Statistically Distinct Plans for Multi-Objective Task Assignment

Nils Wilde, and Javier Alonso-Mora

***Abstract*—We study the problem of finding statistically distinct plans for stochastic task assignment problems such as online multi-robot pickup and delivery (MRPD) when facing multiple competing objectives. In many real-world settings robot fleets do not only need to fulfil delivery requests, but also have to consider auxiliary objectives such as energy efficiency or avoiding human-centered work spaces. We pose MRPD as a multi-objective optimization problem where the goal is to find MRPD policies that yield different trade-offs between given objectives. There are two main challenges: 1) MRPD is computationally hard, which limits the number of trade-offs that can reasonably be computed, and 2) due to the random task arrivals, one needs to consider statistical variance of the objective values in addition to the average. We present an adaptive sampling algorithm that finds a set of policies which i) are approximately optimal, ii) approximate the set of all optimal solutions, and iii) are statistically distinguishable. We prove completeness and adapt a state-of-the-art MRPD solver to the multi-objective setting for three example objectives. In a series of simulation experiments we demonstrate the advantages of the proposed method compared to baseline approaches and show its robustness in a sensitivity analysis. The approach is general and could be adapted to other multi-objective task assignment and planning problems under uncertainty.**

***Index Terms*—Multi-Robot Task Assignment, Pickup and Delivery, Path Planning for Multiple Mobile Robots, Multi-Objective Optimization.**

I. Introduction

Autonomous robots are becoming increasingly capable of solving complex tasks in challenging and dynamically changing environments. These advancements will soon enable the large scale deployment of robot fleets in a wide range of applications including transportation, on-site assistance service, autonomous mobility on demand, environmental monitoring and inspection. For instance, the deployment of mobile robots in hospitals and care homes for assistive tasks such as material transport promises to reduce the workload of perpetually overburdened skilled personnel [1], [2].

Many robot planning problems such as path planning, multi-robot task assignment (MRTA), multi-agent path finding (MAPF) or multi-robot pickup and delivery (MRPD) need to consider multiple competing objectives simultaneously. Usually, the primary goal is to provide the optimal quality of service (QoS), captured by measures such as the average or maximum wait times, delivery delays, system throughput, or the number of on-time deliveries, among others. However, in practice, the deployment of autonomous robots may require the consideration of additional objectives. For instance, service robots might need to balance between quality of service (QoS) and operation cost [3], or consider sustainable costs [4]. Moreover, when navigating in human-centered workspaces robot fleets need to consider established social norms, such as avoiding areas of the environment with high foot traffic [5] or social navigation objectives [6].

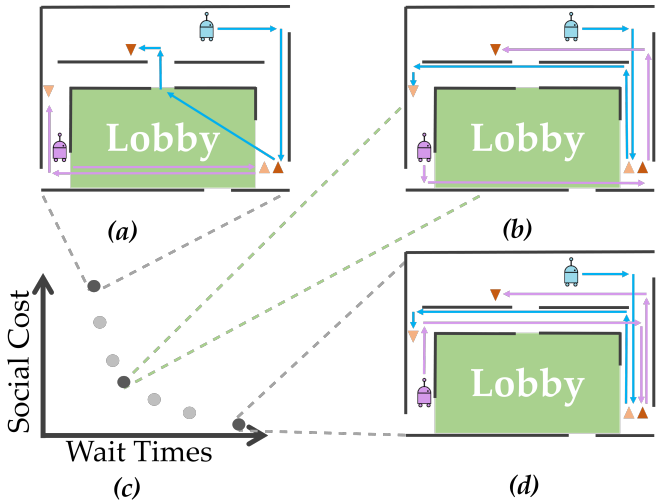


Fig. 1: Schematic example for multi-objective multi-robot pickup and delivery. Two robots are required to deliver two packages (dark and light orange) from a pickup location (triangle up) to a delivery point (triangle down). Different system plans vary in their trade-off between task wait times and a social cost for traversing a lobby with high foot traffic. The plot in (c) shows the values of the two objectives functions attained by the different system plans, called the *Pareto front* of the multi-objective optimization problem.

In this paper, we study how we can compute statistically distinct system plans when optimizing for multiple objectives under uncertainty. We specifically focus on multi-objective MRPD [7], a special case of MRTA where a fleet of robots needs to service dynamically appearing transportation requests. However, the proposed solution technique could be extended to multi-objective MAPF or other MRTA variants when considering stochastic task arrivals or travel times.

Figure 1 shows an example for MRPD with two competing objectives: minimizing wait time of deliveries and avoiding a lobby area where robot traffic is undesired. The system plan shown in (a) prioritizes QoS, while (d) avoids robot traffic in the lobby whenever possible. Arguably, both solutions are not ideal. On one hand, only optimizing QoS completely ignores

Manuscript received: June, 29, 2023; Revised October, 30, 2023; Accepted December, 23, 2023. This paper was recommended for publication by Editor Nancy M. Amato upon evaluation of the Associate Editor and Reviewers' comments.

This research is supported by the European Union's Horizon 2020 research and innovation program under Grant 101017008.

N. Wilde and J. Alonso-Mora are with the Department for Cognitive Robotics, 3ME, Delft University of Technology, Delft, Netherlands, {n.wilde, j.alonsomora}@tudelft.nl.

Digital Object Identifier (DOI): see top of this page.

the social aspects. On the other hand, when robots try to avoid human-centered areas at all cost, the wait times become very high, as illustrated in (c). When tasks have deadlines, the fleet might then not be able to deliver all packages on time. Thus, an important challenge for deploying MRPD systems is balancing between different objectives, *i.e.,* finding intermediate trade-offs. For instance, (b) shows a system where the lobby is only traversed once, allowing the purple robot to still deliver its delivery on time.

Which trade-off is most appropriate often depends on various stakeholders such as the system operator and the people present in the environment. In this paper, we study how we can find a set of MRPD plans *i.e.,* policies, that lead to different trade-offs between such competing objectives. This gives the users of an MRPD system a palette of options to choose a solution from that fits their individual preferences.

We pose MRPD as a multi-objective optimization (MOO) problem, and seek to find policies with *Pareto*-optimal trade-offs, *i.e.,* policies where neither objective can be improved without impairing another objective.

This problem bears two unique challenges: 1) due to computational hardness, it is usually not feasible to compute optimal solutions for MRPD. Thus, the computed trade-offs are not necessarily Pareto-optimal, but only heuristic solutions. Moreover, even heuristic solutions are often computationally burdensome to obtain. Therefore the number of calls to an MRPD solver should be minimized, making it impractical to generate a large ground set of MRPD policies and then selecting a representative subset based on the obtained trade-offs. 2) in *online* MRPD, the requests' arrival times as well as pickup and delivery locations are usually following a stochastic process. Thus, only exploring trade-offs for one specific sequence of requests is of limited interest, since this would represent only a certain time window (*e.g.,* day) of the deployment of an MRPD system. Arguably, it is more relevant to find trade-offs between the different objectives that lead to similar behaviour across different time windows, *i.e.,* different realizations of the stochastic process. Thus, each policy is associated with a *distribution* of cost trade-offs.

We formulate the problem of finding a set of MRPD policies such that the expected values of their cost distributions represent the Pareto-front when optimizing for the expected costs. However, the cost distributions should also remain statistically significantly different from one another such that each policy retains unique characteristics, *i.e.,* is a distinct option for operating the MRPD system. We use the popular approach of applying a linear scalarization to convert the MOO problem into a single objective function constituted by a weighted sum of the competing objectives. Each possible choice of scalarization weights then corresponds to an MPRD policy. However, picking weights that lead to a desirable set of policies is not trivial, since the relationship between weights and MRPD costs is often non-linear [8], [9]. Picking regularly spaced weights can lead to several policies having similar behaviours, *i.e.,* trade-offs, while other possible system behaviours are not covered by any policy. Therefore, we propose an adaptive sampling algorithm to find different scalarization weights. Our approach minimizes the *dispersion* of means of the distribution of cost trade-offs. Further, when adding new policies, we consider the variance of the MRPD costs to avoid choosing policies that are statistically indistinguishable.

A. Contributions

The main contributions of our work are as follows:

i) We pose the problem of stochastic multi-objective MRPD. To reflect the stochastic nature of task arrivals, the formulation considers statistical variance of the objectives in addition to their mean values.
ii) We propose an adaptive sampling algorithm to find MRPD policies that approximate the *expected* Pareto-front and ensures that MRPD plans are statistically *distinguishable*.
iii) We establish completeness of the proposed algorithm.
iv) We provide example MRPD configurations for three different objective functions building on a state-of-the-art algorithm for single-objective MRPD.
v) In simulation experiments, we showcase our approach for two different MRPD scenarios with varying numbers of tasks and robots and demonstrate its advantages compared to several baseline approaches.

B. Related Work

Many robotic planning problems face the challenge of being required to simultaneously optimize multiple objectives, for instance in path and trajectory planning [5], [8], [10], autonomous driving [11]–[14] transportation and mobility on demand [3], multi-robot planning [15]–[19].

Designing robotic systems that face multiple-objectives requires representations of Pareto-fronts, which is a well-known problem in optimization [9], [20]–[23], but also studied for specific robotics applications [3], [8], [24]–[27]. A common approach to multi-objective optimization is linear scalarization, *i.e.,* using the weighted sum of the individual objective functions to pose a single optimization problem [9]. The set of Pareto-optimal solutions is then approximated by exploring different scalarization weights [5], [8], [28]–[30]. However, finding useful weights is often challenging [8], [9], [31], [32].

a) Multi-objective multi-robot problems: Finding trade-offs between objectives is relevant in many multi-robot problems such as multi-robot task assignment (MRTA) [33]–[36], dynamic vehicle routing (DVR) [37]–[40], multi-robot pickup and delivery (MRPD) [7], [41], [42], automated mobility-on-demand (AMoD) systems [43], [44], and multi-agent path finding (MAPF), among others.

The authors of [3] proposed a method for exploring different Pareto-optimal trade-offs for customer service and operation cost for AMoD systems. Our work considers a general multi-objective MRPD formulation without being constrained to specific objective functions. Similar to [3] we also explore different system plans by finding scalarization weights. However, the work in [3] is based on uniform sampling and average values, while we propose an adaptive sampling method for stochastic multi-objective optimization problems and demonstrate its advantages over uniform sampling. In the context of MRTA, the work of [45] considers the problem of balancing

the total team cost and workload balance by simultaneously minimizing the average and maximum cost of robot tours. The authors of [46] pose an MRTA problem that considers task completion times in conjunction with energy consumption of the individual robots as well as task priorities. Multi-objective trade-offs are also considered in multi-agent path finding (MAPF) [15], [17]. In MAPF, the objective is to find *collision free* paths for a fleet of agents, *i.e.,* agents are not allowed to be at the same location at any given time. In contrast, our MRPD formulation does not consider inter-agent collisions, but rather focuses on a pickup and delivery requests arriving online. Moreover, multi-objective task specification in temporal logic for multi-robot systems were studied in [16], [18]. Finally, a recurring problem in multi-robot systems is finding the optimal fleet for a given set of tasks [47] as well as fleet sizing [3], [48]–[50]. Solutions to these problems also solve a multi-objective optimization problem as they balance between the capability of the fleet and its acquisition and operational cost.

In summary, existing work on multi-objective multi-robot systems usually focuses on specific objectives. In contrast, our method is not tailored to certain objective functions and additionally considers the statistical variance in costs due to stochastic task arrivals.

b) User preferences for competing objectives: Researchers in human-robot interaction (HRI) study the problem of *reward learning* which seeks to interactively learn a reward or cost function that best describes a user's preference for robot behaviour [51]. Usually, this reward function is modelled as a weighted sum of features [5], [19], [52]–[55], which is equivalent to the linear scalarization of a MOO problem. Thus, learning a user's reward function corresponds to finding the Pareto-optimal trade-off that best fit their preferences. Effectively computing a representative set of Pareto-optimal solutions improves how well system behaviour can be adapted to user preferences [8]. In earlier work, we studied the problem of learning user preferences for material transport with consideration of task efficiency and following social norms [5], [56]. However, these works considered a set of individual start-goal transportation tasks. In contrast, this paper focuses on online MRPD where we take a fleet of robots and multiple trips per vehicle into account with stochastic task arrivals.

c) Approximating Pareto-fronts: Given the wide-spread applications of multi-objective optimization, several fundamental techniques for computing Pareto-fronts have been studied over the years [9], [20]–[23]. However, popular approaches such as gradient descent methods, evolutionary algorithms [9] or random walks [25] assume that objective values can be easily obtained and thus make use of frequently evaluating the objectives for different parameters. Computing MRPD solutions is computationally burdensome even when using heuristic solutions, making such approaches impractical.

