Designing Heterogeneous Robot Fleets for Task Allocation and Sequencing

Nils Wilde and Javier Alonso-Mora

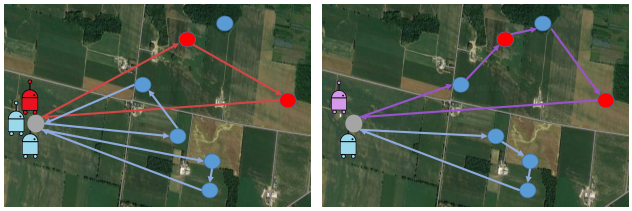
***Abstract*— We study the problem of selecting a fleet of robots to service spatially distributed tasks with diverse requirements within time-windows. The problem of allocating tasks to a fleet of potentially heterogeneous robots and finding an optimal sequence for each robot is known as multi-robot task assignment (MRTA). Most state-of-the-art methods focus on the problem when the fleet of robots is fixed. In contrast, we consider that we are given a set of available robot types and requested tasks, and need to assemble a fleet that optimally services the tasks while the cost of the fleet remains under a budget limit. We characterize the complexity of the problem and provide a Mixed-Integer Linear Program (MILP) formulation. Due to poor scalability of the MILP, we propose a heuristic solution based on a Large Neighbourhood Search (LNS). In simulations, we demonstrate that the proposed method requires substantially lower budgets than a greedy algorithm to service all tasks.**

I. Introduction

Task allocation and sequencing is a fundamental problem in multi-robot systems, with applications in environmental monitoring [1]–[3], service in homes and health-care facilities [4], [5], pickup-and-delivery [6] and autonomous mobility-on-demand [7], [8]. However, many real world problems pose a diverse set of requirements for robot capabilities, leading to specialized robot designs [9], [10]. Deploying a heterogenous fleet of robots offers the most efficient use of resources, yet poses new challenges due to the increased combinatorial complexity.

In this paper, we consider the problem of designing a heterogenous fleet of robots for complex multi-robot missions. Given is a set of available robots with several characteristics such as the capability to service different types of tasks, how they can traverse the environment, their battery life and a deployment cost. The goal is to select robots that can perform their joint mission as well as possible while the total deployment cost remains under a certain budget. For instance, in material transport applications the most efficient solutions might require a combination of high-capacity robots to service densely located tasks as well as cheaper low-capacity robots reaching remote task locations. Similarly, in environmental monitoring data collection might performed by ground vehicles or drones, which differ in their capabilities of traversing the environment, and potentially in the types of measurements they can take.

We illustrate an example in Figure 1. Here a fleet of robots is required to service a set of tasks, each task requires a robot to visit some location in the environment. There are two different types of requirements for the robot to service a task (blue and red). In Figure 1a, a fleet of three low-budget robots is deployed, each robot can only service one task type. Due to battery life limits or tight task deadlines, the fleet is not able to complete all tasks. Figure 1b shows a different fleet where one red and one blue robot are replaced by a single yet more flexible purple robot that can service both task types. This allows for a different allocation of tasks between robots such that all tasks can be serviced.



(a) Fleet 1. (b) Fleet 2.

Fig. 1: Example for a deployment of heterogenous fleets with different capabilities (red and blue) for different task requirements. Fleet 1 consists of three low cost robots that can each only service one task type. Fleet 2 introduces a more versatile robot (purple) that can service both task types.

Designing a heterogeneous fleet for a mission requires algorithms which efficiently find combinations of different robot types that are well adapted to the tasks. In particular, we study the following problem: Given a set of available robot types, assemble a fleet that is able to service the largest number of tasks while the cost of the fleet stays within a budget. First, we characterize the computational complexity of the problem, and present a mixed-linear integer program (MILP) formulation. Due to poor scalability of exact solutions, we propose a heuristic based on a large neighbourhood search (LNS). LNS algorithms iteratively improve solutions to combinatorial problems by removing elements of the solution and subsequently reinserting these elements. Our method takes an integrated approach of simultaneously optimizing i) which robots to include in the fleet and ii) the missions, *i.e.,* tours, executed by each robot. Therefore, we propose two removal heuristics that allow for alternating between improving the current tours and switching out robots in the fleet. In a series of simulation experiments, we show that this approach can find substantially better solutions than a greedy approach for various problem setups.

a) Related Work: Planning and coordination of heterogeneous robot fleets has found wide interest in the multi-robot systems community in recent years [6], [11]–[18]. Yet, few works focus on the problem of deciding which robots to

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

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

include in a heterogeneous fleet. The authors of [9] consider the problem of designing individual underwater robots, and then assembling an optimal fleet. Given the available robots, the fleet design problem is cast into a knapsack variant and solved with a breadth first search. A key difference to our work is that the method is focused on specific mission types and thus imposes different constraints: While we seek to find the best possible fleet given a budget constraint, their work seeks to find a fleet that covers a search area most efficiently. The work of [19] studies the design of homogeneous and heterogeneous robot fleets for logistic warehouses. Both problems are formulated as an Integer Linear Program (ILP). In addition to its focus on pickup-and-delivery, the paper assumes that any type of robot is able to fulfil any task, *i.e.,* transportation request. Our work does not impose this assumption, such that not every fleet is able to service all tasks. Further, we consider additional constraints such as task deadlines and limited battery life. Related to heterogeneous fleet design, researchers investigate the problem of determining the optimal swarm or fleet sizes for homogeneous multi-robot systems [20]–[23]. This either poses a multi-objective optimization problem, trading-off the primary mission objective with the operational cost for the fleet, or a problem of finding the smallest possible fleet such that certain objectives are met. Similarly, we are interested in finding a fleet that maximises the number of serviced tasks given a budget limit for assembling the fleet, yet we consider that multiple robots types are available.

Lastly, our solution is based on a large neighbourhood search (LNS). Such methods have been frequently used to solve multi-robot routing problems such as generalized traveling salesman [24] or MRTA with homogeneous and heterogeneous fleets [17], [25], [26].

b) Contributions: Our main contributions are as follows. First, we characterize hardness of the fleet design problem and provide an MILP formulation of the problem. Second, we present a large neighbourhood search (LNS) algorithm based on two removal heuristics to simultaneously optimize which robots to be employed in the fleet, and what tour each robot takes. In simulation, we show the advantage of the proposed LNS approach over a greedy algorithm in several problem variations.

II. Problem Statement

We consider an offline multi-robot task assignment problem: Given an environment encoded as a graph $G = (V, E)$, a fleet of robots needs to service a fixed set of N tasks $\mathcal{T} = \{T_1, \dots, T_N\}$ where each task requires one of the robots to travel to a vertex $v \in V$, before some deadline t^d. Further, each task has a set of Φ requirements. We write a task as a tuple $T = (v, t^d, \Phi)$.

To service tasks $\mathcal{T}$, we can assemble a fleet of robots by choosing from a set of available robot types $\mathcal{R} = \{R_1, \dots, R_l\}$. Each robot type $R \in \mathcal{R}$ is a tuple $(\Psi, b, \beta, \boldsymbol{D})$. Here Ψ is a set of capabilities to fulfill task requirements; a robot of type R_i can service task T_j when $\Psi_i \supseteq \Phi_j$. