Fast Near-Optimal Heterogeneous Task Allocation via Flow Decomposition

Kiril Solovey[1], Saptarshi Bandyopadhyay[2], Federico Rossi[2], Michael T. Wolf[2], and Marco Pavone[1]

***Abstract*—Multi-robot systems are uniquely well-suited to perform complex tasks such as patrolling and tracking, information gathering, and pick-up and delivery problems, offering significantly higher performance than single-robot systems. A fundamental building block in most multi-robot systems is dynamic task allocation: assigning robots to tasks (e.g., patrolling an area, or servicing a transportation request) as they appear based on the robots' states to maximize reward. In many practical situations, the allocation must account for potentially heterogeneous capabilities (e.g., availability of appropriate sensors or actuators) to ensure the feasibility of execution, and exploit predictive information concerning the likelihood of future tasks to promote a higher reward over a long time horizon. To this end, we present an efficient algorithm for predictive heterogeneous task-allocation achieving an approximation factor of at least 1/2 of the optimal reward. Our approach demonstrates that the problem can be decomposed into several homogeneous subproblems that can be solved efficiently using min-cost flow. Through simulation experiments, we show that our algorithm is faster by several orders of magnitude than a MILP-based approach.**

I. Introduction

Multi-robot systems (i.e., groups of coordinated robots cooperating towards a common goal) are uniquely well-suited to perform tasks such as patrolling, information gathering, and pick-up and delivery problems, offering significantly higher performance compared to single-robot systems. A central problem in such multi-robot applications is *predictive task allocation*: that is, to assign robots to outstanding, spatially-distributed tasks (e.g. patrolling an area or servicing a transportation request) based on the robots' states (e.g., position and power-level) and potentially heterogeneous capabilities (e.g., availability of appropriate sensors or actuators), and accounting not only for current tasks but also for the likelihood that future tasks will appear.

In this paper we consider the following setting. We have several heterogeneous fleets of mobile robots, where every fleet consists of at least one robot (robots within the same fleet have homogeneous capabilities). The robots need to be assigned to several tasks, whose location is stochastic and time varying. (A task for example can be "follow an intruder", "track an object", or "image a scientific phenomenon".) Every fleet is associated with a unique *private* task that it can execute and be rewarded for. In addition, any robot (regardless of the specific fleet they are associated with) can be assigned to and rewarded for executing a *shared* task. In our setting, a robot is rewarded for a task if (i) it resides in the spatial vicinity of the task, (ii) the task is either shared or private to robot's specific fleet, and (iii) no other robot is assigned to this task.

A number of problems of interest fall in this setting. We are particularly motivated by data-gathering and agile science applications for planetary science where multiple spacecraft detect events of interest and then perform follow-up scientific observations. A number of concepts have been proposed in this setting, including multi-spacecraft constellations to study the Martian atmosphere and the dust cycle [1], network of balloons to detect seismic and volcanic events on Venus [2], and swarms of small spacecraft to study small bodies [3]. In all of these applications, the heterogeneous task allocation problem is central: a set of vehicles with heterogeneous sensing capabilities is tasked with observing a variety of scientific events of interest (e.g. dust storms, volcanism, or changes in a body's surface); a dynamic model that approximately predicts the spatio-temporal distribution and evolution of the phenomenon of interest is available; and, while certain tasks (e.g., radio science or medium-resolution imaging) can be performed by all agents, other specialized tasks (e.g. hyperspectral imaging, deployment of sondes, or sampling) can only be performed by a subset of the vehicles.

A. Related work

Task allocation is an enabling subroutine for applications like closely coupled coordination (e.g., deciding which robot takes which place in a formation) and for loosely coupled coordination (e.g., which robot traverses to a new spot to observe an interesting target). Unfortunately, many variants of the problem are known to be computationally prohibitive to solve [4], [5]. As a result, many approaches for task allocation tend to scale poorly with the size of the problem (e.g., number of robots) or provide no guarantees on the solution quality or runtime. Furthermore, fleets of robots with heterogeneous capabilities (e.g., ground vehicles and aerial drones jointly working to achieve a mutual goal), impose even more challenges for designing practical high-quality solution approaches [6], [7], [8], [9], [10].

A number of algorithm families have been proposed specifically for task allocation by the robotics community (see survey in [11]). In *auction-based algorithms* [12], [13], robots bid on tasks based on their state and capabilities. Auction-based algorithms can be readily implemented in a distributed fashion and naturally accommodate heterogeneous robots. Spatial partitioning algorithms [14] rely on partitioning the workspace where the robots operate in regions and assigning each region to one or multiple robots. Tasks within a region are assigned to the robot (or robots) responsible for the region. Spatial partitioning algorithms capture the likelihood of occurrence of future events. Team-forming and Temporal

[1] K. Solovey and M. Pavone are with the Department of Aeronautics & Astronautics, Stanford University, Stanford, CA, 94305, {kirilsol,pavone}@stanford.edu.

[2] S. Bandyopadhyay, F. Rossi, and M. T. Wolf are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, {saptarshi.bandyopadhyay, federico.rossi, michael.t.wolf}@jpl.nasa.gov.

Partitioning Algorithms [15] group heterogeneous robots in teams so that each team is capable of performing all the tasks that might arise. Mixed-Integer Linear Programming (MILP)-based algorithms [16] explicitly represent the task allocation problem as a mixed-integer program. Algorithms in this class can readily capture a rich family of constraints and accommodate heterogeneous robot capabilities. Markov chain-based algorithms [17] model robots' motion according to a stochastic policy prescribed by a Markov chain optimized according to a given cost function. While all those approach cover a wide range of problems and techniques, they generally either do not scale well with the problem size or provide weak theoretical guarantees on the solution cost or runtime.

The predictive nature of our problem strongly relates to the notion of online algorithms [18], [19], [20], which are designed for problems in which the input is revealed gradually, while optimizing a goal function (e.g., minimizing cost). The typical benchmark for online algorithms is the *worst case* ratio between the solution of the online algorithm and the optimal solution for the *offline* case in which the entire input is given a priori. This is termed as the algorithm's *competitive ratio*. The *k-server problem* [21], [22], which has been studied in this online context, bears some resemblance to our setting. In particular, it can be viewed as the centralized case consisting of a single fleet of $k > 1$ homogenenous agents, where the goal is to collect all the rewards while minimizing travel cost in an online fashion. A recent paper introduced an online randomized algorithm for the problem which achieves the best known competitive ratio of $O(\log^6 k)$ [23]. The weighted variant of the problem, which is closer to our heterogeneous setting, requires an even larger competitive ratio of $\Omega(2^{2^{k-4}})$ [24].

Applications of homogeneous task allocation have been extensively explored recently within the setting of transportation and logistics. For instance, the operation of an autonomous mobility-on-demand system, requires to assign ground vehicles to routes in order to fulfill passenger demand, while potentially accounting for road congestion [25], [26], [27]. A recent work proposes an efficient package delivery framework consisting of multiple drones [28]. This work proposes utilize public-transit vehicles on which the drones can hitchhike in order to conserve their limited energy, and thus noticeably increase the drones' service range. From a broader perspective, task allocation can be viewed through the lenses of the vehicle routing problem (VRP) [29] or the orienteering problem (OP) [30]. However, save a few special cases, VRP and OP are typically approached with mixed integer linear programming (MILP) formulations that scale poorly, or by heuristics that do not provide optimality guarantees.