Closely related to our work, the work presented in [31], [32] considers MOO under uncertain parameters, and pose a Pareto-approximation problem where samples are required to be statistically significantly different, focusing on applications in chemical engineering. Similar to our work, this approach iteratively places new samples on the Pareto-front using a divide-and-conquer (DC) approach to find new weights. Both algorithms stop dividing when solutions are no longer significantly different. However, a key difference is that [31], [32] chooses new Pareto-samples by uniformly placing weights (similar to a breadth-first-search). In our work, we place new weights such that we greedily minimize dispersion, *i.e.,* the distance in the objective space. This makes our method more sample efficient and results in a more homogeneous coverage of the Pareto-front when the number of samples is limited. Moreover, we provide a theoretical analysis establishing completeness of our approach. We compare both methods in simulations.

The authors of [57] studied the MOO for robotics when objectives are expensive to evaluate. Their method replaces fitness functions used in GA with an expected improvement in hypervolume. However, this requires a surrogate objective function but no principal approach is given in [57]. In contrast, our work is based on a greedy placement of new samples to reduce the *dispersion*, a measure for the distance between points on the Pareto-front. This can be directly approximated from the objective values that have been already sampled. Our earlier work [8] studies the problem for finding a Pareto-approximation with bounded regret for general multi-objective problems formulated as weighted sums. The number of samples and thus computations of objective values is budgeted, however, a limiting assumption is that an optimal solution can be obtained for any given weight. This makes the solution from [8] unsuitable for multi-objective MRPD since computing exact solutions for MRPD is computationally prohibitively expensive for most practically relevant instances. Similar to our work, the authors of [58] propose an adaptive weighted sum (AWS) method for finding weights in order to approximate the Pareto-front. The AWS method iteratively identifies patches of the Pareto-front that require additional samples. Each patch is subsequently refined by adding constraints to the optimization problem and solving it again. Unfortunately, such constraints cannot be directly incorporated in an online MRPD formulation. Thus, our method does not rely on constraints but adds new weights guided by *dispersion*, a measure for the largest gaps in the current approximation of the Pareto-front. We iteratively add new sample solutions by selecting new weights as the midpoint of existing weights where difference in costs is largest and then optimizing for the new weights.

Lastly, in this paper we consider that the MOO problem has stochastic objective values since some inputs might be random variables, *e.g.,* the transportation requests appear following a stochastic process. The authors of [22] study uncertain objectives where costs cannot be computed exactly. For the case that attainable objective values fall into known bounds, they propose a notion of *probabilistic dominance*. Unfortunately, tight bounds are usually not available in MRPD a-priori.

In summary, the unique challenges of the multi-objective MRPD problem studied in this paper are the computational hardness and expense of obtaining individual sample solutions, which makes most state-of-the-art Pareto-approximation techniques impractical, and the uncertainty in the problem inputs, requiring samples of the Pareto-front to be selected such that resulting solutions are statistically different.

II. Problem Formulation

A. Preliminaries

Notation: Vectors are denoted with bold symbols ($\boldsymbol{w}$) and we use subscript indices to identify its elements (w_i). Uppercase letters denote sets (S), where we identify elements with a superscript index (s^i or $\boldsymbol{w}^i$).

Multi-objective optimization: Consider a multi-objective optimization problem (MOOP) [9] where the domain is some vector space $\mathcal{X}$. We want to find a solution $\boldsymbol{x} \in \mathcal{X}$ that simultaneously minimizes n different functions, *i.e.,* that solves $\min_{\boldsymbol{x}}\{f_1(\boldsymbol{x}), \dots, f_n(\boldsymbol{x})\}$. In general, the solution to a MOOP is not a unique element $\boldsymbol{x}$, but a set of *Pareto-optimal* solutions. We briefly review the definitions of *dominated solutions* and the Pareto-front.

Definition 1 (Dominated solution)**.** Given a MOOP and two solutions $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, vector $\boldsymbol{x}$ *dominates* $\boldsymbol{x}'$ when $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}')$ holds for all $i = 1, \dots, n$ and there exists a $j \in \{1, \dots, n\}$ where $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}')$. This is denoted by $\boldsymbol{x} \prec \boldsymbol{x}'$.

Definition 2 (Pareto Front)**.** Given a MOOP, the set of *Pareto-optimal* solutions is the subset of all solutions that are not *dominated* by another solution. This set is referred to as the *Pareto-front*.

B. Online multi-robot pickup and delivery

We now revisit the standard MRPD problem where tasks require robots to pickup items in some location in the workspace and then transport them to a different location.

The robot environment is encoded in a weighted graph $G = (V, E, d)$ where V and E are vertices and edges, and weights d describe the duration of traversing an edge. Let $R = \{r_1, \dots, r_m\}$ be a fleet of m robots that have to serve a set of n pickup and delivery tasks $\mathcal{T} = \{T_1, \dots, T_n\}$. Each task is a tuple $T = (s, g, t^r, t^d)$ where s and g are vertices in V, representing a pickup and drop-off location, t^r is the release time when the task is requested and $t^d > t^r$ is a deadline. Let $t^a(r)$ be the time a robot r arrives at the pickup vertex s, and let $t^f(r)$ be the time robot r arrives at the drop-off vertex g. A task is serviced successfully when s is visited before g and $t^f(r) \leq t^d$. Further, let L_r be the set of tasks loaded by robot r. All robots have a capacity κ, *i.e.,* $|L_r| \leq \kappa$ must hold at all times. Upon visiting a pickup vertex s the respective task T is added to L_r if $|L_r| < \kappa$, when visiting a drop-off g it is removed[1]. We assume that tasks appear online following a stochastic process $\mathcal{Y}$, and their pickup and drop-off locations are sampled randomly from a distribution over the vertices in the graph.

MRPD constitutes two subproblems: which robot serves which task, and what route each robot takes.

a) Routing Problem: For some robot r, let v denote its current location. To serve tasks $\mathcal{T}$, the robot needs to find a tour τ that starts at v and services all tasks in $\mathcal{T}$. Thus, the routing problem solves

$$\min_{\tau} \gamma(\mathcal{T}, \tau), \tag{1}$$

where $\gamma(\mathcal{T}, \tau)$ is some non-negative cost function. For instance, this can evaluate if the tasks were delivered on time, or the duration between request and delivery time.

b) Assignment Problem: An assignment is a set $\mathcal{A} \subseteq \{(r_i, T_j) | r_i \in R, T_j \in \mathcal{T}\}$ such that for every $T_j \in \mathcal{T}$ there exists exactly one pair in $\mathcal{A}$ containing T_j, *i.e.,* every task is assigned to exactly one robot. However, a robot can be assigned to multiple tasks; thus, each r_i can appear in multiple pairs in $\mathcal{A}$. Finally, let $\mathcal{T}_i(\mathcal{A})$ be the set of tasks assigned to robot r_i under $\mathcal{A}$. The goal is to find an assignment of tasks to robots, as well as tours for each robot, such that the cost γ is minimized for all tasks. The assignment problem is formulated as

$$\begin{aligned} &\min_{\mathcal{A}} \sum_{r_i \in R} \min_{\tau_i} \gamma(\mathcal{T}_i(\mathcal{A}), \tau_i) \\ &s.t.\ \tau_i \text{ serves all tasks } \mathcal{T}_i(\mathcal{A}), \\ &\quad \mathcal{T}_1(\mathcal{A}) \cup \dots \cup \mathcal{T}_m(\mathcal{A}) = \mathcal{T}. \end{aligned} \tag{2}$$

The nested optimization can be solved in a two-stage coupled approach: First, optimal tours for potential pairing of groups of tasks and robots are computed. Based on these tours, a group of tasks is assigned to each robots [43].

In an *offline* problem all tasks are known before robot deployment such that an assignment of tasks and routes for all robots are computed offline and then executed. However, in many practical problems not all tasks are known initially: further tasks might be requested while other tasks are already being serviced. Such *online* settings require frequently adding new tasks to the current assignment and replanning routes to optimally accommodate new requests. Further, our formulation does not consider inter-agent collision. Instead, we assume that collisions are avoided by a low-level controller and the average travel times are abstracted by the cost of edges on the graph. The task arrival can be modelled with a random process $\mathcal{Y}$, making the set of tasks $\mathcal{T}$ is a partially observed random variable. An optimal assignment $\mathcal{A}$ is found by a *policy* π that recomputes the current assignment and routes periodically as new tasks arrive. Thus, we redefine the cost over a policy π and tasks $\mathcal{T}$, denoted by $c(\pi, \mathcal{T})$.

C. Multi-objective MRPD

In practice, the performance of an MRPD system might be evaluated by a user with different objectives in mind. For instance, when operating in a human-centered workspace, users might have preferences for robot navigation based on several, potentially conflicting objectives.

Thus, we consider a multi-objective optimization (MOO) formulation for MRPD with bounded, positive cost functions $c_1(\pi, \mathcal{T}), \dots, c_n(\pi, \mathcal{T})$ replacing the objective of (2) with

$$\min_{\pi}\{c_1(\pi, \mathcal{T}), c_2(\pi, \mathcal{T}), \dots, c_n(\pi, \mathcal{T})\}. \tag{3}$$

We refer to this problem as multi-objective MRPD, abbreviated as MO-MRPD. In this paper, we are interested in exploring possible Pareto-optimal trade-offs to help system operators to efficiently deploy robot fleets. Since the task arrivals are stochastic, the costs $c_1(\pi, \mathcal{T}), \dots, c_n(\pi, \mathcal{T})$ are random variables, which we collect in a vector $\boldsymbol{c}(\pi, \mathcal{T}) =$

[1]Note: To avoid unintentional loading and unloading, we can design the graph G such that s and g are copies of an existing vertex in V.

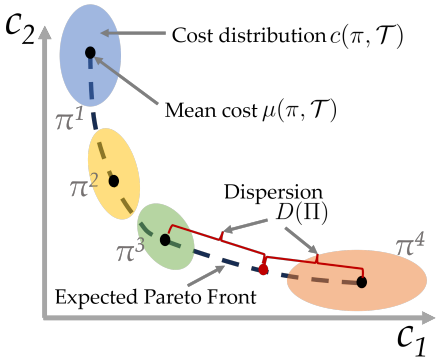


Fig. 2: Illustration of cost distributions, mean cost, expected Pareto-front, and dispersion for four different policies $\pi_1, \ldots, \pi_4$ solving a problem with two objectives.

$[c_1(\pi, \mathcal{T}) \ldots c_n(\pi, \mathcal{T})]$. Further, let $\boldsymbol{\mu}(\pi)$ be the vector containing the expected values of $\boldsymbol{c}(\pi, \mathcal{T})$, *i.e.,* $\mu_i(\pi) = \mathbb{E}_{\mathcal{T}}[c_i(\pi, \mathcal{T})]$. Using the mean costs, we introduce the notion of an *expected Pareto front*.

Definition 3 (Expected Pareto Front)**.** Given cost functions $c_1(\pi, \mathcal{T}), \ldots, c_n(\pi, \mathcal{T})$, the *expected Pareto-front* $\mathcal{P}(\mathcal{T})$ is the set of Pareto-optimal solutions to the MOOP $\min_\pi \{\mu_1(\pi, \mathcal{T}), \mu_2(\pi, \mathcal{T}), \ldots, \mu_n(\pi, \mathcal{T})\}$.

To formalize the goal of our problem, we define dispersion as a measure of distance between points on the expected Pareto-front.

Definition 4 (Dispersion)**.** Given an MO-MRPD instance with cost functions $c_1(\pi, \mathcal{T}), \ldots, c_n(\pi, \mathcal{T})$, let $\Pi = \{\pi^1, \ldots, \pi^k\}$ be a collection of policies. The dispersion of $D(\Pi)$ is the maximum distance between a point $\boldsymbol{p}$ on the expected Pareto-front $\mathcal{P}(\mathcal{T})$ and the closest mean cost vector $\boldsymbol{\mu}(\pi^i, \mathcal{T})$ for any $i = 1, \ldots, k$. In detail,

$$D(\Pi) = \max_{\boldsymbol{p} \in \mathcal{P}(\mathcal{T})} \min_{\pi \in \Pi} ||\boldsymbol{\mu}(\pi, \mathcal{T}) - \boldsymbol{p}||_2. \tag{4}$$

Lastly, we consider two different policies π^i and π^j to be *statistically distinct* if their corresponding multi-variate distributions $\boldsymbol{c}(\pi^i, \mathcal{T})$ and $\boldsymbol{c}(\pi^j, \mathcal{T})$ are statistically significantly different. This can be captured with common statistical measures such as hypothesis tests [59].

D. Problem Statement

Building on the the preliminary concepts, we formally state the problem of approximating the set of optimal solutions to an MO-MRPD problem.

Problem 1 (Approximating MO-MRPD)**.** Given a graph G, a fleet of m robots, a stochastic process $\mathcal{Y}$ to generate a sequence of tasks $\mathcal{T}$, some cost functions $c_1(\pi, \mathcal{T}), \ldots, c_n(\pi, \mathcal{T})$ and some integer $K > 0$, find a set of policies $\Pi = \{\pi^1, \ldots, \pi^k\}$ where $k \leq K$ such that

(i) For any policy $\pi \in \Pi$ and a realization of the task sequence $\mathcal{T}$ the cost vector $\boldsymbol{c}(\pi, \mathcal{T})$ is Pareto-optimal.
(ii) The *dispersion* $D(\Pi)$ is minimized.
(iii) Any two policies $\pi^i, \pi^j \in \Pi$ represent different behaviours, *i.e.,* are statistically distinct.

Solving Problem 1 provides a system operator with sampled options for fleet behaviour, that are optimal, approximate any optimal behaviour that is not part of the samples, and are distinguishable from one another.

E. Generalization

Problem 1 can be generalized to include a broader range of planning problems. Consider any robotic planning problem that i) optimizes for multiple objective functions, and ii) has one or multiple random variables as its input, *e.g.,* random task arrivals or task locations, random service times or random travel times and random rewards. Exploring the solutions space for such problems can then be formulated as in Problem 1, *i.e.,* finding solutions that are Pareto-optimal, represent the expected Pareto-front and are statistically distinct with respect to the randomness of the inputs. Thus, our solution technique presented in the following section can be adapted to problems such as Dynamic Vehicle Routing and MRTA with stochastic task arrivals, single-robot path planning or MAPF in dynamic environments, the stochastic Canadian traveller problem, or orienteering and informative path planning with stochastic travel times or rewards. For the remainder of this paper we will focus on MO-MRPD.

III. Approach

We begin with characterizing the computational hardness of Problem 1 and then present our approach to approximating MO-MRPD solutions. In essence, we cast the multi-objective problem into a scalarized single objective optimization, where the cost functions are traded-off with weights. We then solve Problem 1 by adaptively selecting scalarization weights and thus MRPD policies that result in a set of Pareto-optimal trade-offs. The proposed method does not assume specific cost functions $c_1(\pi, \mathcal{T}), \ldots, c_n(\pi, \mathcal{T})$, it only requires them to be bounded. For a case study, we show how three specific cost functions relevant in MRPD problems can be incorporated in an MRPD solver in Section IV.

A. Hardness Results

Problem 1 is related to the multi-objective shortest path (MOSP) problem which is known to be NP-hard [60]. In general, single-objective MRPD is a variation of dynamic vehicle routing and is already NP-hard for commonly used cost functions such as minimizing delivery time. Yet, MO-MRPD is also computationally intractable even for a single robot and only one task (in which case single-objective MRPD would be trivial).

Lemma 1 (Hardness result)**.** MO-MRPD with a single robot and one task is NP-hard.

Proof. This is shown by a reduction from MOSP to MO-MRPD. A MOSP instance is constituted by a graph $G = (V, E)$, start and goal vertices $s, g \in V$ and some cost functions $\gamma_1, \ldots, \gamma_n$ assigning costs to edges [60]. Its decision

version answers if there exists a path P between a given start and goal vertices $s, g \in V$ such that $\sum_{e \in E(P)} \gamma_i(e) \leq \alpha$ for all i and some constant α. We convert this into an input of an MO-MRPD instance where a single robot is initially located at s, and one task task requiring the robot to go from s to g before an arbitrarily late deadline. The MRPD cost functions are then the MOSP cost functions $\gamma_1, \dots, \gamma_n$. For a sufficiently large budget K the solution to the MO-MRPD instance is a set of policies that correspond to all Pareto-optimal paths from s to g. This trivially answers the MOSP instance. □

In summary, Lemma 1 shows that MO-MRPD is hard even if there is only a single task such that the assignment part of the problem becomes trivial and the routing part is reduced to finding a path between two vertices. Thus, the computational challenge of MO-MRPD does not only stem from complexity of the underlying single-objective MRPD, but finding a set of policies that correspond to multi-objective solutions as described in Problem 1 is itself another intractable problem.

B. Scalarization of MO-MRPD

A common approach to solve multi-objective optimization (MOO) problems such as (3) is using scalarization to obtain a single-objective function. The most common form is linear scalarization where the objectives are combined in a weighted sum:

$$\min_{\pi} \underbrace{w_1 c_1(\pi(\boldsymbol{w}), \mathcal{T}) + \dots + w_n c_n(\pi(\boldsymbol{w}), \mathcal{T})}_{\boldsymbol{w} \cdot \boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T})}. \tag{5}$$

The weight vector $\boldsymbol{w} = [w_1, \dots, w_n]$ describes a trade-off between the different MRPD objectives and thus is an input to the policy $\pi(\boldsymbol{w})$ for solving the routing and assignment problem. Hence, using a lineaer scalarization, Problem 1 becomes one of finding a set of weights $\Omega = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^K\}$. Without loss of generality, we assume that $\boldsymbol{w}$ lies in the set

$$\mathcal{W} = \{\boldsymbol{w} \in \mathbb{R}^n_{\geq 0} \mid \sum_i w_i = 1\}, \tag{6}$$

which we refer to as the weight space. Further, we assume that we have access to a policy $\pi(\boldsymbol{w})$ that solves the MRPD problem for the scalarized cost function in (5) for given weights $\boldsymbol{w}$. In the next Section IV, we will adapt a state-of-the-art MRPD solver to obtain such a policy for three exemplary cost functions.

C. Pareto approximation via weight sampling

We propose an algorithm that finds a set of scalarization weights $\Omega = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^K\}$ such that the corresponding policies $\pi^1, \dots, \pi^K$ solve Problem 1.

A commonly used approach to find different trade-offs of cost functions is sampling weights *uniformly*. However, the corresponding solutions are often not placed uniformly on the Pareto front since the mapping from weights to the objective values is non-linear [8]. This can lead to several weights yielding similar objective values. Thus, we propose an adaptive strategy that greedily minimizes the dispersion.

Algorithm 1: Adaptive sampling of MOO-MRPD

Input: A graph G, robot fleet R, a stochastic process for task arrivals $\mathcal{Y}$, MRPD cost functions $c_1, \dots, c_n$, MRPD policy $\pi(\boldsymbol{w})$, sampling budget K, # training instances η, overlap threshold Δ

Output: Sets of weights Ω and expected costs Γ.

1 $\mathcal{I} \leftarrow$ Set of η random MRPD instances drawn from $\mathcal{Y}$
2 $\Omega \leftarrow \{\boldsymbol{e}^1, \dots, \boldsymbol{e}^n\}$ // Standard basis weights
3 $\Gamma \leftarrow \{\texttt{MRPD}(\boldsymbol{w} = \boldsymbol{e}^1, \mathcal{I}), \dots, \texttt{MRPD}(\boldsymbol{w} = \boldsymbol{e}^n, \mathcal{I})\}$
4 $\mathcal{S} \leftarrow \{(\boldsymbol{e}^1, \dots, \boldsymbol{e}^n)\}$ // Initial set with basis simplex
5 **for** $k = n$ *to* K **do**
6 $\quad \boldsymbol{w}' \leftarrow \texttt{Find\_New\_Weight}(\mathcal{S}, \Gamma, \Delta)$
7 $\quad \boldsymbol{C}'_{\mathcal{I}} \leftarrow \texttt{MRPD}(\boldsymbol{w}', \mathcal{I})$ // Compute MRPD cost
8 $\quad \Omega, \Gamma, \mathcal{S} \leftarrow \texttt{Update}(\Omega, \Gamma, \mathcal{S}, \boldsymbol{w}', \boldsymbol{C}'_{\mathcal{I}}, \Delta)$
9 **return** Ω, Γ

a) Algorithm Description: Algorithm 1 provides a high-level overview of the proposed approach. After a detailed description of its components, we discuss a bi-objective example, including an illustration in Figure 4. We maintain a collection of subsets – in particular *simplexes* – of the weight set $\mathcal{W}$ from which we iteratively sample new weights and then partition the simplexes further, similar to a *bisection* algorithm in higher dimensions. The algorithm uses the sub-routine $\texttt{MRPD}(\boldsymbol{w}, \mathcal{I})$ that solves the scalarized problem from equation (5) for given weights $\boldsymbol{w}$ and η different task sequences $\mathcal{I} = \{\mathcal{T}_1, \dots, \mathcal{T}_\eta\}$. The function $\texttt{MRPD}(\boldsymbol{w}, \mathcal{I})$ returns the set of cost vectors $C_{\mathcal{I}}(\boldsymbol{w}) = \{\boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T}_1), \dots, \boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T}_\eta)\}$. That is, we compute η different cost vectors, each of dimension n to approximate the n-dimensional distribution of $\boldsymbol{c}(\boldsymbol{w}, \mathcal{T})$.

In detail, the algorithm begins by sampling η different realizations of the task arrival process $\mathcal{I} = \{\mathcal{T}_1, \dots, \mathcal{T}_\eta\}$ (line 1). Let $\boldsymbol{e}^1, \dots, \boldsymbol{e}^n$ denote the vectors of the standard basis in $\mathbb{R}^n$, which correspond to solving only the single-objective MRPD problems. First, we add these vectors to the set of sampled solutions Ω and compute their costs (line 2, 3). The tuple $(\boldsymbol{e}^1, \dots, \boldsymbol{e}^n)$ is saved in a list of n-*simplexes* (line 4). In the main loop we iteratively find a new candidate sample $\boldsymbol{w}'$ and compute the cost distribution $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}')$ (lines 6 and 7). We then update the set of n-simplexes $\mathcal{S}$ using the new sample $\boldsymbol{w}'$ (line 8), detailed in Algorithm 2. Finally, the algorithm stops when K sample solutions have been computed.

Next, we will provide details on the two core components of the Algorithm, *i.e.,* how we identify the most promising new weight $\boldsymbol{w}'$ (line 6), and how we update the simplexes (line 8).

b) Ensuring statistical difference: An important characteristic of the Algorithm is that it does not add a new weight $\boldsymbol{w}'$ when its cost distribution is too similar to the distribution of an existing sample. Thus, functions `Find_New_Weight` (line 6) and `Update` (line 8, and Algorithm 2), conduct a test for statistical significance.

To that end, let the function $h(\boldsymbol{w}^i, \boldsymbol{w}^j)$ evaluate the probability of error for a *hypothesis test* between sampled costs $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^i)$ and $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^j)$ [59]. That is, we compute how likely it is that a statistical test would wrongly conclude that

the samples for $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^i)$ correspond to the policy $\pi(\boldsymbol{w}^j)$. Let $\mathtt{KL}(\boldsymbol{w}^i||\boldsymbol{w}^j)$ be the *Kullback-Leibler* divergence between $C_{\mathcal{I}}(\boldsymbol{w}^i)$ and $C_{\mathcal{I}}(\boldsymbol{w}^j)$. The probability of error is then derived from the *Chernov-Stein Lemma* [59] as

$$h(\boldsymbol{w}^i, \boldsymbol{w}^j) = e^{-\mathtt{KL}(\boldsymbol{w}^i||\boldsymbol{w}^j)}. \tag{7}$$

c) Finding the next sample: We now specify the function `Find_New_Weight` from line 6. Let $\boldsymbol{w}^i, \boldsymbol{w}^j$ belong to the same simplex s; we refer to the unordered pair $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ as an edge of s. The idea is to greedily reduce the dispersion between samples. Without having access to the set of Pareto-optimal solutions (which we are trying to approximate), we cannot directly evaluate the dispersion as introduced in Definition 4. Thus, we use an auxiliary measure: Given a simplex $s = (\boldsymbol{w}^1, \dots, \boldsymbol{w}^n)$ let $\boldsymbol{\mu}^i$ denote the mean cost of the policy optimizing for weight $\boldsymbol{w}^i$. We define the *pair-wise dispersion* as $d^{ij} = ||\boldsymbol{\mu}^i - \boldsymbol{\mu}^j||$ for all $i, j = 1, \dots, k$ and $i \neq j$.

