2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Fast Near-Optimal Heterogeneous Task Allocation via Flow Decomposition

Kiril Solovey[1], Saptarshi Bandyopadhyay[2], Federico Rossi[2], Michael T. Wolf[3], and Marco Pavone[1]

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9560880

Abstract— **Multi-robot systems are uniquely well-suited to performing 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 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 heterogeneous capabilities (e.g., availability of appropriate sensors or actuators) to ensure the feasibility of execution, and to promote a higher reward, over a long time horizon. To this end, we present the FLOWDEC algorithm for efficient heterogeneous task-allocation, and show that it achieves an approximation factor of at least $1/2$ of the optimal reward. Our approach decomposes the heterogeneous problem 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 approach.**

I. INTRODUCTION

A central problem in many multi-robot applications, including patrolling, information gathering, and pick-up and delivery problems, is *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 a task-allocation setting where mobile heterogeneous robots need be assigned to time-varying sets of tasks, or, equivalently, rewards. In our formulation, robots are divided into homogeneous fleets based on their ability to collect *private* reward sets (where each fleet is associated with a unique private set). There is also a *shared* reward set that can be collected by robots in any fleet. A robot is rewarded for a specific reward set if (i) it resides in the spatial vicinity of the reward, (ii) the reward is either shared or private to the robot's specific fleet, and (iii) no other robot is assigned to this reward.

A number of problems of interest fall in this setting, including object tracking, intruder following and imaging a scientific phenomenon. 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], networks of balloons to detect seismic and volcanic events on Venus [2–4], and swarms of small spacecraft to study small bodies [5]. In all of these applications, the heterogeneous task-allocation problem is central: a set of robots 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 robots.

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). However, most variants of the problem are known to be computationally prohibitive to solve [6, 7]. Thus, many approaches for task allocation scale poorly with the problem size (e.g., the number of robots) or provide no guarantees on the solution quality or runtime. Furthermore, heterogeneous robot capabilities (e.g., ground vehicles and aerial drones jointly working to achieve a mutual goal), impose additional challenges for designing practical, high-quality solution approaches [8–12].

Several algorithm types have been proposed specifically for task allocation by the robotics community (see e.g. the survey in [13]). In auction-based algorithms [6, 14, 15], 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 [16] rely on partitioning the workspace into regions and assigning each region to one or multiple robots. Tasks within a region are assigned to the robot (or robots) responsible for that region. Spatial partitioning algorithms capture the likelihood of occurrence of future events. Team-forming and temporal partitioning algorithms [17] group heterogeneous robots in teams so that each team is capable of performing all the tasks that might arise. Mixed-integer linear programming (MILP) approaches [18] explicitly represent the task-allocation problem as a mixed-integer program, and can readily capture a variety of constraints including heterogeneity. Markov chain-based

[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 and F. Rossi are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109; {saptarshi.bandyopadhyay, federico.rossi}@jpl.nasa.gov.

[3]M. T. Wolf was with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109 for this work. He is now with Amazon Robotics AI. wolf@robotics.caltech.edu.

algorithms [19] model the robots' motion via a stochastic policy prescribed by a Markov chain optimized according to a given cost function. While all those approaches 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 quality or runtime.

Applications of homogeneous task allocation have been extensively explored recently within the setting of transportation and logistics. E.g., 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 [20–22]. Recent work develops efficient package delivery framework consisting of multiple drones in which drones are assigned to packages and delivery routes [23], and proposes utilizing public-transit vehicles on which drones can hitchhike in order to conserve their limited energy, thus noticeably increasing their service range. From a broader perspective, task allocation can be viewed through the lens of the vehicle routing problem (VRP) [24] or the orienteering problem (OP) [25]. However, save a few special cases, VRP and OP are typically approached with MILP formulations that scale poorly, or heuristics that do not provide optimality guarantees.

The problem that we address in this paper can be represented as a multi-agent pathfinding (MAPF) problem [26]. The goal in MAPF is to compute a collection of paths for the agents to minimize execution time, while accounting for inter-agent conflict constraints. Unfortunately, no polynomial-time approximation algorithms that can solve general MAPF instances exist [27], to the best of our knowledge. A recent approach for MAPF termed conflict-based search [28–31] earned popularity due to its efficiency in moderately-sized instances. However, it does not provide run-time guarantees and does not scale well in settings that require considerable amount of coordination among agents.

