broadly. Alex completed his undergraduate (2007) & M.Sc. (2016) engineering work at Concordia University in Montreal.

Nils Wilde (Member, IEEE) is currently a Postdoctoral Fellow in the Autonomous Multi-Robots Lab working with Javier Alonso-Mora at TU Delft. Until August 2021 he was a postdoctoral fellow at the Autonomous Systems Lab at the University of Waterloo where he also did my PhD in Electrical and Computer Engineering (ECE) under the co-supervision of Dana Kulić and Stephen L. Smith from 2016 to 2020. His research interests include robot motion planning, multi-robot coordination, human-robot interaction (HRI), and multi-objective planning, focusing on the development algorithmic frameworks on the intersection of control, learning and optimization.

Armin Sadeghi is currently a software engineer at RideCo, anon-demand transit technology and services company based in Waterloo, Canada. He completed this work during this work during his time as a Postdoctoral Fellow in the Autonomous Systems Lab. His general research interests are in the areas of control and optimization with focus on multi-robot task allocation in ride-sourcing and coverage control applications. He completed his PhD in September 2020 from the Autonomous Systems Lab. He received his undergraduate degrees in Mechanical and Aerospace Engineering from the Sharif University of Technology and his Master's degree in Electrical Engineering from the University of Waterloo.

Javier Alonso-Mora (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in robotics from ETH Zürich, Zürich, Switzerland, in 2010 and 2014, respectively.

He is currently an Associate Professor with the Delft University of Technology, Delft, The Netherlands. Until October 2016, he was a Post-Doctoral Associate with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. He was also a member of the Disney Research, Zürich. His main research interests include autonomous navigation of mobile robots, with a special emphasis in multi-robot systems and robots that interact with other robots and humans. Toward the smart cities of the future, he applies these techniques in various fields, including self-driving cars, automated factories, aerial vehicles, and intelligent transportation systems.

Dr. Alonso-Mora was a recipient of the European Research Council (ERC) Starting Grant in 2021, the IEEE International Conference on Robotics and Automation (ICRA) Best Paper Award on Multi-Robot Systems in 2019 and the Veni Grant from the Netherlands Organization for Scientific Research in 2017.

Stephen L. Smith (Senior Member, IEEE) received the B.Sc. degree in engineering physics from Queen's University, Canada, in 2003, the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Canada, in 2005, and the Ph.D. degree in mechanical engineering from the University of California, Santa Barbara in 2009.

He is currently a Professor in the Department of Electrical and Computer Engineering at the University of Waterloo, Canada, where he holds a Canada Research Chair in Autonomous Systems. He is Co-Director of the Waterloo Artificial Intelligence Institute. From 2009 to 2011 he was a Postdoctoral Associate in the Computer Science and Artificial Intelligence Laboratory at MIT. Prof. Smith has received several awards including the Early Researcher Award from the Province of Ontario, the NSERC Discovery Accelerator Supplement Award, and two Outstanding Performance Awards from the University of Waterloo.

He is a licensed Professional Engineer (PEng), an Associate Editor of the IEEE Transactions on Robotics and the IEEE Open Journal of Control Systems. He was previously Associate Editor for the IEEE Transactions on Control of Network Systems (2017 - 2022), and was a General Chair of the 2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). His main research interests lie in motion planning and optimization for autonomous systems, with a focus on learning and optimal decision making.