A strong candidate for a new weight is then the midpoint of an edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ where the pairwise dispersion is largest. However, the midpoint $\boldsymbol{w}'$ of edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ might not yield a new solution, but instead result in the same costs as either $\boldsymbol{w}^i$ or $\boldsymbol{w}^j$, *i.e.,* $\boldsymbol{\mu}' = \boldsymbol{\mu}^i$ might hold. To ensure convergence, we introduce a discount factor $\alpha((\boldsymbol{w}^i, \boldsymbol{w}^j))$. Given an edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ with mean costs $(\boldsymbol{\mu}^i, \boldsymbol{\mu}^j)$, the factor counts how often `Find_New_Weight` previously returned a weight $\boldsymbol{w}'$ that was the midpoint of some edge $(\boldsymbol{w}^l, \boldsymbol{w}^p)$ with the same mean costs, *i.e.,* where $(\boldsymbol{\mu}^i, \boldsymbol{\mu}^j) = (\boldsymbol{\mu}^l, \boldsymbol{\mu}^p)$. Further, let $H_{\Delta}(\boldsymbol{w}^i, \boldsymbol{w}^j)$ be a binary variable describing the outcome of the statistics test of distributions $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^i)$ and $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^j)$ with respect to some threshold Δ. We use the convention that 1 describes the case when $h(\boldsymbol{w}^i, \boldsymbol{w}^j) \leq \Delta$ and thus the distributions are sufficiently different. The *discounted pair-wise dispersion* is then defined as

$$D((\boldsymbol{w}^i, \boldsymbol{w}^j)) = \frac{H_{\Delta}(\boldsymbol{w}^i, \boldsymbol{w}^j)}{2^{\alpha((\boldsymbol{w}^i, \boldsymbol{w}^j))}} ||\boldsymbol{\mu}^i - \boldsymbol{\mu}^j||. \tag{8}$$

The function `Find_New_Weight` selects the edge e' among all simplexes $s \in \mathcal{S}$ that maximizes $D(e')$ and returns its midpoint $\boldsymbol{w}'$. In Section III-D we further discuss the necessity of the discount factor as part of our proof of convergence.

d) Update Function: Algorithm 2 updates the set of simplexes as well as sampled weights and solutions. Since we chose $\boldsymbol{w}'$ to be the midpoint of weights of a simplex, it lies on a simplex's boundary and thus may be inside more than one simplex. Thus, we split every simplex s containing $\boldsymbol{w}'$ into smaller simplexes.

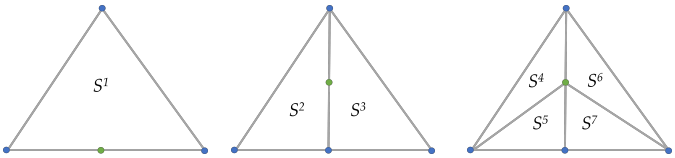
To split a simplex s, we identify the weights $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$ in s for which the new weight $\boldsymbol{w}'$ is the midpoint (line 3). We then create two new simplexes by individually substituting $\boldsymbol{w}'$ for $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$ (lines 4-6). The new sample $\boldsymbol{w}'$ is only added to the set of samples Ω (and its corresponding solution $\boldsymbol{C}'_{\mathcal{I}}$ to the set Γ) when it passes the statistics test with respect to all previously sampled solution (lines 7-9). We illustrate the update function in Figure 3 for an example with three objectives. The initial simplex is defined by the three basis weights (since all weights lie in $\mathcal{W}$, *i.e.,* their elements sum to 1, we can project the three dimensional case into 2D).

Algorithm 2: Update simplexes

Input: Set of weights Ω and solutions Γ, set of simplexes $\mathcal{S}$, new weight $\boldsymbol{w}'$, samples from cost distribution $\boldsymbol{C}(\boldsymbol{w}')$, threshold Δ

Output: Updated samples, solutions and simplexes, Ω, Γ, $\mathcal{S}$

1 **for** *s in $\mathcal{S}$ where $\boldsymbol{w}'$ lies in the convex hull of s* **do**
2 $\quad \mathcal{S} \leftarrow \mathcal{S} \setminus \{s\}$
3 $\quad$ Find weights $\boldsymbol{w}^i, \boldsymbol{w}^j \in s$ where $\boldsymbol{w}'$ is the midpoint
4 $\quad s^i = s \setminus \{\boldsymbol{w}^i\} \cup \{\boldsymbol{w}'\}$ // Swap in new weight
5 $\quad s^j = s \setminus \{\boldsymbol{w}^j\} \cup \{\boldsymbol{w}'\}$ // Swap in new weight
6 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{s^i, s^j\}$ // Add new simplexes
7 **if** *$h(\boldsymbol{w}', \boldsymbol{w}) \leq \Delta$ for all $\boldsymbol{w}$ in Ω* **then**
8 $\quad \Omega \leftarrow \Omega \cup \{\boldsymbol{w}'\}$ // Add new sample
9 $\quad \Gamma \leftarrow \Gamma \cup \{\boldsymbol{C}'_{\mathcal{I}}\}$ // Save cost distribution
10 **return** Ω, Γ, $\mathcal{S}$



(a) Initial simplex. (b) Iteration 1. (c) Iteration 2.

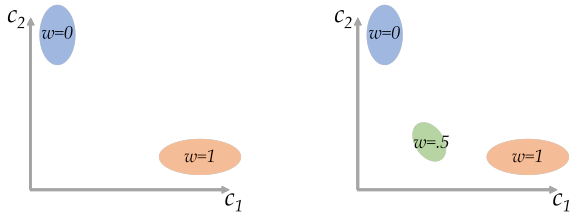
Fig. 3: Illustration of multiple splits in Algorithm 2 over two iterations, green indicates the new sample $\boldsymbol{w}'$.

In the first iteration, a new sample $\boldsymbol{w}'$ is placed on the bottom edge, and the simplex is split. In the second iteration, `Find_New_Weight`$(\mathcal{S}, \Gamma)$ may decide to place a new sample on the central edge. In that case Algorithm 2 splits both simplexes s^2 and s^3.

e) Example Illustration: To conclude the description of the approach, Figure 4 provides an illustration of the Pareto-approximation constructed by Algorithm 1 for two cost functions. Since we assumed the sum of scalarization weights to be one, we can simplify the notation and represent each weight vector $\boldsymbol{w}$ simply by its first entry w. Initially, we compute the two basis solutions $w = 0$ and $w = 1$ to get their cost distributions $\boldsymbol{C}_{\mathcal{I}}(0)$ and $\boldsymbol{C}_{\mathcal{I}}(1)$. The algorithm picks $w = .5$ as the midpoint of the only simplex $\{0, 1\}$, computes the cost distribution $\boldsymbol{C}_{\mathcal{I}}(.5)$ and subsequently adds $w = .5$ to Ω. In the second iteration, there are two simplexes $\{0, .5\}$ and $\{.5, 1\}$ where the former has a larger dispersion (as illustrated in Figure 4). Thus, the midpoint $w = .25$ is chosen. However, the resulting distribution $\boldsymbol{C}_{\mathcal{I}}(.25)$ overlaps with $\boldsymbol{C}_{\mathcal{I}}(0)$, resulting in a high probability of failing a hypothesis test. Thus, $w = .25$ is *not* added to Ω, and future calls of `Find_New_Weight` will not return the midpoint of the simplex $\{0, .25\}$. Finally, in iteration 3, the midpoint $w = .375$ of simplex $\{.25, .5\}$ is selected. The resulting distribution $\boldsymbol{C}_{\mathcal{I}}(.375)$ passes the selection criterion and thus is added to the sample set Ω.

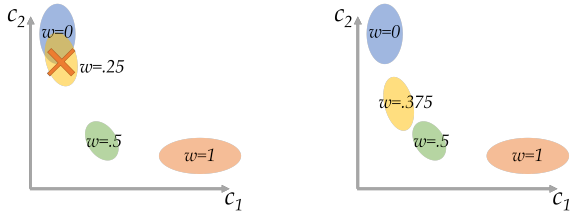
D. Theoretical Results

In this section we establish several theoretical properties of of Algorithm 1. We begin with characterizing the runtime.



(a) Basic weights $w = 0$ and $w = 1$.

(b) Iteration 1, add $w = .5$.



(c) Iteration 2, reject $w = .25$.

(d) Iteration 3, add $w = .375$.

Fig. 4: Illustration of Algorithm 1. Colored ellipses show the estimated cost distributions $\boldsymbol{C}_{\mathcal{I}}(w)$ for iteratively sampled weights.

Let $t_\pi(m, n)$ be the runtime of the MRPD-solver π for a problem with m robots and n tasks. Algorithm 1 requires exactly K calls of the MRPD solver for each of the η training instances, yielding a runtime of $O(K\eta t_\pi(m, n))$. Thus, given a polynomial time approximation or heuristic for MRPD, the runtime remains in polynomial time.

Next, we show that the proposed algorithm satisfies all three conditions formulated in Problem 1. Thus, let $\Pi = \{\pi^1, \dots, \pi^k\}$ be the policies that correspond to the weights $\Omega = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^k\}$ returned by Algorithm 1.

a) Optimal solutions: The first condition is that the policies yield *approximately* Pareto-optimal system behaviour. This strongly depends on the chosen MRPD policy. In Section IV we show the implementation of different MRPD cost functions to solve (5) using a state-of-the-art algorithm [43], which finds optimal solutions given sufficient computational budget.

b) Minimal Dispersion: Secondly, the solution Π should minimize *dispersion*. To that end, we establish completeness of Algorithm 1, *i.e.,* for a sufficiently large budget K the solution found by the algorithm has the minimal attainable dispersion. We begin with a supporting lemma.

Lemma 2 (Piece-wise constant)**.** The function $\boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T})$ is piece-wise constant over $\boldsymbol{w}$ for a fixed and finite sequence of tasks $\mathcal{T}$.

Proof. The solution space for a policy $\pi(\boldsymbol{w})$ is the set of finite sequences of vertices on the graph G for each robot, which itself is finite in size. Thus, the cost must be piece-wise constant. □

Lemma 2 implies that, given a fixed MRPD policy $\pi(\boldsymbol{w})$, there exists a finite sized set of weights $\Omega^* = \{\boldsymbol{w}^1, \boldsymbol{w}^2, \dots\}$ such that every $\boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T})$ is attained by a policy $\pi(\boldsymbol{w})$ for exactly one element in Ω^*. Let Π^* be the corresponding set of policies, which by definition achieves the smallest possible dispersion $D(\Pi^*)$. Despite Lemma 2 there can exist solutions that are only attained for a singleton $\boldsymbol{w}$, making any sampling based method only asymptotically complete. This motivates the following assumption.

Assumption 1 (Non-singleton solution weights)**.** Let Γ^* be the set of optimal solutions to (5). We assume that each solution $\boldsymbol{c}^i$ in Γ^* is optimal for some set of weights $\mathcal{W}^i \subseteq \mathcal{W}$, inscribing a ball in $\mathbb{R}^{n-1}$ of radius $r^i > 0$.

Thus, each solution can be found by any weight $\boldsymbol{w}^i$ lying in $\mathcal{W}^i$. The dimension $n - 1$ of the ball comes from the weight space $\mathcal{W}$ being a subset of $\mathbb{R}^n$, constrained by one equality. We can now establish the second key result for the proposed algorithm.

Theorem 1 (Completeness)**.** Under Assumption 1, Algorithm 1 is complete, *i.e.,* finds all solutions Γ^* given a sufficiently large but finite budget K.

Proof. To prove the theorem, we first establish two claims:

Claim 1. `Find_New_Weight` will expand any edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ where $\boldsymbol{\mu}^i \neq \boldsymbol{\mu}^j$ within finite iterations.

Claim 2. Any simplex s is not split further (*i.e.,* disregarded) only if all solutions corresponding to a weight $\boldsymbol{w}$ in s have already been sampled.

Let the input Δ of Algorithm 1 be 1 such that $H_\Delta(\boldsymbol{w}^i, \boldsymbol{w}^j) = 1$ always holds and the H-test never prevents a new weight from being added to Ω. As a shorthand let D^{ij} denote the pair-wise dispersion $D((\boldsymbol{w}^i, \boldsymbol{w}^j))$.

Subproof of Claim 1. To show the first claim, consider some simplex S with an edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$. Since $\boldsymbol{\mu}^i \neq \boldsymbol{\mu}^j$, we have $D^{ij} \geq \delta > 0$. We now prove that this edge will eventually be expanded. At some iteration k let D^{lp} be the current maximimum *discounted pair-wise dispersion*, attained for edge $(\boldsymbol{w}^l, \boldsymbol{w}^p)$ in simplex s'. Thus, `Find_New_Weight` will return the weight $\boldsymbol{w}^q$ that is the midpoint of $(\boldsymbol{w}^l, \boldsymbol{w}^p)$. This leads to two cases: 1) The solution for $\boldsymbol{w}^q$ is a new solution, 2) the solution is identical to either the solution corresponding to $\boldsymbol{w}^l$ or $\boldsymbol{w}^p$. In the first case, the algorithm explores a new element of the expected Pareto-front. Since the set of all solutions is finite, this can only happen for a finite number of iterations. However, no immediate progress is made in the second case when $\boldsymbol{\mu}^q$ is not a new solution. Without loss of generality, we order $\boldsymbol{w}^l$ and $\boldsymbol{w}^p$ such that $\boldsymbol{\mu}^q = \boldsymbol{\mu}^p$ holds. We now show that edges of the new simplexes created by Algorithm 2 have a smaller discounted pair-wise dispersion than the old edge $(\boldsymbol{w}^l, \boldsymbol{w}^p)$. The new edges are $(\boldsymbol{w}^l, \boldsymbol{w}^q)$, $(\boldsymbol{w}^q, \boldsymbol{w}^p)$, and $(\boldsymbol{w}^q, \boldsymbol{w}^r)$ for any $\boldsymbol{w}^r$ in s' where $\boldsymbol{w}^r \neq \boldsymbol{w}^l$ and $\boldsymbol{w}^r \neq \boldsymbol{w}^p$. First, $D^{qp} = 0$ trivially holds since $\boldsymbol{\mu}^q = \boldsymbol{\mu}^p$. Further, the discount factor for the new edge $(\boldsymbol{w}^l, \boldsymbol{w}^q)$ equals $\alpha(\boldsymbol{w}^l, \boldsymbol{w}^p) + 1$. Hence, $D^{lq} = {}^1\!/_2 D^{lp}$. Lastly, there are $n - 4$ other edges $(\boldsymbol{w}^q, \boldsymbol{w}^r)$. However, since $\boldsymbol{\mu}^q = \boldsymbol{\mu}^p$, the discounted pair-wise dispersion D^{qr} is equal to the existing edge $(\boldsymbol{w}^p, \boldsymbol{w}^r)$. Finally, any future iteration

returning the midpoint of either $(\boldsymbol{w}^p, \boldsymbol{w}^r)$ or $(\boldsymbol{w}^q, \boldsymbol{w}^r)$ will increment the discount factor of both edges, and thus update their discounted pair-wise dispersion.

Thus, each iteration removes a current maximizer of the discounted pair-wise dispersion, and introduces a) two new edges with $D^{qp} = 0$ and $D^{lq} = {}^1\!/_2 D^{lp}$, b) $n-4$ new edges that are *coupled* to existing edges, *i.e.,* share the same discount factor α. Hence, letting N be the total number of edges among all simplexes, it takes at most $(N-1)\log_2 {}^{D^{lp}}\!/_{\delta}$ iterations until the edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ becomes the maximizer and its midpoint is returned by `Find_New_Weight`. ■

Subproof of Claim 2. The second claim ensures that no solution is missed by not expanding a simplex further. Following the first claim, a simplex s is not expanded only if $D^{ij} = 0$ for all edges $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ of s. Since we set $\Delta = 1$ it must always hold that $H_\Delta(\boldsymbol{w}^i, \boldsymbol{w}^j) = 1$. Hence, we have $D^{ij} = 0$ if and only if $\boldsymbol{\mu}^i = \boldsymbol{\mu}^j$, following the definition in equation (8). If this holds for all edges in s, all vertices of the simplex must have equal solutions $\boldsymbol{C}_{\mathcal{I}}$. Since the set of weights $\mathcal{W}^i$ yielding the same solution is convex [56], all weights in the interior of s must also correspond to the same solution $\boldsymbol{C}_{\mathcal{I}}$. Thus, expanding s further cannot yield any solution that has not been sampled yet. ■

In conclusion, Claim 1 ensures that every edge with non-equal solutions is expanded within a finite number of iterations and Claim 2 guarantees we only disregard parts of the weight space when they cannot lead to finding a new solution. Lastly, we need to establish that it is sufficient to only add new samples on the midpoint of edges. We notice that by adding new weights as the midpoint of some edge in simplex s, the newly created simplexes s^i and s^j have edges that pass through the circumcenter of s. By Assumption 1, there will eventually be an edge $(\boldsymbol{w}^i, \boldsymbol{w}^j)$ passing through each set $\mathcal{W}^i$ corresponding to a solution. Thus, by searching over midpoints of edges, a sample will be placed in each set $\mathcal{W}^i$ after finite iterations, concluding the proof. □

To summarize, Theorem 1 ensures that Algorithm 1 achieves minimum dispersion for a sufficiently large but finite K.

Remark (Necessity of discount factor)**.** An intuitive simplification of the proposed method could be to only consider the edge with maximum pair-wise dispersion without a discount factor. However, such an approach is not complete, as we will illustrate in a simple counterexample. Consider an instance of Problem 1 with only one task and a single robot located at the task's pickup location. For three given cost functions let there be only four unique solutions on how the robot can move from the pickup location to the goal, attaining the following mean cost vectors: $\boldsymbol{\mu}^1 = [0\ 5\ 2]$, $\boldsymbol{\mu}^2 = [5\ 0\ 2]$, $\boldsymbol{\mu}^3 = [10\ 10\ 1]$, and $\boldsymbol{\mu}^4 = [6\ 6\ 1.1]$ (taking the mean here is trivial since the task arrival is fixed to be deterministic). We notice that all four vectors are Pareto-optimal since they are not dominated by any other vector. Algorithm 1 begins with the basis weights $\boldsymbol{w}^1 = [1\ 0\ 0]$, $\boldsymbol{w}^2 = [0\ 1\ 0]$, $\boldsymbol{w}^3 = [0\ 0\ 1]$ and computes the corresponding solutions, which in this case will be $\boldsymbol{\mu}^1$,$\boldsymbol{\mu}^2$ and $\boldsymbol{\mu}^3$, respectively. Among these three solutions, the maximum pairwise dispersion is found between $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$. However, observe that the optimal solution for any convex combination $w' = [\lambda\ 1-\lambda\ 0]$ of $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ is either $\boldsymbol{\mu}^1$ or $\boldsymbol{\mu}^2$. Hence, placing a sample on the edge between these two weights does not discover a new solution. Thus, without the discount factor, the algorithm will perpetually place new weights on the line between $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ without ever finding the fourth solution $\boldsymbol{\mu}^4$ and therefore does not converge.

c) Statistically different solutions: Lastly, we consider the third property that cost distributions of different policies are statistically distinguishable.

Lemma 3 (Distinguishable solutions)**.** The policies $\Pi = \{\pi^1, \ldots, \pi^k\}$ have statistically significantly different cost distributions $\boldsymbol{c}(\pi, \mathcal{T})$.

Proof. Algorithm 2, ensure that only weights $\boldsymbol{w}'$ are added to the solution set when the sampled costs $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}')$ passes an H-test against all already sampled solutions $\boldsymbol{w}$ with threshold Δ. For a sufficiently large η, the sampled cost vectors $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w})$ approximate the distributions $\boldsymbol{c}(\pi(\boldsymbol{w}), \mathcal{T})$ for all $\boldsymbol{w}$. Thus, the choice of Δ provides an upper bound on how likely it is that two solutions are not statistically significantly different. □

In conclusion, we have shown that Algorithm 1 satisfies all three requirements posed in Problem 1. Next, we will show example configurations for different MRPD objectives.

IV. Exemplary MO-MRPD configuration

In this section, we show an approach to solving the scalarized MO-MRPD, *i.e.,* the weighted cost function in (5) for three objectives.

A. Cost functions

We consider three MRPD objectives: i) the quality of service (QoS), capturing how timely deliveries are, ii) a social cost for traversing human-centered spaces similar to [5], and iii) the total travel distance, representing the overall robot traffic and energy consumption.

The quality of service $c^Q(\pi, \mathcal{T})$ measures the time between each task $T \in \mathcal{T}$ being announced and being completed - also called the *system time* of all tasks. Let τ be the tour a robot takes following policy π. We defined $t^f(T, \tau(\pi))$ as the time tour τ visits the destination vertex of task T, and $t^r(T)$ as the release time of the task. Given a task's deadline $t^d(T)$, and some large constant M, the QoS of a task is then given by

$$q(T, \tau(\pi)) = \begin{cases} t^f(T, \tau(\pi)) - t^r(T) \text{ if } t^f(T, \tau(\pi)) \leq t^d(T), \\ M \text{ otherwise.} \end{cases} \tag{9}$$

The QoS for all tasks then is

$$c^Q(\pi, \mathcal{T}) = \sum_{T \in \mathcal{T}} q(T, \tau(\pi)). \tag{10}$$

The second cost is the social cost, capturing the intrusiveness of a robot tour into human-centered parts of the environment. To this end a subset E' of the edges on the graph are labelled as *'avoid robot traffic'*, which we encapsulate in a binary indicator function $\phi(e)$. The social cost $c^S(\tau)$ then counts how

many such labelled edges are visited by a tour, *i.e.,* appear in the tours edge sequence $E(\tau)$:

$$c^S(\tau) = \sum_{e \in E(\tau)} \phi(e). \tag{11}$$

Lastly, the total distance $c^T(\tau)$ is simply the length of a robot's tour τ, defined as the sum of all edge lengths:

$$c^T(\tau) = \sum_{e \in E(\tau)} d(e). \tag{12}$$

B. Solving linearly scalarized MO-MRPD for fixed weights

We now specify a policy for solving (5) given fixed weights $\boldsymbol{w}$. We begin by showing how we compute tours for a single robot and its assigned tasks, before describing the task assignment procedure.

a) Routing Problem: Given a set of tasks $\mathcal{T}$ that are assigned to a robot, we need to find a tour starting at the robot's current location v that visits all pickup and dropoff locations s_j and g_j for all $T_j \in \mathcal{T}$, and minimizes

$$w_1 c^Q(\mathcal{T}, \tau) + w_2 c^S(\tau) + w_3 c^T(\tau), \tag{13}$$

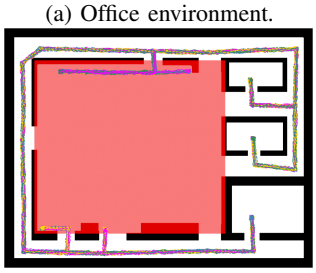
subject to ordering and capacity constraints.