Contribution. We present an efficient approximation algorithm, termed FLOWDEC, for heterogeneous task-allocation in the context of maximizing the collection of time-varying rewards. From the theoretical perspective we prove that our algorithm achieves an approximation factor of at least $1/2$ of the optimal reward in polynomial time. The algorithm also exhibits good performance in practice. Specifically, in simulation experiments, we demonstrate that the algorithm achieves a speedup of several orders of magnitude over a MILP approach, and its runtime scales modestly with the problem size. Moreover, we prove that the runtime is insensitive to the number of agents present in each fleet.

From an algorithmic standpoint, the FLOWDEC algorithm decomposes the heterogeneous problem into several homogeneous subproblems, that can be solved efficiently using min-cost flow [32], without significant degradation in solution quality. Our approach shares some similarity with a recent work that also considers homogeneous decomposition [33], albeit the problem setting (computing multiple travelling-salesman routes in an undirected and time-invariant graph), as well as the algorithmic techniques and analysis that are developed there, are quite different from ours.

Organization. The organization of this paper is as follows. In Section II we provide basic definitions and the problem formulation. In Section III we first study the homogeneous

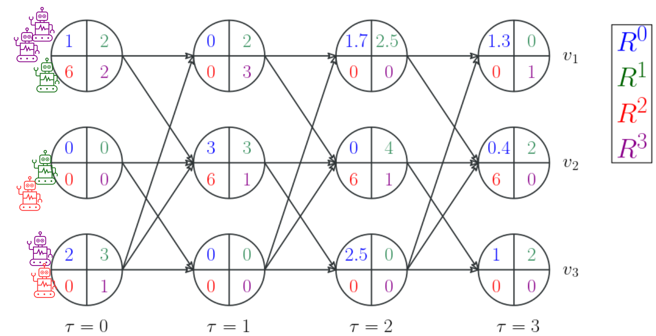

Fig. 1: Illustration of the problem setting. In this example, the workspace $\mathcal{G}$, which consists of three vertices v_1, v_2, v_3 and seven edges (going from, e.g., the vertices at time $\tau = 0$ to $\tau = 1$), is expanded over the time horizon $T = 3$. There are $F = 3$ fleets of heterogeneous agents, where $a_1 = 2$ (green), $a_2 = 2$ (red), $a_3 = 3$ (magenta). 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 initial positions are $p^1_0 = (1,1,0), p^2_0 = (0,1,1), p^3_0 = (2,0,1)$.

subproblem. Then, in Section IV we describe the FLOWDEC algorithm and provide its theoretical analysis. In Section V we provide experimental results demonstrating the good performance and scalability of our approach. We conclude with a discussion of future research directions 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 locations- for the robots, and edges $(i, j) \in \mathcal{E}$ denoting transitions between locations (see example in Figure 1). We use $\mathcal{E}^+_i$ to denote the set of outgoing neighbors of a vertex $i \in \mathcal{V}$, namely $\{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. We similarly define $\mathcal{E}^-_i$ to represent the set of incoming neighbors, i.e., $\{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$.

We consider a discrete-time, finite-horizon framework, where the horizon is specified by a positive integer T. We use $\tau \in [0\,..\,T]$ to denote a given time step, where $[r\,..\,r']$ denotes an integer interval $\{k \in \mathbb{N} : r \leqslant k \leqslant r'\}$ between two integers such that $r < r'$.

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 $\mathcal{A}^1, \ldots, \mathcal{A}^F$, where agents within the same fleet are assumed to have homogeneous capabilities. We denote by $a_f = |\mathcal{A}^f|$ the number of agents in a fleet $f \in \mathcal{F}$, where $\mathcal{F} := [1\,..\,F]$ is the set of fleet indices. The agents are mobile and transition from one vertex $i \in \mathcal{V}$ to another $j \in \mathcal{V}$ every time step, assuming that $(i, j) \in \mathcal{E}$. For a fleet $f \in \mathcal{F}$, the positions of its agents at time $\tau \in [0\,..\,T]$ are specified by the vector $p^f_\tau \in [0\,..\,a_f]^{|\mathcal{V}|}$, the value $p^f_\tau[i]$ specifies the number of agents of $\mathcal{A}^f$ located at a vertex $i \in \mathcal{V}$. Figure 1 provides a graphical representation of the problem formulation.