Finally, we mention that in case that the locations of the tasks are deterministic and known in advance, our problem can be cast into an instance of the multi-agent pathfinding (MAPF) problem [31]. Here the goal is to compute a collection of paths—one for each agent—such that a certain criteria is optimized, while accounting for inter-agent conflict constraints. In our setting, a constraint ensuring that a reward for a given task will be assigned to at most one agent in a given time step. Unfortunately, solving MAPF optimaly is NP-Hard [32], and no polynomial-time approximation algorithms that can solve general MAPF instances exist, to the best of our knowledge. We mention a recent approach for MAPF termed conflict-based search [33], [34], [35], [36] that earned some popularity due to its efficiency in moderately-sized instances. However, it does not provide run-time guarantees and do not scale well in settings that require considerable amount of coordination between agents.

B. Contribution

In general, to the best of our knowledge, no algorithm is available to solve the *predictive* task allocation problem for time-varying reward collection for teams of *heterogeneous* efficiently. Thus, we present an efficient algorithm for predictive heterogeneous task-allocation achieving an approximation factor of at least 1/2 of the optimal reward. Our algorithm does not only enjoy from favorable guarantees in theory, but also good performance in practice. In particular, through simulation experiments, we show that our algorithm is faster by several orders of magnitude than a MILP-based approach, and its runtime scales modestly with the problem size. Moreover, we demonstrate that the runtime is insensitive to the number of agents preset in each fleet. From an algorithmic standpoint, our approach demonstrates that the heterogeneous problem can be decomposed into several homogeneous subproblems that can be solved efficiently using min-cost flow.

The organization of this paper is as follows. In Section II we provide basic definitions and problem formulation. In Section III we first study the homogeneous subproblem and demonstrate that it can be solved optimality using min-cost flow. Then, in Section IV we demonstrate that the heterogeneous problem can be broken into several homogeneous subproblems, to yield polynomial-time near-optimal algorithm. In Section V we provide experimental results attesting to good performing and scalability of our approach. We conclude this work with discussion and future work in Section VI.

II. Preliminaries and Problem Formulation

We first describe the problem ingredients and then proceed to a formal definition of the problem. The robots' workspace is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices $i \in \mathcal{V}$ denoting physical regions for the robots, and edges $(i, j) \in \mathcal{E}$ denoting transitions between regions (See example in Fig. 1.) For a given $i \in \mathcal{V}$, we use $\mathcal{E}^+(i)$ to denote the set of all vertices $j \in \mathcal{V}$ such that $(i, j) \in \mathcal{E}$. We also define $\mathcal{E}^-(i)$ to be the set of all vertices $j \in \mathcal{V}$ such that $(j, i) \in \mathcal{E}$.

We consider a discrete-time, finite-horizon framework, where the horizon is specified by some $T \in \mathbb{N}_{>0}$. We use $\tau \in [T]$ to denote a given type step (without loss of generality, we assume that the current time step is 0), where $[r] := \{0, \dots, r\}$ for a non-negative integer $r \in \mathbb{N}_{\geqslant 0}$.

Throughout this paper, we refer to the robots as "robots" or "agents" interchangeably. The set of all agents is denoted by $\mathcal{A}$, which is subdivided into F disjoint fleets $\mathcal{A}^1, \dots, \mathcal{A}^F$, where agents within the same fleet are considered to be of homogeneous capabilities. We denote by $a_f = |\mathcal{A}^f|$ the number of agents in a given fleet f, and by $\mathcal{F} = \{1, \dots, F\}$ the set of fleet indices.

Given the graph $\mathcal{G}$, time horizon $T \in \mathbb{N}_{>0}$, agent fleets $\mathcal{A}^1, \ldots, \mathcal{A}^F$, initial positions p_0, and potential rewards $R^0, \ldots, R^F$,

$$\begin{aligned} & \text{maximize} && \mathcal{R}(x,y,z) := \sum_{i\in\mathcal{V}} \sum_{\tau\in[T]} \sum_{f\in\mathcal{F}} \left(R^0_\tau[i] \cdot y^f_\tau[i] + R^f_\tau[i] \cdot z^f_\tau[i] \right), && \text{(1a)} \\ & x^f_\tau[i,j] \in [a_f] && \forall (i,j) \in \mathcal{E},\ f \in \mathcal{F},\ \tau \in [T-1], \\ & y^f_\tau[i] \in \{0,1\} && \forall i \in \mathcal{V},\ f \in \mathcal{F},\ \forall \tau \in [T], \\ & z^f_\tau[i] \in \{0,1\} && \forall i \in \mathcal{V},\ f \in \mathcal{F},\ \forall \tau \in [T], \\ & \text{subject to} \\ & \sum_{j\in\mathcal{E}^+(i)} x^f_0[i,j] = p^f_0[i], && \forall i \in \mathcal{V}, && \forall f \in \mathcal{F}, && \text{(1b)} \\ & \sum_{i\in\mathcal{E}^-(j)} x^f_{\tau-1}[i,j] = \sum_{\ell\in\mathcal{E}^+(j)} x^f_\tau[j,\ell], && \forall j \in \mathcal{V} && \forall f \in \mathcal{F}, && \forall \tau \in [T-2]\setminus\{0\}, && \text{(1c)} \\ & y^f_\tau[i] \leqslant \sum_{j\in\mathcal{E}^-(i)} x^f_\tau[i,j], && \forall i \in \mathcal{V}, && \forall f \in \mathcal{F}, && \tau \in [T-1], && \text{(1d)} \\ & y^f_T[j] \leqslant \sum_{i\in\mathcal{E}^-(j)} x^f_{T-1}[i,j], && \forall j \in \mathcal{V}, && \forall f \in \mathcal{F}, && && \text{(1e)} \\ & \sum_{f\in\mathcal{F}} y^f_\tau[i] \leqslant 1, && \forall i \in \mathcal{V}, && && \forall \tau \in [T], && \text{(1f)} \\ & z^f_\tau[i] \leqslant \sum_{j\in\mathcal{E}^+(i)} x^f_\tau[i,j], && \forall i \in \mathcal{V}, && \forall f \in \mathcal{F}, && \tau \in [T-1], && \text{(1g)} \\ & z^f_T[j] \leqslant \sum_{i\in\mathcal{E}^+(j)} x^f_{T-1}[i,j], && \forall j \in \mathcal{V}, && \forall f \in \mathcal{F}, && && \text{(1h)} \\ & z^f_\tau[i] \leqslant 1, && \forall i \in \mathcal{V}, && \forall f \in \mathcal{F} && \forall \tau \in [T]. && && \text{(1i)} \end{aligned}$$

TABLE I
Definition of the heterogeneous task-allocation problem.