The complexity of ordering and capacity constraints prevents us from using a TSP-approximation algorithm. Instead, we construct a new graph $G^{\boldsymbol{w}} = (V, E, d')$ with the same vertices and edges as the given graph G. However, edge costs are defined as $d'(e) = w_1 d(e) + w_2 \phi(e) + w_3 d(e)$ *i.e.,* capture the different costs. We notice that when only travelling from some start location to a drop-off location, optimizing QoS and total time is equivalent. However, a minimum-cost tour on $G^{\boldsymbol{w}}$ does not necessarily minimize (13). We compute tours using a two-step heuristic: First, we find an initial tour using a min-cost insertion approach, with respect to the cost in (13). Paths connecting the robot's location and the pickup and drop-off locations are then shortest paths on graph $G^{\boldsymbol{w}}$, yet the ordering of locations is picked such that (13) is minimized to a local optimum. Afterwards, we improve the tour using a large neighbourhood search (LNS) [61] with random deletion and insertion.

b) Assignment Problem: Let Q be a set of newly arrived tasks. The assignment problem decides which robot services which task in Q. In general, our framework is agnostic to the assignment algorithm. A popular state-of-the-art method is the group assignment algorithm from [43], which we employ in the simulation experiments. Unlike greedy methods, this framework actively combines different tasks. After collecting newly arrived new tasks for a fixed time period, the algorithm forms groups of tasks that could be serviced by a single robot while satisfying all deadlines are grouped together. Then, a disjoint subset of these groups is assigned to the robots using an Integer Linear Program (ILP).

c) Optimality considerations: The presented routing and assignment approach is a heuristic that finds locally optimal solutions. Yet, the approach can easily be modified to be optimal for each individual time instance by i) computing tours using exhaustive search over all possible orderings of pickup and drop-off locations, and ii) running the group assignment with groups sizes up to the number of currently outstanding tasks. Then, we would compute Pareto-optimal solutions at the current time of planning for newly arrived tasks. However, the practical benefits are limited due to the poor scalability of exhaustive search. Moreover, the heuristic implementation of the group assignment framework was shown to be effective in finding high quality solutions for large scale problems [43].

(a) Office environment.



(b) Lobby environment.

Fig. 5: Simulation environments for the experiments. Free space is shown in white, static obstacles in black. The red area indicates parts of the environment where robot traffic is undesired (*the upper floor and the main lobby, respectively*). The dashed lines show an robot routes for an example MO-MRPD solution: in (a) QoS is of high priority, such that there is a lot of robot traffic in the social space while in (b) the robots avoid the lobby as much as possible.

V. Evaluation

We evaluate our proposed method for finding a set of MO-MRPD policies in a set of simulation experiments. We consider instances of MO-MRPD with two and three objectives, namely the QoS, social cost and total length cost described in Section IV.

A. Experiment Setup

a) Environment and MRPD settings: We consider two different environments: a real-world office floorplan as well as an artificial map with a central lobby where robot traffic is undesired, both are shown in Figure 5. In both maps, the task sequences are generated using a Poisson process, while pickup and dropoff locations are sampled uniformly random from a set of predetermined locations. Task deadlines are manually chosen such that all tasks can be serviced on time when only optimizing for QoS, but deadlines might be missed when considering the other two costs. Throughout the experiment, we vary the MRPD system to feature between 2 and 8 robots with each a capacity of $\kappa = 4$ for servicing 100 to 200 tasks.

b) Algorithm settings: Algorithm 1 uses varying computation budgets K, the threshold for the h-test is set to $\Delta = .1$, and the number of instances is $\eta = 20$.

c) Baseline Comparison: We compare the proposed adaptive sampling method (Adaptive Sampling - `AS`) against several baselines. We consider two variants of finding policies by placing scalarization weights *uniformly*. The first places K weights uniformly (`Uni-A`), where K is the budget given to `AS`. Since our Algorithm has an early-stopping mechanism such that no samples are added when they fail the h-test it might use fewer than K samples. Thus, we additionally compare with uniform sampling (`Uni`) placing as many samples as were returned by `AS`.

The third baseline, proposed in [31], [32], is closely related to our work. Using a divide-and-conquer approach it divides the set of weights into regular simplexes and places samples on its vertices. We denote this algorithm as `DC`. Similar to our approach, they stop dividing a simplex further when the associated solutions are statistically too similar, evaluated by the overlap of confidence ellipsoids. We slightly modify their approach to use our H-test as the stopping criterion for a better comparison. A key difference is that their method does not use a greedy objective to select the next simplex to explore and keeps exploring until every branch of computation reaches the stopping criterion. We limit the number of computations for `DC` to the same budget K given to `AS`, and let it explore branches in a breath-first-search manner.

d) Separation of training and testing instances: The proposed algorithm as well as the `DC` baseline use several problem instances to estimate the statistical variance of the objectives. In Algorithm 1 this is denoted by the parameter η. For the main simulation results we use $\eta = 20$ random instances to run the algorithms. For evaluation we use $\eta^{\texttt{test}} = 20$ different randomly generated instances. At the end of the evaluation section, we show that our method and evaluation are robust towards changes in η and $\eta^{\texttt{test}}$.

e) Evaluation measures: We now introduce our performance measure to evaluate how well a set of sampled policies $\Omega = \{\boldsymbol{w}^1, \dots, \boldsymbol{w}^k\}$ with corresponding cost distributions $\boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^1), \dots, \boldsymbol{C}_{\mathcal{I}}(\boldsymbol{w}^k)$ solves Problem 1. First, we normalize the cost vectors. Given the means $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^k$, let $\mu_i^{\min}$ and $\mu_i^{\max}$ be the minimum and maximum mean values for cost function i. We then normalize each cost c_i^j for all $j = 1, \dots, k$ using the minimum and maximum means, *i.e.,*

$$\bar{c}_i^j = \frac{(c_i^j - \mu_i^{\min})}{(\mu_i^{\max} - \mu_i^{\min})}. \tag{14}$$

Given these normalized cost, we denote the their distributions as $\bar{\boldsymbol{C}}_{\mathcal{I}}(\boldsymbol{w}^1), \dots, \bar{\boldsymbol{C}}_{\mathcal{I}}(\boldsymbol{w}^k)$ with means $\bar{\boldsymbol{\mu}}^1, \dots, \bar{\boldsymbol{\mu}}^k$. Algorithm performance is then captured by four measures:

i) `Hypothesis_Error`. The hypothesis error characterizes how *statistically distinguishable* the behaviour produced by the different policies is. Thus, we let $h(\boldsymbol{w}^i, \boldsymbol{w}^j)$ be the probability of failing a type-1 hypothesis error between distributions $\bar{\boldsymbol{C}}_{\mathcal{I}}(\boldsymbol{w}^i)$ and $\bar{\boldsymbol{C}}_{\mathcal{I}}(\boldsymbol{w}^j)$.

ii) `Dispersion`. The dispersion captures how well the expected Pareto-front is sampled, *i.e.,* the size of gaps between sampled points on the Pareto-front. Implementing Definition 4 is impractical since the set of Pareto-optimal solutions is not available. Thus, we consider the following approximation: Given a set of policies and their mean cost vectors $\bar{\boldsymbol{\mu}}^1, \dots, \bar{\boldsymbol{\mu}}^k$, the *approximated dispersion* is the radius of the largest ball such that a) the center $\boldsymbol{p}$ of the ball is located on a line connecting two points $\bar{\boldsymbol{\mu}}^i$ and $\bar{\boldsymbol{\mu}}^j$, b) $\boldsymbol{p}$ is not *dominated* by any other point $\bar{\boldsymbol{\mu}}^q$, and c) the ball does not contain any mean cost vector $\bar{\boldsymbol{\mu}}^q$ for all $i, j, q = 1, \dots, k$.

iii) `Variance`. Variance captures how homogenous the mean cost vectors $\bar{\boldsymbol{\mu}}^1, \dots, \bar{\boldsymbol{\mu}}^k$ are placed, *i.e.,* how *uniformly* we cover the expected Pareto-front. We approximate this measure by computing a minimum spanning tree (MST) where $\bar{\boldsymbol{\mu}}^1, \dots, \bar{\boldsymbol{\mu}}^k$ correspond to vertices and edge lengths to the euclidean pair-wise distances. The variance is then the variance of the edge lengths in the MST.

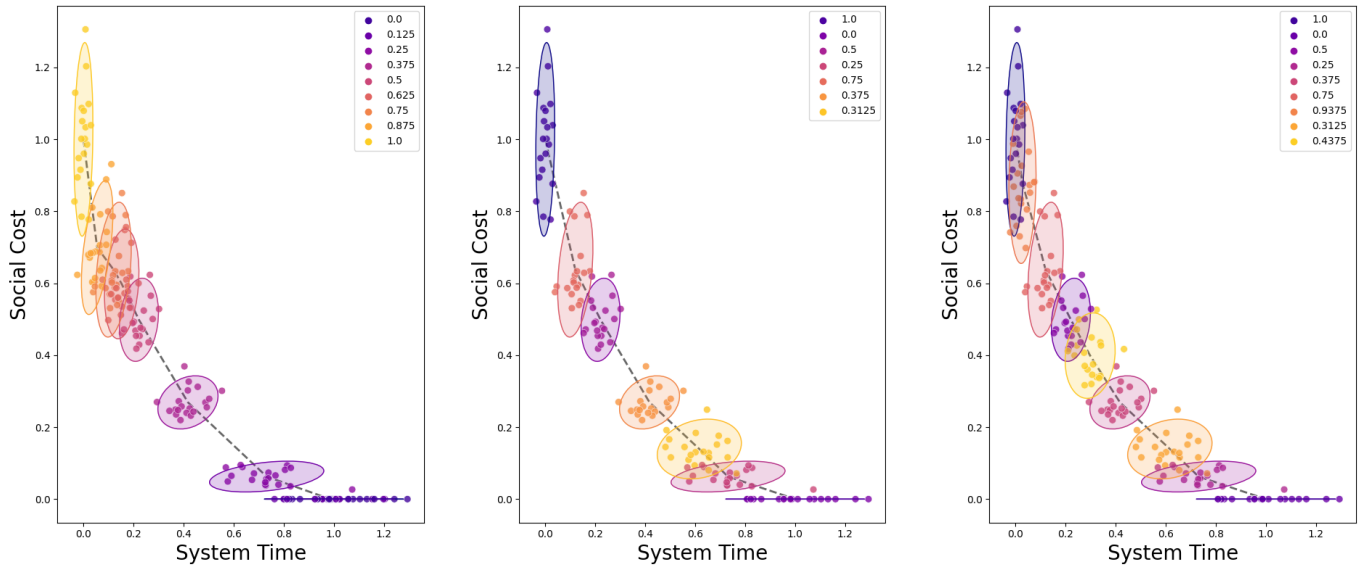
iv) `Coverage`. Coverage [20] captures how dominant the computed solutions are. Thus, we computes the volume of the subset $[0, 1]^n$ that is *not* dominated by the vectors $\bar{\boldsymbol{\mu}}^1, \dots, \bar{\boldsymbol{\mu}}^k$. We approximate the measure by sampling points in $[0, 1]^n$ and checking if they are dominated by any $\bar{\boldsymbol{\mu}}^i$ for $i = 1, \dots, k$.

For all measures, a smaller value indicates better performance. To ensure meaningful comparison between different experiment setups, we need to normalize the coverage measure. The best achievable coverage can vary greatly between problem setups since the expected Pareto-front may take different shapes, despite the normalized cost vectors. Hence, we use the variance achieved by `Uni-A` as a normalizing constant for each experiment. The other measures do not require normalization: The hypothesis error is an absolute measure. Dispersion and variance are comparable given the normalization of cost vectors in equation (14).

B. Qualitative Analysis

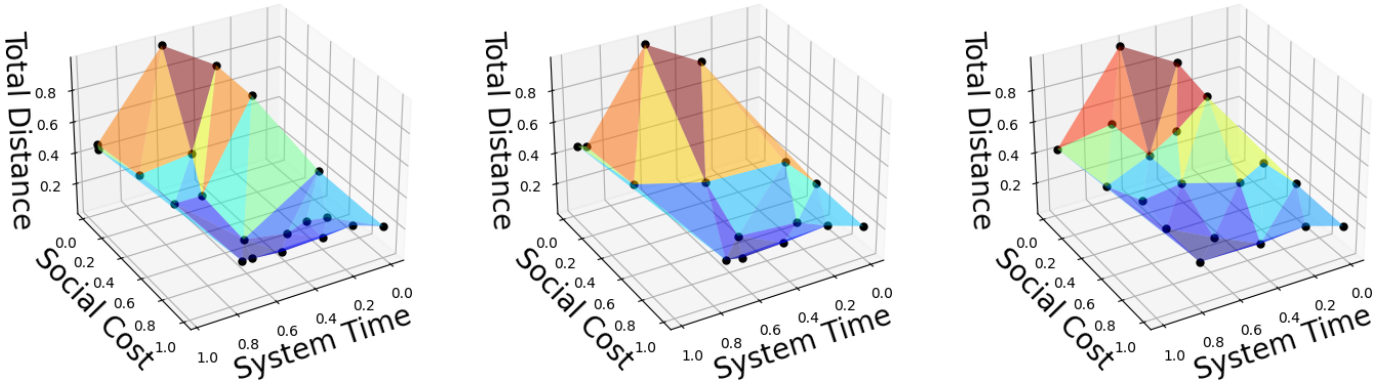
a) Two Objectives: First, we consider an example experiment in the office environment with two objectives (QoS and social cost). Figure 6 shows an exemplary comparison of the policies generated by the different approaches for a budget of $K = 10$. We plot the distributions $\bar{\boldsymbol{C}}(\boldsymbol{w})$ for each sampled weight $\boldsymbol{w} \in \Omega$, together with ellipses showing two standard deviations around the distribution means $\bar{\boldsymbol{\mu}}$. The black line shows the linear interpolation between the means.

We observe that the proposed method places samples most homogeneously covering large parts of the expected Pareto front (dispersion .25) and with only marginal overlap between the distributions (mean h-test .05). In contrast, `Uni`, exhibits a several gaps between the distributions (dispersion .37), while other distributions overlap substantially (mean h-test .28). The baseline `DC` achieves a small overlap between solutions (mean h-test .01), yet shows the largest gaps between samples (dispersion .38). Moreover, despite having the same budget as `AS`, `DC` places fewer samples (7 compared to 9), indicating a lower efficiency in finding statistically different solutions. We recall that we can only generate system plans for sampled weights $\boldsymbol{w}$. Thus, while the interpolation between samples (grey line) is similar between `DC` and `AS`, only `AS` results in more nuanced system plans.



(a) Uniform sampling (`Uni`) (b) Baseline (`DC`) (c) Proposed Adaptive Sampling (`AS`)

Fig. 6: Example of Pareto approximations with budget $k = 10$ for 4 robots, 100 tasks, operating in the lobby environment for a time horizon of $t = 3000$. Each point cloud shows a cost vector $\boldsymbol{c}(\boldsymbol{w}, \mathcal{T})$ for a policy $\pi(\boldsymbol{w})$, ellipses illustrate 2 standard deviations of the sampling distribution $\boldsymbol{C}_{\mathtt{I}}(\boldsymbol{w})$. The grey line interpolating cost means shows an approximation of the expected Pareto front. The ordering in the legends corresponds to the order in which samples were placed.



(a) Uniform sampling (`Uni`) (b) Baseline (`DC`) (c) Proposed Adaptive Sampling (`AS`)

Fig. 7: Example of Pareto approximations with budget $k = 30$ for 8 robots, 200 tasks, operating in the office environment for a time horizon of $t = 3000$. Points show the mean cost vectors $\boldsymbol{\mu}(\boldsymbol{w})$ and the 3D surface the Delaunay triangulation between vectors. Surface colors only support the 3D illustration.

The plot also shows the order samples are placed using `AS`, indicated by the color gradient. Starting with the basis solutions that yield the endpoints of the Pareto-fronts, `AS` adds new samples placed in the largest current gaps, greedily reducing dispersion.

b) Three Objectives: In a second example we show results for the lobby environment considering all three objectives (QoS, social cost and total distance), illustrated in Figure 7.

Overall, the result is similar to the 2D case: the proposed method exhibits smaller gaps between the solutions (dispersion .24) while avoiding substantial overlap (mean h-test $< .01$). In contrast, `Uni` and `DC` oversample solutions with low QoS and low total distance but high transit count, leading to high overlap (mean h-test .14 and .08, respectively), and larger gaps (dispersion .32 and .37, respectively). In contrast, `AS` produces more evenly spaced solutions over the entire expected Pareto-front, yielding a better approximation.

C. Quantitative Analysis

Next, we will provide a more in-depth analysis with several quantitative measures for various MRPD problem settings. We compare our method with all three baselines under different MRPD settings with varying fleet size and task load for both environments.

a) Results for two objectives: We conduct further experiments considering two objectives, QoS and transit count. We employ fleets of 2, 4 and 8 robots to service 100 tasks in the lobby, and 200 tasks in the office environments, yielding

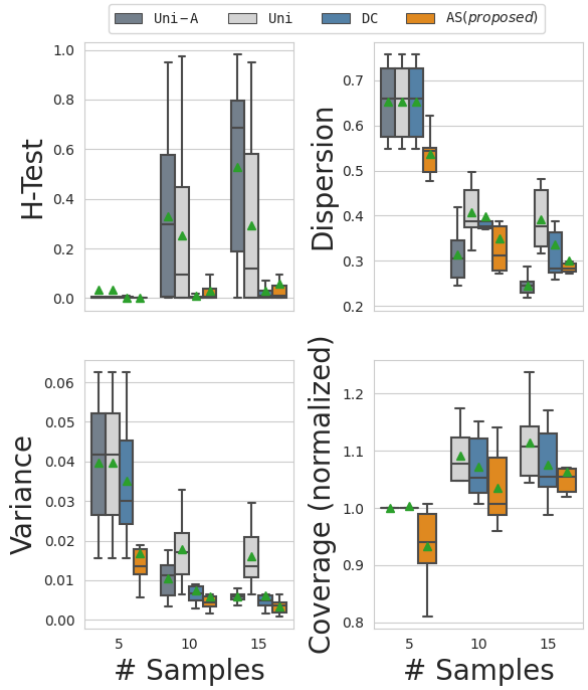


Fig. 8: Results for MRPD with two objectives. Coverage is normalized with respect to `Uni-A`.

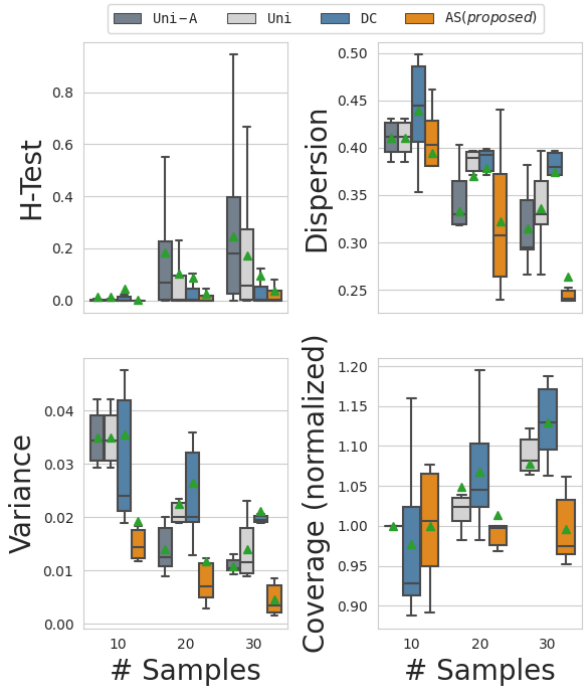


Fig. 9: Results for MRPD with three objectives. Coverage is normalized with respect to `Uni-A`.

6 different problem settings. The sampling budget K takes values 5, 10 and 15.

We illustrate the quantitative measures in Figure 8. We observe large differences between methods for the `Hypothesis_Error`: while all approaches achieve a small error for $K = 5$, values increase drastically for both uniform approaches. In contrast, `DC` and the proposed method `AS` keep the probability of failing an h-test under the threshold $\Delta = .1$.

For the `Dispersion` measure `AS` achieves the lowest values for $K = 5$ and is second best after `Uni-A` for larger K. However, we recall that `Uni-A` always uses the full budget K while `AS` often uses fewer samples. Indeed, `AS` shows clear advantages over `Uni` which uses the same number of samples. Further, `AS` also outperforms `DC` with respect to dispersion, yet with a smaller margin than compared to `Uni`.

The `Variance` shows additional insights into how the sampled solutions are spaced along the Pareto-front. Here, the proposed method shows the best performance among all approaches, *i.e.,* it places samples most evenly, yet the margin to `DC` is relatively small.

The `Coverage` omits the result for `Uni-A` since we normalize to its values. We observe that `AS` outperforms `Uni` and `DC` for all budgets; however, the difference decreases for larger K. We observe that the normalized coverage values increase with larger K for `Uni`, `DC` and `AS`. While coverage itself decreases monotonically with larger K, the normalized value, *i.e.,* the relative value compared to `Uni-A`, becomes poorer. Lastly, we notice that for $K = 5$ the coverage of `AS` lies below 1.0, indicating a better value than obtained by `Uni-A`

In summary, the proposed method outperforms `Uni-A` and `DC` on all measures. Only `DC` and `AS` are able to produce policies that are statistically significantly different. Moreover, the cost distributions of `AS` are spaced more evenly with smaller dispersion and yield a tighter approximation of the expected Pareto-front. This indicates the effectiveness of using dispersion to guide the sample placement in `AS`. In `DC`, new samples are placed without such guidance, resulting in more *unsuccessful* samples, *i.e.,* samples that end up being rejected as they are too similar to existing samples. These results highlight that `AS` is able to find better sets of MO-MRPD policies for different problem setups.

b) Results for three objectives: We rerun the experiment with the total distance as a third objective and increase the sampling budget to $K \in \{10, 20, 30\}$. The quantitative results are shown Figure 9.

The outcome of the `Hypothesis_Error` is mostly comparable to the experiment with two objectives. The uniform approaches still show an increase for larger K - albeit smaller compared to two objectives – such that their mean values still approach .2 while the upper end of the distributions exceed .8 and .6, respectively. In contrast, `AS` and the baseline `DC` show only a small increase in error for larger K, with a minor advantage for `AS`.

With respect to `Dispersion` our proposed method `AS` exhibits a strong advantage over all baselines as K increases, even outperforming `Uni-A` which uses more samples. In particular, `Uni-A` and `DC` show only a minor to no improvement when increasing the budget from $K = 20$ to $K = 30$. In contrast, `AS` is able to further reduce dispersion significantly. Similarly, on the `Variance` measure `AS` achieves the lowest value for all K, with a continued improvement as K increases. Lastly, the results for `Coverage` show a strong advantage of

AS: For all budgets K, the normalized variance has a mean of close to 1, indicating that it is similar to Uni-A. In contrast, the values for Uni and DC increase for larger K.

In conclusion, the proposed method AS shows a strong performance on all measures. Indeed, the difference to the baselines is substantially larger than for the experiments with two objectives. Thus, the proposed approach is able to place samples efficiently in higher dimensions, allowing it to produce better sets of MO-MRPD policies.

c) Runtime: We briefly report average runtimes for the MRPD computation. Solving a single instance takes $\approx 3.9s$ for two objectives and $\approx 5.0s$ for three objectives. Thus, approximating a Pareto-front with $K = 10$ samples using $\eta = 20$ training instances takes approximately 780 to $1,000s$. Yet, for larger instances such as city scale ride-pooling [43], the runtime of a single instance can be much larger, *e.g.* up to one day. The computation of the different training instances could be improved using parallelization.

d) Sensitivity to the number instances: To generate statistically different policies, our method considers multiple η randomly generated MRPD instances. To verify that the proposed algorithm performs well for different values of η, we investigate the sensitivity of the numerical results to the number of MRPD instances η. Further, we validate that the number of test instances $\eta^{\texttt{test}}$ used in the evaluation yields representative results. In the main experiments used $\eta = \eta^{\texttt{test}} = 20$, we now let η and $\eta^{\texttt{test}}$ take values in $\{10, 20, 30\}$. We highlight the results for two experiment settings and run each with two and three features. The first setting is the lobby environment with $m = 2$ robots and a sampling budget of $K = 10$ and $K = 20$ for two and three objectives, respectively. Under this setting we observed the largest deviations from the results reported in the previous sections. The second setting is the lobby environment with $m = 8$ robots and sampling budget of $K = 15$ and $K = 30$ for two and three objectives, respectively. Under this setting the H-test results of the proposed method (AS) are among the highest values in the previous experiments (upper end of the boxplots in Figures 8 and 9).

The results are summarized in Table I. First, we consider changes in the number of training instances η, *i.e.,* comparing columns in both tables. We observe a difference between $\eta = 10$ and $\eta = 20$. Using too few training instances can lead to underestimating the variance of costs. Yet, our method still outperform Uni-A and Uni (see Figures 8), and it can be expected that the results for DC would be similarly affected by $\eta = 10$. Additionally, there is no significant difference between $\eta = 20$ and $\eta = 30$. The impact of η is generally smaller in experiments with three objectives where all results remain under the tuning parameter $\Delta = .1$

Next, we consider the effect of varying the number of test instances $\eta^{\texttt{test}}$ used in evaluation. Between the rows in both tables, we observe an increase in the H-test between $\eta^{\texttt{test}} = 10$ and $\eta^{\texttt{test}} = 20$. A larger number of test instances yields a larger statistical variance, which leads to higher chance of failing the H-test. Thus, a result for $\eta^{\texttt{test}} = 10$ suggests a low H-test result, while the error can be higher when considering a larger statistic. However, there is no significant increase between $\eta^{\texttt{test}} = 20$ and $\eta^{\texttt{test}} = 30$. Therefore, we conclude that previously reported results for $\eta^{\texttt{test}} = 20$ are reliable.