A. Shared and private rewards

The problem consists of allocating agents to rewards along the time-expanded vertices of $\mathcal{G}$. For now, we assume that the distributions of rewards are known in advance. We discuss a

Given the graph $\mathcal{G}$, time horizon T, agent fleets $\mathcal{A}^1,\ldots,\mathcal{A}^F$, initial positions p_0, and reward sets $R^0,\ldots,R^F$, the objective is to *maximize*

$$\mathcal{R}(x,y,z) := \sum_{j\in\mathcal{V}}\sum_{\tau\in[0..T]}\sum_{f\in\mathcal{F}}\left(R^0_\tau[j]\cdot y^f_\tau[j] + R^f_\tau[j]\cdot z^f_\tau[j]\right) \quad \text{(1a)}$$

with the decision variables

$x^f_\tau[i,j]\in[0\,..\,a_f]$, $\quad\forall(i,j)\in\mathcal{E}, f\in\mathcal{F},\tau\in[0\,..\,T-1]$,

$y^f_\tau[j]\in\{0,1\}, z^f_\tau[j]\in\{0,1\}$, $\quad\forall j\in\mathcal{V}, f\in\mathcal{F},\tau\in[0\,..\,T]$,

subject to

$$\sum_{i\in\mathcal{E}^+_j} x^f_0[j,i] = p^f_0[j], \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \quad \text{(1b)}$$

$$\sum_{i\in\mathcal{E}^-_j} x^f_{\tau-1}[i,j] = \sum_{\ell\in\mathcal{E}^+_j} x^f_\tau[j,\ell], \quad \forall j\in\mathcal{V}, f\in\mathcal{F}, \tau\in[1\,..\,T-2], \quad \text{(1c)}$$

$$\sum_{i\in\mathcal{E}^+_j} x^f_\tau[j,i] \geqslant y^f_\tau[j], \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \tau\in[0..T-1], \quad \text{(1d)}$$

$$\sum_{i\in\mathcal{E}^-_j} x^f_{T-1}[i,j] \geqslant y^f_T[j], \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \quad \text{(1e)}$$

$$\sum_{f\in\mathcal{F}} y^f_\tau[j] \leqslant 1, \qquad \forall j\in\mathcal{V}, \forall\tau\in[0\,..\,T], \quad \text{(1f)}$$

$$\sum_{i\in\mathcal{E}^+_j} x^f_\tau[j,i] \geqslant z^f_\tau[j], \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \tau\in[0..T-1], \quad \text{(1g)}$$

$$\sum_{i\in\mathcal{E}^-_j} x^f_{T-1}[i,j] \geqslant z^f_T[j], \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \quad \text{(1h)}$$

$$z^f_\tau[j] \leqslant 1, \qquad \forall j\in\mathcal{V}, f\in\mathcal{F}, \tau\in[0\,..\,T]. \quad \text{(1i)}$$

TABLE I: Definition of the heterogeneous task-allocation problem.

predictive extension where the reward sets are not known in advance in Section II-C.

There are $F+1$ types of reward sets $R^0, R^1, \ldots, R^F$, which determine the values gained by the agents for visiting any given vertex $i\in\mathcal{V}$ at time $\tau\in[0\,..\,T]$, and there are constraints that specify which fleets $f\in\mathcal{F}$ can collect a specific reward R^t of type $t\in[0\,..\,F]$. 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 a given vertex $i\in\mathcal{V}$ and time step $\tau\in[0\,..\,T]$, 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, rewards 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 $f=t$. I.e., when $t\neq 0$ then for any $i\in\mathcal{V},\tau\in[0\,..\,T]$, 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 provide a formal definition of our problem in the form of an integer program (Table I). The input to the problem consists of the workspace graph $\mathcal{G}$, 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 rewards sets $R^0,\ldots,R^F$. 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 time step $\tau\in[0\,..\,T]$ and (ii) assigning agents to rewards.

The solution is described through two types of decision variables. The integer variable $x^f_\tau[i,j]\in[0\,..\,a_f]$ denotes a transition of agents in fleet $f\in\mathcal{F}$ from vertex $i\in\mathcal{V}$ to $j\in\mathcal{V}$, at time $\tau\in[0\,..\,T-1]$, assuming that $(i,j)\in\mathcal{E}$. The decision variables $y^f_\tau[i]\in\{0,1\}$ and $z^f_\tau[i]\in\{0,1\}$ indicate whether an agent from fleet $f\in\mathcal{F}$ is assigned to collect the shared reward $R^0_\tau[i]$, or private reward $R^f_\tau[i]$, respectively.

The objective function is given in Eq. (1a). Eq. (1b) ensures that agents will start at their initial positions as specified by p_0. Eq. (1c) ensures the continuity of the agents. Eq. (1d), (1e), ensure that an agent from fleet f is assigned to a shared reward $R^0_\tau[j]$ only if one of the agents of this fleet is at vertex j in time τ; similarly, Eq. (1g) and (1h) enforce this condition with respect to private reward sets. Eq. (1f), (1i) limit the number of agents assigned to every reward type in a given vertex to 1.