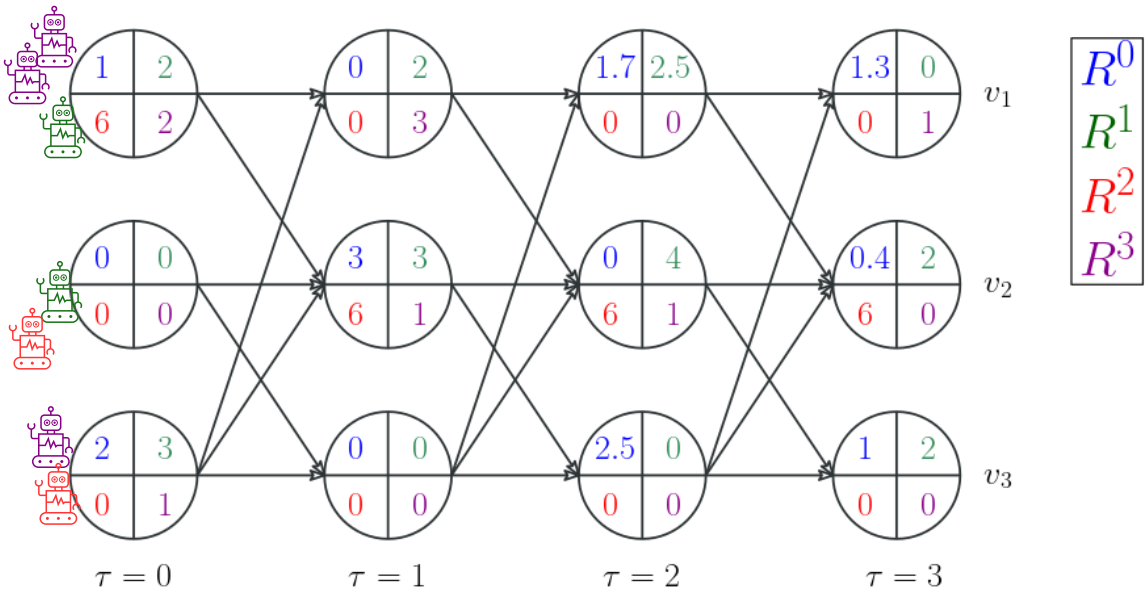

Fig. 1. Illustration of the problem setting. In this example, the workspace $\mathcal{G}$, which consists of three vertices v_1, v_2, v_3 and seven edges, is expanded over the time horizon $T = 3$. There are F fleets of heterogeneous agents, where $a_1 = 3$ (green), $a_2 = 2$ (red), $a_3 = 2$ (magenta), and the initial positions are illustrated for $\tau = 0$. For every vertex $v \in \mathcal{V}$ and time step τ, the values in the corresponding time-expanded vertex represent the following rewards: shared reward $R^0_\tau[v]$ (blue), and private reward $R^1_\tau[v], R^2_\tau[v], R^3_\tau[v]$ for fleets 1 (green), 2 (red), and 3 (magenta), respectively.

The agents are mobile and transition from one vertex $i \in \mathcal{V}$ to another $j \in \mathcal{V}$ every time step, assuming that $(i,j) \in E$.[1] For a given fleet $f \in \mathcal{F}$, the positions of its agents at time $\tau \in [T]$ are specified by $p^f_\tau \in [a_f]^{|\mathcal{V}|}$, where for a given $i \in \mathcal{V}$, $p^f_\tau[i]$, specifies the number of agents of $\mathcal{A}^f$ located at the vertex.

[1] Agents can stay put in vertex i if $(i,i) \in \mathcal{E}$.

A. Shared and Private Task Sets

The problem consists of allocating agents to tasks which consist of collecting rewards along the time-expanded vertices of $\mathcal{G}$. For now, we assume that the distributions of rewards are known in advance. An extension in which rewards are predicted according to a stochastic process is discussed later on.

There are $F+1$ types of rewards $R^0, R^1, \ldots, R^F$, which determine the values gained by the agents for visiting any given vertex $i \in \mathcal{V}$ at time $\tau \in [T]$. The type $t \in [F]$ denotes the ability of agents to collect the reward, or to execute a task associated with the reward. Specifically, rewards of type $t = 0$ are considered to be *shared*, i.e., can be collected by any robot of any fleet $f \in \mathcal{F}$. In particular, for any $i \in \mathcal{V}, \tau \in [T]$, the system gains the reward $R^0_\tau[i_\tau]$ if $\sum_{f\in\mathcal{F}} p^f_\tau[i] \geqslant 1$, i.e., at least one agent (from any fleet) visits the vertex i at time τ. In contrast, tasks of type $t \neq 0$ are considered to be *private*, and can only be gained via agents from the particular fleet $\mathcal{A}^f$ such that $f = t$. That is, when $t \neq 0$ then for any $i \in \mathcal{V}, \tau \in [T]$, the system gains the reward $R^t_\tau[i]$ if $p^t_\tau[i] \geqslant 1$, i.e., at least one agent from $\mathcal{A}^t$ visits the vertex i at time τ.

B. Problem Formulation

We are in a position to provide a formal definition of our problem in the form of an integer program. The input to the problem consists of the workspace graph G, time horizon $T \in \mathbb{N}_{>0}$, agent fleets $\mathcal{A}^1, \ldots, \mathcal{A}^F$ with $F \in \mathbb{N}_{>0}$ with known initial positions $p^f_0[i]$ for all $i \in V$, and potential rewards $R^0, \ldots, R^F$.

The goal of this work is to obtain a task-allocation scheme, which maximizes the total collected reward. The task-allocation scheme consists of (i) specifying the locations of all agents for every $\tau \in [T]$ and (ii) assigning agents to rewards. We present a formal description of the problem in the form of an integer program in Table I below.

The scheme is described through two types of decision variables. We use the discrete variable $x^f_\tau[i,j] \in [a_f]$ to denote a transition of agents in $\mathcal{A}^f$ from $i \in \mathcal{V}$ to $j \in \mathcal{V}$, at time $\tau \in [T-1]$, assuming that $(i,j) \in \mathcal{E}$. We use the decision variable $y^f_\tau[i] \in \{0,1\}$ to indicate whether an agent from $\mathcal{A}^f$ is assigned to collect the shared reward $R^0_\tau[i]$. Similarly, for a given $f \in \mathcal{F}$, the variable $z^f_\tau[i] \in \{0,1\}$ indicates whether an agent in $\mathcal{A}^f$ is assigned to collect the private reward $R^f_\tau[i]$.

The objective function is given in Equation (1a) in Table I. Equation (1c) ensures the flow conservation of agents. Equations (1d), (1e), (1g), (1h) ensure that an agent is assigned to a task only if it is present in the corresponding vertex. Equations (1f), (1i) limit the number of agents assigned to every type of task in a given vertex to 1.