	2 Objectives			3 Objectives		
$\eta^{\texttt{test}}$	$\eta = 10$	$\eta = 20$	$\eta = 30$	$\eta = 10$	$\eta = 20$	$\eta = 30$
10	.04	.00	.00	.05	.01	.02
20	.14	**.00**	.00	.09	**.02**	.04
30	.15	.00	.00	.08	.03	.04

(a) Results for the H-test in the lobby environment with $m = 2$ robots and budgets $k = 10$ (two objectives) and $k = 20$ (three objectives).

	2 Objectives			3 Objectives		
$\eta^{\texttt{test}}$	$\eta = 10$	$\eta = 20$	$\eta = 30$	$\eta = 10$	$\eta = 20$	$\eta = 30$
10	.11	.08	.08	.05	.05	.05
20	.17	**.12**	.12	.04	**.04**	.05
30	.16	.11	.11	.04	.04	.05

(b) Results for the H-test in the lobby environment with $m = 8$ robots and budgets $k = 15$ (two objectives) and $k = 30$ (three objectives).

TABLE I: Sensitivity of results for AS to the number of instances for different experiments. Shown are mean H-Test values when using different numbers of training instances η and different numbers of test instances $\eta^{\texttt{test}}$. Bold entries highlight the settings from the main experiments.

Lastly, we also consider the sensitivity towards the random sampling of tasks for a fixed η. That is, we use our main settings $\eta = 20$ and $\eta^{\texttt{test}} = 20$ and repeat experiments with 10 different random seeds. Overall, we found that the randomization of the seeds has only very little impact on the results: On all four measures reported in Figures 8 and 9, the standard deviation over different seeds is $\approx .01$, and below .001 for the Variance measure. Thus, the reported results are robust to statistical differences when sampling task sequences in Algorithm 1.

Overall, the sensitivity analysis shows that the proposed algorithm performs well under different settings for η: For a lower value of $\eta = 10$ AS still outperforms the baselines, while increasing η to 30 does not affect the results reported in the main experiments. Moreover, the evaluation results for $\eta^{\texttt{test}} = 20$ are reliable, *i.e.,* a larger number of test instances does not change the outcome. Finally, the sampling of training instances has only minimal impact on the algorithm performance.

e) Summary: In conclusion, the numerical results show that the proposed algorithm AS computes sets of MRPD policies that outperform the baselines on several metrics for different MRPD settings and sampling budgets. The H-test shows that policies found by the proposed method are statistically more distinguishable than policies found by uniform sampling. While the baseline DC is also able to produce statistically distinguishable policies, the dispersion, variance and coverage measures show that AS produces policies that better approximate the expected Pareto-front. Finally, we verified that the algorithm is robust to changes in the number of training instances η, and the quantitative results are reliable.

VI. Discussion and Future Work

We studied the problem of multi-objective MRPD where we want to find a set of policies that lead to different optimal trade-offs between given objectives. A key feature of the problem is considering the statistics of the different objective values, caused by the stochastic nature of online task arrivals. Our problem formulation does not only seek to find different MRPD policies that approximate the expected Pareto-front, but also requires the policies to attain statistically different objective values.

By means of linear scalarization we converted the problem into one of finding a set of weights that balance the cost functions. We proposed an adaptive sampling method (`AS`) and proved its completeness. Further, we presented how a state-of-the-art MRPD algorithm can be adapted to optimize for weighted cost functions for commonly used objective functions. In simulation experiments, we demonstrated that `AS` is able to produce sets of high quality MRPD plans, outperforming several baseline approaches. Thus, the proposed framework provides system operators with a variety of options for configuring the robot behaviour to their preferences.

While we specifically focused on MO-MRPD, the proposed algorithm in Section III can be applied to multi-objective formulations of a wider range of problems, such as multi-robot task assignment (MRTA) and multi-agent path finding (MAPF). Indeed, our approach does not make restrictive assumptions about the underlying MRPD solver. Further, the challenge of optimizing for competing objectives as well as stochastic problem inputs (*e.g.,* requests or goals) are prevalent in many applications.

One limitation of the proposed method is that it relies on linear scalarization of the multi-objective problem. The main shortcoming of linear scalarization is that it is not *Pareto-complete*: While every solution to the linear scalarization of the multi-objective problem is Pareto-optimal, there might exist Pareto-optimal solutions that are not a solution to (5). Thus, future work should consider other forms of scalarization to approach the MO-MRPD. One such method is using a weighted maximum instead of a weighted sum, also referred to as a Chebyshev scalarization. However, this poses major challenges for solving the MRPD problem given a choice of scalarization weights, since existing MRPD solvers are not able to optimize for such cost functions.

Future work should also consider HRI frameworks that help users to select the policy that best fits their preferences. One approach could be choice-based learning where the user iteratively chooses between two presented options [5], [19], [54], [55]. Adapting this to MO-MRPD should explore how the variance of MRPD policies affect human choices, *i.e.,* how humans can choose between different policies when their cost distributions are similar. Such a framework would complement the presented theoretical work on exploring different MO-MRPD policies and thus further help system operators to adapt MRPD systems to their specific requirements.

Finally, our work focused specifically on online MRPD. However, the problem of finding trade-offs between competing objectives under stochastic demands is relevant in other variants of multi-robot task assignment (MRTA) and dynamic vehicle routing. Thus, future work could study how the proposed framework could be applied in a broader range of planning problems: This could include considering inter-agent collisions as formalized in Multi-Agent Path Finding (MAPF), or other deployment problems such as multi-robot informative path planning and team orienteering.

References

[1] K. Niechwiadowicz and Z. Khan, "Robot based logistics system for hospitals-survey," in *IDT Workshop on interesting results in computer science and engineering*, 2008.

[2] S. Abubakar, S. K. Das, C. Robinson, M. N. Saadatzi, M. C. Logsdon, H. Mitchell, D. Chlebowy, and D. O. Popa, "Arna, a service robot for nursing assistance: System overview and user acceptability," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1408–1414.

[3] M. Cáp and J. Alonso-Mora, "Multi-objective analysis of ridesharing in automated mobility-on-demand," in *Robotics: Science and Systems (RSS)*, 2018.

[4] G. Niranjani and K. Umamaheswari, "Minimization of sustainable-cost using tabu search for single depot heterogeneous vehicle routing problem with time windows," *Wireless Personal Communications*, vol. 126, no. 2, pp. 1481–1514, 2022.

[5] N. Wilde, A. Blidaru, S. L. Smith, and D. Kulić, "Improving user specifications for robot behavior through active preference learning: Framework and evaluation," *International Journal of Robotics Research (IJRR)*, vol. 39, no. 6, pp. 651–667, 2020.

[6] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "Socnavbench: A grounded simulation testing framework for evaluating social navigation," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 3, pp. 1–24, 2022.

[7] A. Camisa, A. Testa, and G. Notarstefano, "Multi-robot pickup and delivery via distributed resource allocation," *IEEE Transactions on Robotics*, 2022.

[8] 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 International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2022.

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

[10] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective path-based d* lite," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3318–3325, 2022.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Z. Ren, S. Rathinam, and H. Choset, "Multi-objective conflict-based search for multi-agent path finding," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8786–8791.

[16] M. Cai, K. Leahy, Z. Serlin, and C.-I. Vasile, "Probabilistic coordination of heterogeneous teams from capability temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1190–1197, 2021.

[17] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective conflict-based search using safe-interval path planning," *arXiv preprint arXiv:2108.00745*, 2021.

[18] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Multi-objective search for optimal multi-robot planning with finite ltl specifications and resource constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 768–774.

[19] N. Wilde, A. Sadeghi, and S. L. Smith, "Learning submodular objectives for team environmental monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 960–967, 2022.

[20] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[21] 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.

[22] J. Teich, "Pareto-front exploration with uncertain objectives," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 314–328.

[23] 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.

[24] F. Z. Saberifar, D. A. Shell, and J. M. O'Kane, "Charting the trade-off between design complexity and plan execution under probabilistic actions," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 135–141.

[25] 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.

[26] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

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

[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] J. Xu, A. Spielberg, A. Zhao, D. Rus, and W. Matusik, "Multi-objective graph heuristic search for terrestrial robot design," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9863–9869.

[30] D. Kent and S. Chernova, "Human-centric active perception for autonomous observation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1785–1791.

[31] W. Mores, P. Nimmegeers, I. Hashem, S. S. Bhonsale, and J. F. Van Impe, "Multi-objective optimization under parametric uncertainty: A pareto ellipsoids-based algorithm," *Computers & Chemical Engineering*, vol. 169, p. 108099, 2023.

[32] S. Bhonsale, W. Mores, P. Nimmegeers, I. Hashem, and J. Van Impe, "Ellipsoid based pareto filter for multiobjective optimisation under parametric uncertainty: A beer study," *IFAC-PapersOnLine*, vol. 55, no. 20, pp. 409–414, 2022.

[33] 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.

[34] L. Luo, N. Chakraborty, and K. Sycara, "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2014.

[35] A. Ray, A. Pierson, H. Zhu, J. Alonso-Mora, and D. Rus, "Multi-robot task assignment for aerial tracking with viewpoint constraints," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1515–1522.

[36] N. Wilde and J. Alonso-Mora, "Online multi-robot task assignment with stochastic blockages," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 5259–5266.

[37] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic Vehicle Routing for Robotic Systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[38] B. H. O. Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos, "Recent dynamic vehicle routing problems: A survey," *Computers & Industrial Engineering*, vol. 160, p. 107604, 2021.

[39] 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.

[40] A. Botros, B. Gilhuly, N. Wilde, A. Sadeghi, J. Alonso Mora, and S. L. Smith, "Optimizing task waiting times in dynamic vehicle routing," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, 2023.

[41] X. Bai, A. Fielbaum, M. Kronmüller, L. Knoedler, and J. Alonso-Mora, "Group-based distributed auction algorithms for multi-robot task assignment," *IEEE Transactions on Automation Science and Engineering*, 2022.

[42] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *International Journal of Production Research*, vol. 54, no. 1, pp. 215–231, 2016.

[43] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.

[44] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 208–232, 2019.

[45] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[46] A. T. Tolmidis and L. Petrou, "Multi-objective optimization for dynamic task allocation in a multi-robot system," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1458–1468, 2013.

[47] A. Zhao, J. Xu, J. Salazar, W. Wang, P. Ma, D. Rus, and W. Matusik, "Graph grammar-based automatic design for heterogeneous fleets of underwater robots," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3143–3149.

[48] M. Chandarana, M. Lewis, K. Sycara, and S. Scherer, "Determining effective swarm sizes for multi-job type missions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4848–4853.

[49] M. Chaikovskaia, J.-P. Gayon, Z. E. Chebab, and J.-C. Fauroux, "Sizing of a fleet of cooperative robots for the transport of homogeneous loads," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1654–1659.

[50] A. Rjeb, J.-P. Gayon, and S. Norre, "Sizing of a heterogeneous fleet of robots in a logistics warehouse," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 95–100.

[51] H. J. Jeon, S. Milli, and A. D. Dragan, "Reward-rational (implicit) choice: A unifying formalism for reward learning," in *Advances in Neural Information Processing Systems (NIPS)*, Dec. 2020.

[52] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[53] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[54] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proceedings of Robotics: Science and Systems (RSS)*, Jul. 2017.

[55] 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 *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

[56] N. Wilde, D. Kulić, and S. L. Smith, "Bayesian active learning for collaborative task specification using equivalence regions," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1691–1698, 2019.

[57] M. Tesch, J. Schneider, and H. Choset, "Expensive multiobjective optimization for robotics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 973–980.

[58] I. Y. Kim and O. 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.

[59] M. Thomas and A. T. Joy, *Elements of information theory*. Wiley-Interscience, 2006.

[60] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2005, vol. 491.

[61] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

VII. Biography Section

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.

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.