We mention that our problem formulation and solution approach can be easily extended to accommodate edge traversal costs by modifying the objective functions in Tables I and II. For simplicity of presentation we do not account for edge costs in the current formulation.

C. Predictive setting

The above formulation can be extended to the predictive setting, where we are given the values of the reward sets for the first time step $R^0_0,\ldots,R^f_0$, and a stochastic model of the evolution of rewards with respect to time. 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 gives rise to a receding-horizon implementation: given the current state of the system (e.g., locations of agents and values of current rewards) and the current time step τ we predict the reward values $R^0,\ldots,R^F$ for T time steps into the future, and compute a corresponding solution x^τ, y^τ, z^τ. We then execute this solution for the first time step, obtain current values of the reward, and repeat this process in the next time step $\tau+1$. For the simplicity of presentation, we shall focus on the static setting of the problem from now on. We do note that our experiments are for the predictive case.

D. Discussion

A few comments are in order. First, the discrete nature of the graph representation is well-suited to the discrete, combinatorial nature of the task allocation problem, where agents must be assigned to *discrete* tasks. The formulation is highly flexible: for example, vertices can represent tasks to be performed, events to be observed, or regions to be surveyed, with edges representing the travel time between these events. The approach is also amenable to settings where events and tasks appear in a continuous workspace, via appropriate discretization. However, in such cases a careful and problem-specific selection of the appropriate degree of discretization is critical to ensure good performance.

Our approach does not account for collision avoidance between the agents. In typical autonomy frameworks (e.g., [34, 35]), team-level decisions such as task allocation are handled at a different, higher level compared to collision avoidance, which is often resolved through local controllers (e.g., [23, 29–31]). Unless the workspace is heavily cluttered with obstacles, such a hierarchical architecture typically results in good performance, while offering greatly-reduced complexity compared to a hypothetical integrated architecture.

III. Algorithm for the homogeneous case

In preparation to the FlowDec algorithm for the heterogeneous problem, we describe a key ingredient, which is an efficient solution to the homogeneous problem. The homogeneous problem consists of maximizing the collected reward for a single fleet $f \in \mathcal{F}$ of homogeneous agents and a private reward set R^f. Our main insight is that an optimal solution to the homogeneous problem can be found efficiently by solving a min-cost flow (MCF) problem [32].

Given the graph $\mathcal{G}$, time horizon T, agent fleet A^f, initial positions p_0^f, and private reward set R^f, $$\text{maximize} \quad \mathcal{R}^f(x,z) := \sum_{j \in \mathcal{V}} \sum_{\tau \in [0..T]} R^f_\tau[j] \cdot z^f_\tau[j],$$ subject to (1b), (1c), (1g), (1h), (1i), with respect to $\mathcal{A}^f$.

TABLE II: Definition of the heterogeneous task-allocation problem.

Denote by $\mathcal{H}(R^f, p_0^f)$ the homogenenous optimization problem described in Table II, for a private reward set R^f and initial positions p_0^f of a fleet $f \in \mathcal{F}$. Note that the problem of assigning the set of shared rewards R^0 to all the agents $\mathcal{A}$, while ignoring the assignment of private rewards, can be viewed as the homogeneous problem $\mathcal{H}(R^0, p)$. The following lemma states that an optimal solution for the homogeneous problem can be obtained in polynomial time.

Lemma 1 (Efficient solution of $\mathcal{H}$)**.** *For a given fleet $f \in \mathcal{F}$, private reward set R^f, and initial positions p_0^f, the optimal solution for the homogeneous problem $\mathcal{H}(R^f, p_0^f)$ can be computed in $\mathcal{O}\left(T^2 mn \log(Tn) + T^2 n^2 \log(Tn)\right)$ time, where $m = |\mathcal{E}|, n = |\mathcal{V}|$. This bound also holds for the homogeneous problem $\mathcal{H}(R^0, p_0)$ with the shared reward R^0.*

To prove this result we show that the homogeneous problem $\mathcal{H}(R^f, p_0^f)$ can be equivalently represented as MCF. Given a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, edge capacities u_e and costs c_e for every edge $e \in \mathbb{E}$, the objective in MCF is to assign flow values h_e to each edge e, to minimize the total flow cost $\sum_{e \in \mathbb{E}} h_e \cdot c_e$, while satisfying edge capacity constraints u_e. In the full proof of Lemma 1, which we defer to the extended version of the paper [36], we define the MCF ingredients $\mathbb{G}, u, c$ such that the constraints of the resulting MCF problem correspond to the constraints of the homogeneous problem $\mathcal{H}(R^f, p_0^f)$, and the flow variables h describe the agents' locations x^f and reward assignments z^f. Moreover, an optimal solution to this MCF problem yields an optimal solution to $\mathcal{H}(R^f, p_0^f)$. The runtime complexity bound in Lemma 1 follows from using Orlin's algorithm for MCF [37].