C. Predictive Formulation

We mention that that the above formulation can be easily adapted to the predictive setting. Namely, in this case the precise value of the rewards $R^0, \ldots, R^f$ are only known for the current timestep $\tau = 0$, but a stochastic model of the evolution of rewards with respect to time is available in order to predict the rewards' behavior. To exploit the aforementioned deterministic formulation, it is straightforward to show that by plugging into the problem defined in Table I the expected values of the stochastic rewards, the solution would maximize the expected gained reward. This in turn gives rise to a receding-horizon implementation: given the current state of the system (e.g., locations of agents and values of current rewards) we predict the reward values $R^0, \ldots, R^F$ for T time steps into the future, and then compute a plan specified by x, y, z. We then execute this plan for the first time step, obtain current values of the reward, and repeat this process all over again.

For example, suppose we have two tasks which represent tracking of two distinct objects (one in each task), where the reward for successful tracking are $1, 2$, respectively. Also assume that the objects are confined to a 3×3 grid graph, and that in each time step, each object selects one of its neighbors uniformly at random (and cannot stay in the same vertex). Figure 2 shows the expected reward R^t for the two task classes (depicted in blue and red respectively) across three time steps.

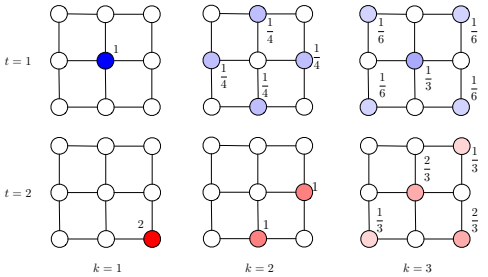

Fig. 2. Expected reward for two task classes. The tasks require tracking of an object moving to adjacent cells according to a random walk.

D. Discussion

A few comments are in order. We note that both the problem formulation in Table I and our approximation algorithm can be straightforwardly extend to a setting where the goal is to maximize the reward minus the travel cost, where the latter is represented by costs assigned to the edges of G.

Our current problem formulation and algorithm do not account for collisions between agents that can arise when two or more agents reside in the same vertex of the graph, as is required in a MAPF formulation wherein agents are typically operating in close proximity to each other. However, for space applications, as we mention in Section I, collisions are typically not an issue due to the vast areas over which the agents operate. We do mention that we plan to explore algorithmic extensions that avoid collisions as well (see Section VI).

Finally, we mention that a setting with a similar partition between homogeneous and heterogeneous subproblems has be explored in [37], albeit for a different problem formulation consisting of computing multiple travelling-salesman routes in an undirected and time-invariant graph.

III. Algorithm for the Homogeneous Case

In preparation to tackling the heterogeneous problem presented in the previous section, we first consider the simplified homogeneous subcase. Namely, the homogeneous problem consist of a single fleet of homogeneous agents $\mathcal{A}^f$ and a single individual reward set R^f. In this section we show that an optimal solution to the homogeneous problem can be found efficiently using an approach for the min-cost flow problem,

which will be defined below. An algorithm for solving the homogeneous setting will then be employed in the next section as a subroutine to achieve a polynomial-time approximation for the heterogeneous case.

Given the graph $\mathcal{G}$, time horizon $T \in \mathbb{N}_{>0}$, agent fleet A^f, initial positions p_0, and potential rewards R^f,

$$\text{maximize} \quad \mathcal{R}^f(x,z) := \sum_{i\in\mathcal{V}} \sum_{\tau\in[T]} R^f_\tau[i] \cdot z^f_\tau[i], \tag{2a}$$

subject to (1b), (1c), (1g), (1h), (1i), with respect to $\mathcal{A}^f$.

TABLE II
Definition of the heterogeneous task-allocation problem.

For a given reward set R^f and initial positions of $\mathcal{A}^f$ encoded through 0^f, we denote by $\mathcal{H}(R^f, p^f_0)$ the homogenenous optimization problem in Table III. Note that the problem of assigning the set of shared tasks R^0 to all the agents $\mathcal{A}$, while ignoring the assignment of private tasks, can be viewed as the homogeneous problem $\mathcal{H}(R^0, p)$.

The motivation for considering the homogeneous case is given in the following theorem, which states that an optimal solution to this problem can be obtained in (low-degree) polynomial time.

Theorem III.1. *For any $f \in \mathcal{F}$, the optimal solution for $\mathcal{H}(R^f, p^f_0)$ (similarly for $\mathcal{H}(R^0, p_0)$) can be computed in*

$$\mathcal{O}\left(T^2 mn \log(Tn) + T^2 n^2 \log(Tn)\right)$$

time, where $m = |\mathcal{E}|, n = |V|$.

Proof. In this proof we show that the homogeneous problem can be transformed into a min-cost flow (MCF) problem [38], for which an optimal solution can be efficiently found. In the remainder of this proof we define the ingredients for MCF, then provide a formal definition of the problem, and then discuss its complexity.

MCF is defined for a directed graph whose edges are weighted and have capacity constraints. For a given instance $\mathcal{H}(R^f, p^f)$ we define the graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where

$$\mathbb{V} = \{s, g\} \cup \{v^i_\tau, w^i_\tau | i \in \mathcal{V}, \tau \in T\}.$$

That is, the vertex set consists of the vertices s and g, which will be used as the source and sink of the flow, respectively, as well as two copies v^i_τ, w^i_τ of every vertex $v \in \mathcal{V}$ for each time step $\tau \in T$.

The edge set is defined for several subsets $\mathbb{E} = E_s \cup E_t \cup E_0 \cup E_R \cup E_1$, where

$$\begin{aligned}
E_s &= \{(s, v^i_0) | \exists q \in \mathcal{A}^f p^q_0[i] \neq 0\},\\
E_g &= \{(w^i_T, g) | i \in \mathcal{V}\},\\
E_0 &= \{(v^i_\tau, w^i_\tau) | i \in \mathcal{V}, \tau \in [T]\},\\
E_R &= \{(v^i_\tau, w^i_\tau) | i \in \mathcal{V}, \tau \in [T], R^f_\tau[i] \neq 0\},\\
E_\mathcal{E} &= \{(w^i_\tau, v^j_{\tau+1}) | (i,j) \in \mathcal{E}, \tau \in [T-1]\}.
\end{aligned}$$

Namely, edges in E_s connect the source vertex s to all the vertices which function as start positions for the agents. Edges in E_g connect all the vertices in the final time step T to the sink node g. Edges in E_0 connect between every two copies v^i_τ, w^i_τ of a vertex $i \in V$ for a given time step τ. Agents traversing the latter set of edges will receive no reward. On the other hand, every edge $(v^i_\tau, w^i_\tau) \in E_R$ will be used to represent the collection of the reward $R^f_\tau[i]$ by the agent traversing this edge. Edges in $E_\mathcal{E}$ simulate the traversal of agents along the edges of $\mathcal{E}$ between one time step to the next.