IV. Algorithm for the heterogeneous case

We present an efficient algorithm, which we call FlowDec, for the heterogeneous task-allocation problem described in Table I. The FlowDec algorithm decomposes the problem into several homogeneous subproblems that are solved using min-cost flow, as described in Section III. In the remainder of this section we describe FlowDec, and determine its approximation and runtime guarantees.

A. *The* FlowDec *algorithm*

The FlowDec algorithm (Algorithm 1) accepts as input the agent fleets $\mathcal{A}^1, \ldots, \mathcal{A}^F$, initial positions p_0, and reward sets $\mathbb{R} := \{R^0, R^1, \ldots, R^F\}$. Recall that we wish to find an assignment x, y, z such that the expression $\mathcal{R}(x,y,z)$ is maximized. Also recall that for a given fleet $f \in \mathcal{F}$, timestep $\tau \in [0..T]$, and vertices $i, j \in V$, $x^f_\tau[i,j]$ represents the transitions of the agents in $\mathcal{A}^f$ from i to j, and $y^f_\tau[i]$, $z^f_\tau[i]$ indicate whether those agents are assigned to collect the rewards $R^0_\tau[i]$ and $R^f_\tau[i]$, respectively, at vertex i. For a given fleet $f \in \mathcal{F}$, denote by x^f the corresponding values of the x assignment for agents belonging to fleet f, i.e., $x^f = \{x^f_\tau\}_{\tau \in [0..T-1]}$. The sets y^f, z^f are similarly defined.

FlowDec computes two candidate solutions using the subroutines PrivateFirst and SharedFirst, respectively, and returns the one that yields the larger reward of the two. Next we elaborate on those two subroutines.

Algorithm 1: FlowDec $(\mathbb{R}, \mathcal{A} = \{\mathcal{A}^1, \ldots, \mathcal{A}^F\}, p_0)$

```
(x, y, z) ← PrivateFirst(ℝ, 𝒜, p_0);
(x′, y′, z′) ← SharedFirst(ℝ, 𝒜, 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 $\hat{R}^f$ that combines the private reward set R^f and the shared reward set R^0, where the latter is rescaled by F^{-1}, i.e., $\hat{R}^f_\tau[j] = R^f_\tau[j] + R^0_\tau[j] \cdot F^{-1}$. An assignment over $\hat{R}^f$ for every $f \in \mathcal{F}$ is then obtained by solving the homogeneous problem $\mathcal{H}(\hat{R}^f, p_0^f)$. Note that z^f implicitly encodes both an assignment to a private reward and shared reward. 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 5-9 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 time step τ and vertex j, we iterate over all fleets $f \in \mathcal{F}$ and assign $y^f_\tau[j] = 1$ for the first agent we encounter that is assigned to a shared reward in the corresponding vertex.

Algorithm 2: PrivateFirst$(\mathbb{R}, \mathcal{A}, p_0)$

```
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∈[0..T]};
5  for τ ∈ [0..T], j ∈ 𝒱 do
6      for f ∈ 𝓕 do
7          if z_τ^f[j] == 1 then
8              y_τ^f[j] ← 1;
9              break ;
10 return (x, y, z);
```

The SharedFirst subroutine (Algorithm 3) prioritizes the assignment of shared rewards, by first computing an assignment for all the agents $\mathcal{A}$ to the shared reward set R^0, to maximize the total reward. This is achieved by solving the homogeneous problem $\mathcal{H}(R^0, p_0)$. It then generates an updated private 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 $R^f_\tau[j] + R^0_\tau[j]$, for every time step τ and vertex j. Next, for every fleet f the private assignment (x^f, z^f) over $\bar{R}^f$ is computed by solving $\mathcal{H}(\bar{R}^f, p^f_0)$. Note that z here represents simultaneously assignments for shared and private rewards.

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

```
1 (x̄, ȳ) ← H(R^0, p_0);
2 for f ∈ F, τ ∈ [0..T], j ∈ V do
3 |   R̄^f_τ[j] ← R^f_τ[j] + R^0_τ[j] · ȳ^f_τ[j];
4 (x^f, z^f) ← H(R̄^f, p^f_0), ∀f ∈ F;
5 x ← {x^f}_{f∈F}, z ← {z^f}_{f∈F};
6 return (x, z, z);
```

B. Analysis of FLOWDEC

We prove that the FLOWDEC algorithm is guaranteed to achieve a solution within a constant factor of the optimum. Let (x, y, z) be a solution of FLOWDEC. We use $\mathcal{R}(x^f, y^f, z^f)$ to represent the portion of the total reward $\mathcal{R}(x, y, z)$ that is attributed to fleet $f \in \mathcal{F}$ (where assignment values of agents in other fleets are set to 0). Similarly, denote by $\mathcal{R}(x, y, \mathbf{0})$ and $\mathcal{R}(x, \mathbf{0}, z)$ the shared and private portion of the total reward, respectively, where $\mathbf{0}$ represents the zero vector (whose dimension will be clear from context).

Let (X, Y, Z) be a solution to the heterogeneous problem (Table I) that maximizes the expression $\mathcal{R}(X, Y, Z)$ and define OPT $:= \mathcal{R}(X, Y, Z) = S^* + P^*$, where $S^* := \mathcal{R}(X, Y, \mathbf{0})$ and $P^* := \mathcal{R}(X, \mathbf{0}, Z) = \sum_{f \in \mathcal{F}} \mathcal{R}(X^f, \mathbf{0}, Z^f)$. We are ready to state our main theoretical contribution:

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

Before proceeding to the proof 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 FLOWDEC enjoys the best of both worlds.

Claim 1 (PRIVATEFIRST's solution quality). *If (x, y, z) is the solution returned by* PRIVATEFIRST*, then $\mathcal{R}(x, y, z) \geqslant P^* + F^{-1} \cdot S^*$.*

Proof. Fix a fleet $f \in \mathcal{F}$ 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 is free to choose $(x^f, z^f) := (X^f, Y^f + Z^f)$ (line 2 in Algorithm 2). Hence,

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

□

Claim 2 (SHAREDFIRST's solution quality). *If (x, y, z) is the solution returned by* SHAREDFIRST *then $\mathcal{R}(x, y, z) \geqslant S^*$.*

Proof. The proof follows from the inequality $\mathcal{R}(\bar{x}, \bar{y}, \mathbf{0}) \geqslant \mathcal{R}(X, Y, \mathbf{0})$, where $\bar{x}, \bar{y}$ are defined Algorithm 3, line 1. □

We are ready for the main proof:

Proof of Theorem 1. Claims 1 and 2 imply that FLOWDEC obtains a solution (x, y, z) with reward at least $\max\left\{P^* + F^{-1} \cdot S^*, S^*\right\}$. If either of S^* or P^* is zero, then FLOWDEC would return the optimal solution. Thus, assume instead that S^* and P^* are positive. Thus, there exists $\Delta > 0$ such that $P^* = \Delta S^*$. The approximation factor of FLOWDEC can be expressed as follows:

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

where the last inequality follows from $\operatorname{argmin}_{\Delta > 0} \max\left\{\frac{\Delta + F^{-1}}{\Delta + 1}, \frac{1}{\Delta + 1}\right\} = (F-1)/F$, which is used since we wish to find a lower-bound that is applicable to any combination of P^* and S^*. □

We conclude with a runtime analysis of FLOWDEC.

Corollary 1 (FLOWDEC runtime). FLOWDEC *can be implemented in $\mathcal{O}(FT^2 mn \log n + FT^2 n^2 \log^2 n)$ time.*

Proof. The bottleneck of FLOWDEC is solving multiple homogeneous subproblems. Since PRIVATEFIRST and SHAREDFIRST solve $2F+1$ homogeneous problems, respectively, and each such computation requires $\mathcal{O}(T^2 mn \log n + T^2 n^2 \log^2 n)$ time (Lemma 1), the total runtime follows. □

V. EXPERIMENTAL RESULTS

We validate our theoretical results from the previous section through simulation experiments. We show that our FLOWDEC approach is faster than a MILP approach by several orders of magnitude, and we observe that the approximation factors that FLOWDEC achieves in practice are higher than the worst-case lower bound of $\frac{F}{2F-1}$. Additionally, we observe experimentally that the running time of FLOWDEC is insensitive to the number of agents in each fleet.

A. Implementation and scenario details

The results were obtained using a laptop with 2.80GHz 4-core i7-7600U CPU, and 16GB of RAM. We implemented the FLOWDEC algorithm in C++, using network simplex for MCF in the LEMON Graph Library [38, 39]. Our code is available at github.com/StanfordASL/heterogeneous-task-allocation. For comparison, we used the the MILP implementation in CPLEX [40] with the default stopping criterion using a relative-gap and absolute tolerances of 1e-4 and 1e-6, respectively.

We tested both implementations on a predictive problem formulation, as described in Section II-C. We consider the problem of heterogeneous tracking of multiple moving objects, where the reward values are chosen to incentivise agents to visit graph vertices where objects are located. In particular, for a given graph $\mathcal{G}$, time horizon T, number of fleets F, and initial object count $I \in \mathbb{N}_+$, we chose uniformly at random for each reward type $t \in [0..F]$, I vertices of $\mathcal{G}$ (with repetitions) which represent initial locations of objects to be tracked as part of the reward set R^t. In particular, we set the value of $R_0^t[i]$ to be the number of objects at a given vertex $i \in \mathcal{V}$. For the subsequent time steps $\tau \in [1..T]$ we set R_τ^t to be the expected reward assuming that the objects move to neighboring vertices via a random walk. Initial agent locations are chosen in a uniform random fashion.

B. Results

We compare the solution quality and runtime of the MILP approach and our FlowDec algorithm. We then study the scalability of the FlowDec algorithm on larger test cases for which the MILP approach has timed out.

1) Comparison between FlowDec *and* MILP*:* In this setup, we fix the graph $\mathcal{G}$ to be a 10×10 grid, set the initial number of tracked objects for every reward to be $I = 3$, and set the number of agents within each fleet $f \in \mathcal{F}$ to be $a_f = 5$. In Table III we report the running time of the MILP solution and the FlowDec algorithm, as well as the approximation factor that was achieved by FlowDec, for scenarios of varying sizes. We set the time horizon T and the fleet number F to be in the range $[2..128]$. The reported running times are averaged over 20 randomly-generated scenarios for each parameter combination. The reported approximation factor is the minimum (i.e., worst) result over the 20 runs. We terminate the run of each algorithm if it exceeds 10 minutes, in which case we do not its running time or the approximation factor.

In terms of running time, we observe that the MILP approach behaves similarly to FlowDec only for the smallest test cases, e.g., when $T \in \{2, 4\}$. However, as the problem size increases the running time of the MILP approach grows significantly faster than that of FlowDec. For instance, already for $T = 8$, when $F = 2$ FlowDec is nearly 3 times faster than the MILP approach, and when $F = 128$ it is more than 10 times faster. As T is increased we encounter more scenarios in which the MILP implementation is forced to time out, whereas FlowDec finishes fairly quickly. For example, when $T = 16$, FlowDec finishes within a few seconds, for all fleet sizes, whereas the MILP approach times out for $F = 128$, which yields a speedup of at least 100 for FlowDec. For $T = 128$, the MILP approach is able to solve only the smallest scenario, whereas FlowDec solves all of them. In terms of solution quality, we observe that FlowDec typically achieves approximation factors that are larger than the theoretical lower bound, which suggests that this bound is loose. We note that the smallest approximation factor that we observed with FlowDec is 0.64 (for $F = 32$).

time hor.	type	fleets: 2	4	8	16	32	64	128
2	MILP	0.00	0.01	0.01	0.03	0.07	0.16	0.36
	Flow	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
	Flow	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
	Flow	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	-
	Flow	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	-	-
	Flow	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	-	-	-	-
	Flow	0.84	1.50	2.78	5.22	12	20	43
	APX	0.97	0.91	0.92				
128	MILP	14.20	-	-	-	-	-	-
	Flow	2.48	4.85	8.25	17	31	59	115
	APX	0.96						

TABLE III: Comparison between FlowDec and a MILP approach in terms of runtime 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 "Flow" rows the corresponding running times (in seconds). The label "-" indicates that MILP did not finish within the 10-minute time limit. In the "APX" row we report the approximation factor of FlowDec, i.e., the quotient between the reward values obtained by FlowDec and MILP, respectively.

time hor.	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 FlowDec for a 50×50 grid as a function of the number of fleets F and time horizon T.

2) Scalability of FlowDec*:* We rerun the previous experiment for a larger 50×50 graph to test how the runtime of the FlowDec 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 in Corollary 1, we observe that the runtime increases linearly with the number of fleets F. The runtime increases linearly with the time horizon T as well, which suggests that the theoretical quadratic increase is overly conservative.

Finally, we note that the runtime of FlowDec is only mildly affected by the size of each individual fleet, e.g., as we increase the value of a_f from 5 to 500, we observe a modest increase of $\%10$ to the runtime. This is in contrast to the MILP approach which is highly sensitive to this value.

VI. Conclusion

We presented a near-optimal algorithm, termed FlowDec, for heterogeneous task allocation and demonstrated its good performance through extensive simulation tests. 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 factor of FlowDec can be improved by introducing a third subroutine which would better estimate the optimal reward for scenarios in which the subroutines SharedFirst and PrivateFirst under approximate. It would also be interesting to test whether a specialized MCF solver for object tracking [41] can speed up the solution of homogeneous problems. We also plan to consider extending our approach to account for collision-avoidance constraints by exploiting the fact that those constraints can be captured via a MCF-based solution for the homogeneous problem. Finally, we plan to extend our algorithm to a distributed setting, by potentially relying on a dual decomposition that would be applied to the homogeneous subproblems [42].

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). This work was supported in part by the Toyota Research Institute (TRI) and the Center for Automotive Research at Stanford (CARS).

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, Albuquerque, NM, White Paper for the NASA 2021 Decadal Survey SAND2020-2849R 684580, 3 2020. 1

[3] S. Krishnamoorthy, V. H. Lai, A. Komjathy, M. T. Pauken, J. A. Cutts, R. F. Garcia, D. Mimoun, J. M. Jackson, D. C. Bowman, E. Kassarian, L. Martire, A. Sournac, and A. Cadu, "Aerial seismology using balloon-based barometers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 10 191–10 201, 2019. 1

[4] A. Didion, A. Komjathy, B. Sutin, B. Nakazono, A. Karp, M. Wallace, G. Lantoine, S. Krishnamoorthy, M. Rud, J. Cutts, P. Lognonné, B. Kenda, M. Drilleau, J. Makela, M. Grawe, and J. Helbert, "Remote sensing of venusian seismic activity with a small spacecraft, the VAMOS mission concept," in *IEEE Aerospace Conference*, 2018, pp. 1–14. 1

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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] N. Ayanian, "DART: diversity-enhanced autonomy in robot teams," *International Journal of Robotics Research*, vol. 38, no. 12-13, 2019. 1

[15] S. Koenig, P. Keskinocak, and C. A. Tovey, "Progress on agent coordination with cooperative auctions," in *Conference on Artificial Intelligence*. AAAI Press, 2010. 1

[16] 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

[17] 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. 1

[18] 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. 1

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

[20] 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

[21] 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

[22] 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

[23] 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, 3

[24] P. Toth and D. Vigo, *Vehicle routing – Problems, methods, and applications*, 2nd ed., K. Scheinberg, Ed. SIAM, 2014. 2

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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, 3

[30] 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, 3

[31] 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, 3

[32] D. P. Williamson, *Network Flow Algorithms*. Cambridge University Press, 2019. 2, 4

[33] 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. 2

[34] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The CLARAty architecture for robotic autonomy," in *IEEE Aerospace Conference Proceedings*, vol. 1, 2001, pp. 1/121–1/132 vol.1. 3

[35] M. T. Wolf, A. Rahmani, J.-P. de la Croix, G. Woodward, J. Vander Hook, D. Brown, S. Schaffer, C. Lim, P. Bailey, S. Tepsuporn, M. Pomerantz, V. Nguyen, C. Sorice, and M. Sandoval, "CARACaS multi-agent maritime autonomy for unmanned surface vehicles in the Swarm II harbor patrol demonstration," in *Unmanned Systems Technology*, vol. 10195, 2017, pp. 1–11. 3

[36] K. Solovey, S. Bandyopadhyay, F. Rossi, M. T. Wolf, and M. Pavone, "Fast near-optimal heterogeneous task allocation via flow decomposition," *CoRR*, vol. abs/2011.03603, 2020. 4

[37] J. B. Orlin, "A faster strongly polynomial minimum cost flow algorithm," *Operations Research*, vol. 41, no. 2, pp. 338–350, 1993. 4

[38] "LEMON Graph Library," https://lemon.cs.elte.hu/trac/lemon, accessed: 07.2020. 5

[39] P. Kovács, "Minimum-cost flow algorithms: an experimental evaluation," *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015. 5

[40] IBM, "ILOG CPLEX optimization studio," 2020. 5

[41] C. Wang, Y. Wang, Y. Wang, C. Wu, and G. Yu, "muSSP: Efficient min-cost flow algorithm for multi-object tracking," in *Neural Information Processing Systems*, 2019, pp. 423–432. 6

[42] D. P. Bertsekas, *Nonlinear programming*, 3rd ed. Belmont, USA: Athena Scientific, 2016. 6