Next, we assign costs c for the edges in $\mathbb{E}$. In particular, for any $e \in \mathbb{E} \backslash E_R$ we set $c_e := 0$. For a given $e' \in (v^i_\tau, w^i_\tau) \in E_R$ we set $c_{e'} := -R^f_\tau[i]$. The final ingredient to the MCF problem is edge capacities u. For any $e \in E_g \cup E_0 \cup E_\mathcal{E}$ we set infinite capacity $u_e := \infty$. To ensure that the correct number of agents will arrive to the starting positions, we specify for a given $e' = (s, v^i_0) \in E_s$ the capacity $u_{e'} := \sum_{q\in\mathcal{A}^f} p^q_0[i]$. Finally, to ensure that every reward will be collected by at most one agent, we specify $u_{e''} := 1$ for any $e'' \in E_R$.

We are ready to provide the corresponding formulation of MCF. The goal is to assign flow values $h_e \in [0, \infty)$, which represent the number of agents traversing e, to all edges $e \in \mathbb{E}$ in order to minimize the expression $\sum_{e\in\mathbb{E}} h_e c_e$ under the constraints

$$h_e \in [0, u_e], \forall e \in \mathbb{E}, \tag{3a}$$

$$\sum_{e\in E_s} h_e = |\mathcal{A}^f|, \tag{3b}$$

$$\sum_{e\in E_g} h_e = |\mathcal{A}^f|, \tag{3c}$$

$$\sum_{v'\in\mathbb{E}^+(v)} h_{vv'} - \sum_{v'\in\mathbb{E}^-(v)} h_{v'v} = 0, \forall v \in \mathbb{V} \setminus \{s, g\}. \tag{3d}$$

Constraint (3a) ensures that capacities are maintained; (3b) ensures that all the agents will leave the source; (3c) ensures that all agents will end up at the sink vertex[2]; the final constraint ensures equality between the number of agents leaving and entering a vertex $v \in \mathbb{V} \setminus \{s, g\}$.

By definition of $\mathbb{G}, c, u$ its clear that the above MCF formulation is equivalent to the homogeneous problem, with one slight change. Namely, MCF permits fractional assignments to h, whereas the values of x, z are integral. If all edge capacities are integral (which is the case in our setting), the linear relaxation of MCF enjoys a totally-unimodular constraint matrix form [39]. Hence, the fractional solution will necessarily have an integer optimal solution.

We note that the above MCF problem can be solved in $\mathcal{O}((M' + N) \log N(M + N \log N))$ time using Orlin's MCF algorithm [40], [38], where $M = |\mathbb{E}| = \mathcal{O}(Tm)$, $N = |\mathbb{V}| = \mathcal{O}(Tn)$, and M' is the number of $\mathbb{G}$ edges whose capacity is finite. In our case $M' = \mathcal{O}(Tn)$, which implies that the total time complexity of MCF is $\mathcal{O}(T^2 mn \log n + T^2 n^2 \log^2 n)$. □

IV. Algorithm for the Heterogeneous Case

In this section we present an efficient algorithm to tackle the task-allocation problem described in Table I. Recall that we wish to find an assignment x, y, z such that the expression $\mathcal{R}(x, y, z)$ is minimized, where for a given $f \in \mathcal{F}$, $\tau \in [T]$, $j \in V$, $x^f_\tau[i, j]$ represents the transitions of the agents in $\mathcal{A}^f$, and $y^f_\tau[i]$, $z^f_\tau[i]$ indicate whether it is assigned in vertex i to execute $R^0_\tau[i]$ and $R^f_\tau[i]$ respectively.

[2]For our specific graph structure $\mathbb{G}$, and considering the constraint (3d), the constraint (3c) is redundant. We leave it for clarity and completeness.

For a given $f \in \mathcal{F}$, denote by x^f the corresponding values of the x assignment for agents of fleet f, i.e., $x^f = \{x^f_\tau\}_{\tau \in [T-1]}$ for $\tau \in [T]$. The sets y^f, z^f are similarly defined.

A. Algorithm

We present the private-first-shared-first (PFSF) algorithm for heterogeneous task allocation (see Algorithm 1). It accepts as input the specification of the agents and the entire reward set $\mathbb{R} = \{R^0, R^1, \ldots, R^F\}$. It then obtains two candidate solutions $(x, y, z), (x', y', z')$ and returns the one that yields the larger reward of the two.

Algorithm 1: PFSF $(\mathbb{R}, p_0)$

```
(x, y, z) ← PrivateFirst(ℝ, p_0);
(x', y', z') ← SharedFirst(ℝ, p_0);
if R(x, y, z) > R(x', y', z') then
    return (x, y, z);
return (x', y', z');
```

The subroutine PRIVATEFIRST (Algorithm 2) prioritizes the assignment of private rewards over shared rewards. This is achieved by assigning to each fleet $f \in \mathcal{F}$ a new reward set $\hat{R}^f$ which consists of the combination of its private reward set R^f and the shared reward set R^0, where the latter's rewards are rescaled by F^{-1} with $F = |\mathcal{F}|$. That is, $\hat{R}^f_\tau[j] = R^f_\tau[j] + R^0_\tau[j] \cdot F^{-1}$. An assignment over $\hat{R}^f$ for every $f \in \mathcal{F}$ is then obtained. Note that z^f implicitly encodes both an assignment to a private reward, as well as the shared one. That is, an agent assigned to perform a reward $\hat{R}^f_\tau[j]$ can be interpreted as being assigned to both $R^f_\tau[j]$ and $R^0_\tau[j]$. In lines 4-8 the solutions of the individual fleets are combined to eliminate cases where several agents (from different fleets) are assigned to the same shared reward. To do so, for a given τ, j, we go iteratively over all fleets $f \in \mathcal{F}$ and set $y^f_\tau[j] = 1$ for the first agent we encounter that is assigned to a shared reward in the corresponding vertex.

Algorithm 2: PRIVATEFIRST$(\mathbb{R}, p_0)$

```
1  R̂^f ← R^f + R^0 · F^{-1}, ∀f ∈ F;
2  (x^f, z^f) ← H(R̂^f, p_0^f), ∀f ∈ F;
3  x ← {x^f}_{f∈F}, z ← {z^f}_{f∈F};
4  y ← {{0}_{q∈A}}_{τ∈[T]};
5  for τ ∈ [T], j ∈ V do
6      for f ∈ F do
7          if z^f_τ[j] == 1 then
8              y^f_τ[j] ← 1;
9              break ;
10 return (x, y, z);
```

The SHAREDFIRST subroutine (Algorithm 3) prioritizes the assignment of shared rewards, by first computing the an assignment for all the agents $\mathcal{A}$ to the shared reward set R^0, to maximize the total reward. This is achieved by calling $\mathcal{H}(R^0, p_0)$. It then generates an updated reward set $\bar{R}^f$ for every fleet $f \in \mathcal{F}$, where the value of a reward $\bar{R}^f_\tau[j]$ is equal to $R^f_\tau[j]$ in case that $R^0_\tau[j]$ was not assigned to f, and otherwise equal to $R^f_\tau[j] + R^0_\tau[j]$. Next, for every fleet f the private assignment (x^f, z^f) over $\bar{R}^f$ is computed. Observe that z represents simultaneously assignments for shared and private rewards.

Algorithm 3: SHAREDFIRST$(\mathbb{R}, p_0)$

```
(x̄, ȳ) ← H(R^0, p_0);
for f ∈ F, τ ∈ [T], j ∈ V do
    R̄^f_τ[j] ← R^f_τ[j] + R^0_τ[j] · ȳ^f_τ[j];
(x^f, z^f) ← H(R̄^f, p_0), ∀f ∈ F;
x ← {x^f}_{f∈F}, z ← {z^f}_{f∈F};
return (x, z, z);
```

B. Analysis

In this section, we show that the PFSF algorithm is guaranteed to achieve a solution within a constant factor of the optimum. Then we provide a computational analysis of the approach.

Let (x, y, z) be an assignment returned by PFSF. With a slight abuse of notation, we use $\mathcal{R}(x^f, y^f, z^f)$ to represent the reward obtained by agents of fleet $f \in \mathcal{F}$ (where assignments values of agents in other fleets are set to 0). Similarly, $\mathcal{R}(x, y, \mathbf{0}), \mathcal{R}(x, \mathbf{0}, z)$ represent the reward gained from shared rewards and private rewards, respectively, where $\mathbf{0}$ represents the zero vector (whose dimension will be clear from context).

Let (X, Y, Z) be the optimal assignment. It will be convenient to define $\text{OPT} := \mathcal{R}(X, Y, Z) = S^* + P^*$, where

$$S^* := \mathcal{R}(X, Y, \mathbf{0}),$$
$$P^* := \mathcal{R}(X, \mathbf{0}, Z) = \sum_{f \in \mathcal{F}} \mathcal{R}(X^f, \mathbf{0}, Z^f).$$

We are ready to state our main theoretical contribution:

Theorem IV.1 (Approximation factor of PFSF)**.** *Let (x, y, z) be the solution returned by* PFSF*. Then $\mathcal{R}(x, y, z) \geqslant \text{OPT} \cdot \frac{F}{2F-1}$.*

Before proceeding to the proof of the main theorem, we establish two intermediate results, concerning the guarantees of PRIVATEFIRST and SHAREDFIRST, when considered separately. In particular, we show that each of the subroutines provide complementary approximations with respect to S^* and P^*, and PFSF enjoys the best of both worlds. See illustration in Figure 3

Lemma IV.2. *Approximation factor of* PRIVATEFIRST *] Let (x, y, z) be the solution returned by* PRIVATEFIRST*. Then $\mathcal{R}(x, y, z) \geqslant \Delta \cdot S^* \cdot F^{-1} + P^*$.*

Proof. Fix $f \in \mathcal{F}$ and note that

$$\mathcal{R}(x^f, y^f, z^f) \geqslant \mathcal{R}\left(X^f, \mathbf{0}, Z^f\right) + F^{-1} \cdot \mathcal{R}(X^f, Y^f, \mathbf{0}),$$

since PRIVATEFIRST could choose to assign $(x^f, z^f) = (X, Y + Z)$ (line 2), which would yield the desired result.

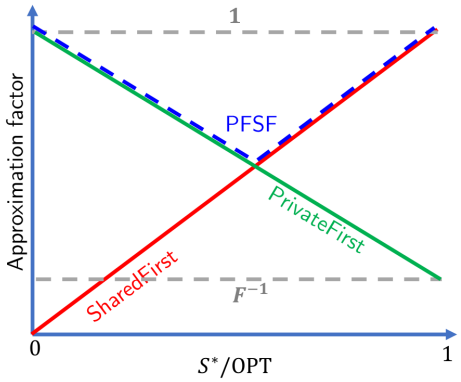

Fig. 3. Visualization of the approximation factors (y-axis) achieved by PFSF (dashed blue curve), PRIVATEFIRST (green curve), SHAREDFIRST (red curve), as a function of the ratio between S^* and OPT.

Hence,

$$\begin{aligned}\mathcal{R}(x,y,z) &= \sum_{f\in\mathcal{F}} \mathcal{R}(x^f,y^f,z^f)\\ &\geqslant \sum_{f\in\mathcal{F}} \mathcal{R}\left(X^f,\mathbf{0},Z^f\right) + F^{-1}\sum_{f\in\mathcal{F}} \mathcal{R}(X^f,Y^f,\mathbf{0})\\ &= P^* + F^{-1}\cdot S^*.\end{aligned}$$

□

Lemma IV.3 (Approximation factor of SHAREDFIRST)**.** *Let (x,y,z) be the solution returned by* SHAREDFIRST*. Then $\mathcal{R}(x,y,z)\geqslant S^*$.*

Proof. The proof is straightforward, as it must hold that $\mathcal{R}(\bar{x},\bar{y},\mathbf{0})\geqslant\mathcal{R}(X,Y,\mathbf{0})$, where $\bar{x},\bar{y}$ are as defined in line 1 of SHAREDFIRST. □

We are ready for the main proof:

Proof of Theorem IV.1. Due to Lemma IV.2, and Lemma IV.3, it is clear that PFSF obtains a solution (x,y,z) with reward at least $\max\left\{P^*+F^{-1}\cdot S^*, S^*\right\}$. Next, observe that if either of S^* or P^* is zero, the PFSF would necessarily return the optimal solution. Now, assume that S^* and P^* are strictly positive, and hence there exists $\Delta>0$ such that $P^*=\Delta S^*$. We have that

$$\begin{aligned}\frac{\mathcal{R}(x,y,z)}{\mathcal{R}(X,Y,Z)} &\geqslant \frac{\max\left\{\Delta\cdot S^*+F^{-1}\cdot S^*, S^*\right\}}{\Delta S^*+S^*}\\ &= \max\left\{\frac{\Delta+F^{-1}}{\Delta+1}, \frac{1}{\Delta+1}\right\}\\ &\geqslant \max\left\{\frac{\frac{F-1}{F}+F^{-1}}{\frac{F-1}{F}+1}, \frac{1}{\frac{F-1}{F}+1}\right\}\\ &= \frac{F}{2F-1},\end{aligned}$$

where we used the fact that $\operatorname{argmin}_{\Delta>0}\max\left\{\frac{\Delta+F^{-1}}{\Delta+1}, \frac{1}{\Delta+1}\right\} = (F-1)/F$. □

We conclude this section with a computational analysis of the overall approach.

Corollary IV.4. PFSF *can be implemented in $\mathcal{O}(FT^2mn\log n + FT^2n^2\log^2 n)$ time.*

Proof. This result directly follows from Theorem III.1, as we perform $2F+1$ computations of $\mathcal{H}$, where each computation is a bottleneck for both PRIVATEFIRST and SHAREDFIRST. □

V. EXPERIMENTS

In this section we validate our theoretical results from the previous section through simulation experiments. In particular, we demonstrate that our PFSF algorithm scales well and finds a near-optimal solutions fairly quickly on commodity hardware. Moreover, we show that our approach is faster than a MILP-based approach by several order of magnitude, and we observe that the approximation factors that we achieve in practice are better than the worst-case bound of $\frac{F}{2F-1}$. Additionally, in accordance with the complexity analysis, we observe experimentally that the running time of PFSF is unaffected by number of agents within each fleet (as opposed to the number of fleets).

A. Implementation and Scenario Details

All results were obtained using a commodity laptop equipped with 2.80GHz × 4 core i7-7600U CPU, and 16GB of RAM, running 64bit Ubuntu 18.04 OS. We implemented the PFSF algorithm in C++, where we used the LEMON Graph Library for the implementation of min-cost flow [41]. To the best of our knowledge, this library provides the fastest implementation of min-cost flow algorithms [42] (we mention that we used the network-simplex algorithm, which turned to be the fastest in our experiments). We compared our approach with a MILP-based solution using CPLEX [43].

We tested both implementations on the predictive problem formulation described in Section II-C. In particular, for a given graph structure G, time horizon T, fleets number F, and initial object count $I\in\mathbb{N}_+$, we choose uniformly at random for each $t\in[F]$, I vertices of G (with repetitions) which represent initial locations of objects to be tracked as part of task t. That is we set the values of R_0^t to be the number of objects at the corresponding vertices. For the next time steps $\tau\in[T]\setminus\{0\}$ we set R_τ^t to be the expected reward assuming that the objects move to neighboring vertices in a uniform random fashion, as depicted in Figure 2. We chose the initial locations of the agents in a uniform random fashion as well.

B. Results

We report results for several test cases. In particular, we first compare the performance of the MILP solution and our PFSF algorithm, in terms of value of gained reward and running time. We then proceed to study the scalability of the PFSF algorithm on larger test cases.

1) Comparison between PFSF and MILP: In this setup, we fix the graph G to be a 10×10 grid, set the initial number of tracked objects for every task to be $I=3$, and set the number of agents within each fleet $f\in\mathcal{F}$ to be $a_f=5$ (below we test PFSF on much larger fleets with $a_f=500$). In Table III we report the running time of the MILP solution and the PFSF algorithm, as well as the approximation factor that was achieved by PFSF, for scenarios of varying sizes. In particular, we set both the time horizon T and the fleet number F to be between 2 and 128. The numbers in the table represent the average results over 20 randomly-generated scenarios for

each parameter combination. We terminate the run of each of the algorithms if it exceeded a budget of 10 minutes.

In terms of running time, we observe that MILP behaves similarly to PFSF only for the smallest test cases, e.g., when $T = 2$. However, as the problem size increases the running time of MILP grows significantly faster than that of PFSF. For instance, already for $T = 4$, when $F = 2$ PFSF is nearly 3 times faster than MILP, and when $F = 128$ it is more than 20 times faster. As T is increased we start to see more scenarios in which the MILP is forced to time out, whereas PFSF finishes fairly quickly. For example, when $T = 16$, PFSF finishes within a few seconds, for all fleet sizes, whereas MILP times out for $F = 128$, which yields a speedup of at least 100 for PFSF. For $T = 128$, MILP was able to solve only the smallest scenario, whereas PFSF solved all.

In terms of solution quality, we observe that PFSF typically achieves approximation factors that are larger than the theoretical $F/(2F - 1)$ lower bound, which suggest that the real lower bound should be higher. We note that the smallest approximation factor that we observed with PFSF is 0.64.

2) Scalability of PFSF: We rerun the previous experiment with the PFSF algorithm for a larger grid graph of dimensions 50×50 to test how the runtime of the algorithm is affected by its different parameters. We report in Table IV the runtime of the algorithm, where $I = 3, a_f = 5$ were set as in the previous experiment. In accordance with the theoretical complexity bound, we observe that the runtime increases linearly with the number of fleets F. Interestingly, the time horizon T increases linearly as well, which suggests that theoretical quadratic increase is overly conservative.

Finally, we note that runtime of PFSF is only mildly affected by the size of each individual fleet. That is, when we increase the value of a_f from 5 to 50 and 500, we only observe a slight increase of no more than %10 to the runtime. This is in contrast to the MILP based approach which is highly sensitive to this value.

VI. Conclusion

In this paper, we present a predictive task assignment algorithm for heterogeneous agents. We prove the algorithm's optimality-bound and demonstrate its effectiveness using simulations. We envisage this algorithm will see wide-spread use in many multi-agent and swarm applications.

Our work suggests a few interesting directions for future research, which we highlight below. From an algorithmic perspective we plan to explore whether the approximation bound on the gained reward can be improved by potentially introducing a third subroutine, in addition to privateFirst and sharedFirst, which better estimates the cost in situations where the existing routines provide worse approximation (see Figure 3). We also aim to explore whether our algorithmic framework can be extended to the setting of the problem where collisions between agents should be avoided, as in the MAPF problem. Another important future endeavor would be to extend PFSF to the distributed setting, by possibly relying dual decomposition that would be applied to the homogeneous subproblems [44, Chapter 7].

Acknowledgments

Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The authors thank Xiaoshan Bai, Shushman Choudhury, Devansh Jolta, Erez Karpas, and Javier Alonso-Mora for fruitful discussions.

References

[1] R. Lillis, D. Mitchell, L. Montabone, S. Guzewich, S. Curry, P. Withers, M. Chaffin, T. Harrison, C. Ao, N. Heavens *et al.*, "Mars orbiters for surface-atmosphere-ionosphere connections (mosaic)," *LPI*, no. 2326, p. 1733, 2020. 1

[2] S. Krishnamoorthy, A. Komjathy, J. A. Cutts, P. Lognonne, R. F. Garcia, M. P. Panning, P. K. Byrne, R. S. Matoza, A. D. Jolly, J. B. Snively, S. Lebonnois, and D. Bowman, "Seismology on venus with infrasound observations from balloon and orbit." Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), White Paper for the NASA 2021 Decadal Survey SAND2020-2849R 684580, 3 2020. 1

[3] N. Stacey and S. D'Amico, "Autonomous swarming for simultaneous navigation and asteroid characterization," in *AAS/AIAA Astrodynamics Specialist Conference*, 2018. 1

[4] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004. 1

[5] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013. 1

[6] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2020. 1

[7] N. Agatz, P. Bouman, and M. Schmidt, "Optimization Approaches for the Traveling Salesman Problem with Drone," *Transportation Science*, vol. 52, no. 4, pp. 965–981, 2018. 1

[8] S. M. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich, "Optimization of a Truck-Drone in Tandem Delivery Network using k-means and Genetic Algorithm," *Journal of Industrial Engineering and Management*, vol. 9, no. 2, pp. 374–388, 2016. 1

[9] C. C. Murray and A. G. Chu, "The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-Assisted Parcel Delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86 – 109, 2015. 1

[10] X. Wang, S. Poikonen, and B. Golden, "The Vehicle Routing Problem with Drones: Several Worst-Case Results," *Optimization Letters*, vol. 11, no. 4, pp. 679–697, Apr 2017. 1

[11] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone, "Review of multi-agent algorithms for collective behavior: a structural taxonomy," in *IFAC Workshop on Networked & Autonomous Air & Space Systems*, 2018, in Press. 1

[12] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004. 1

[13] N. Ayanian, "DART: Diversity-enhanced autonomy in robot teams," *The International Journal of Robotics Research*, p. 0278364919839137, 2017. 1

[14] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1259–1274, 2011. 1

[15] S. L. Smith and F. Bullo, "The dynamic team forming problem: Throughput and delay for unbiased policies," *Systems & control letters*, vol. 58, no. 10-11, pp. 709–715, 2009. 2

[16] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs," in *Cooperative control: models, applications and algorithms*. Springer, 2003, pp. 23–41. 2

[17] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and distributed control of a large-scale swarm of autonomous agents," *IEEE Trans. Robotics*, vol. 33, no. 3, pp. 1103–1123, 2017. 2

[18] A. Fiat and G. J. Woeginger, *Online algorithms: The state of the art*. Springer, 1998, vol. 1442. 2

[19] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 2005. 2

time horizon	criterion	fleets 2	4	8	16	32	64	128
2	MILP	0.00	0.01	0.01	0.03	0.07	0.16	0.36
	PFSF	0.01	0.01	0.02	0.03	0.06	0.12	0.26
	APX	1.00	1.00	0.93	0.86	0.64	0.75	0.75
4	MILP	0.02	0.04	0.07	0.16	0.39	0.90	1.77
	PFSF	0.01	0.03	0.05	0.09	0.17	0.35	0.67
	APX	1.00	0.97	0.87	0.86	0.77	0.74	0.85
8	MILP	0.11	0.26	0.58	1.39	3.62	12	21
	PFSF	0.04	0.07	0.13	0.25	0.46	0.94	1.85
	APX	0.96	0.92	0.82	0.80	0.84	0.88	0.94
16	MILP	0.45	1.40	3.78	20	93	453	DNF
	PFSF	0.10	0.19	0.36	0.67	1.27	2.54	5.92
	APX	0.96	0.93	0.88	0.92	0.89	0.95	
32	MILP	1.5	5.6	20	284	567	DNF	DNF
	PFSF	0.29	0.52	1.00	1.87	3.76	7.11	16
	APX	0.94	0.89	0.82	0.88	0.94		
64	MILP	4.78	21	203	DNF	DNF	DNF	DNF
	PFSF	0.84	1.50	2.78	5.22	12	20	43
	APX	0.97	0.91	0.92				
128	MILP	14.20	DNF	DNF	DNF	DNF	DNF	DNF
	PFSF	2.48	4.85	8.25	17	31	59	115
	APX	0.96						

TABLE III

Comparison between PFSF and a MILP-based solution in terms of running time and solution quality for 10×10 grid graphs. For every combination of number of fleets F and time horizon T we report in the "MILP" and "PFSF" rows the corresponding running times (in seconds) of the two algorithms. The label "DNF" denotes that MILP did not finish within the 10 minutes time limit. In the "APX" row we report the approximation factor achieved by PFSF, i.e., denoting the quotient between the reward achieved by PFSF and optimal reward obtained by MILP (this value is only reported if MILP managed to find a solution within the time limit).

time horizon	fleets 2	4	8	16	32	64	128
2	0.1	0.2	0.4	0.8	1.4	4.8	7.3
4	0.3	0.6	1.1	2.2	4.2	13	19
8	1.0	1.6	3.2	6.6	12	38	55
16	3	5	9	19	35	118	214
32	9	14	29	59	132	214	414
64	31	48	100	189	375	700	1423
128	74	154	299	518	994	2040	4055

TABLE IV

Running time (seconds) of PFSF for a 50×50 grid graph as a function of the fleets number F and time horizon T.

[20] P. V. Hentenryck and R. Bent, *Online stochastic combinatorial optimization*. The MIT Press, 2009. 2

[21] E. Koutsoupias, "The k-server problem," *Computer Science Review*, vol. 3, no. 2, pp. 105 – 118, 2009. 2

[22] D. Bertsimas, P. Jaillet, and N. Korolko, "The k-server problem via a modern optimization lens," *European Journal of Operational Research*, vol. 276, no. 1, pp. 65 – 78, 2019. 2

[23] J. R. Lee, "Fusible HSTs and the randomized k-server conjecture," in *Foundations of Computer Science*, 2018, pp. 438–449. 2

[24] N. Bansal, M. Eliáš, and G. Koumoutsos, "Weighted k-server bounds via combinatorial dichotomies," in *Foundations of Computer Science*, 2017, pp. 493–504. 2

[25] K. Solovey, M. Salazar, and M. Pavone, "Scalable and Congestion-Aware Routing for Autonomous Mobility-on-Demand via Frank-Wolfe Optimization," in *Robotics: Science and Systems*, 2019. 2

[26] A. Wallar, M. V. D. Zee, J. Alonso-Mora, and D. Rus, "Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4539–4546. 2

[27] M. W. Levin, "Congestion-aware system optimal route choice for shared autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 82, pp. 229 – 247, 2017. 2

[28] S. Choudhury, K. Solovey, M. Kochenderfer, and M. Pavone, "Efficient Large-Scale Multi-Drone Delivery using Transit Networks," in *International Conference on Robotics and Automation*, 2020. 2

[29] P. Toth and D. Vigo, *Vehicle Routing – Problems, Methods, and Applications*, 2nd ed., K. Scheinberg, Ed. SIAM, 2014. 2

[30] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315 – 332, 2016. 2

[31] J. Yu and S. M. LaValle, "Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016. 2

[32] J. Yu and S. M. LaValle, "Structure and Intractability of Optimal Multi-robot Path Planning on Graphs," in *AAAI Conference on Artificial Intelligence*, 2013. 2

[33] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges," in *Symposium on Combinatorial Search*, 2017. 2

[34] H. Ma, J. Li, T. Kumar, and S. Koenig, "Lifelong Multi-agent Path Finding for Online Pickup and Delivery Tasks," in *International Conference on Autonomous Agents and Multiagent Systems*, 2017, pp. 837–845. 2

[35] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Conflict-based Search with Optimal Task Assignment," in *International Conference on Autonomous Agents and Multiagent Systems*, 2018, pp. 757–765. 2

[36] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path Planning for Multi-Agent Pickup and Delivery," in *International Conference on Autonomous Agents and Multiagent Systems*, 2019, pp. 1152–1160. 2

[37] A. Prasad, H. Choi, and S. Sundaram, "Min-max tours and paths for task allocation to heterogeneous agents," *IEEE Transactions on Control of Network Systems*, 2020. 4

[38] D. P. Williamson, *Network Flow Algorithms*. Cambridge University Press, 2019. 5

[39] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Pearson, 1993. 5

[40] J. B. Orlin, "A Faster Strongly Polynomial Minimum Cost Flow Algorithm," *Operations Research*, vol. 41, no. 2, pp. 338–350, 1993. 5

[41] "LEMON Graph Library," https://lemon.cs.elte.hu/trac/lemon, accessed: 07.2020. 7

[42] P. Kovács, "Minimum-cost flow algorithms: an experimental evaluation," *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015. 7

[43] IBM, "Ilog cplex optimization studio," 2020. 7

[44] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Belmont, USA: Athena Scientific, 2016. 8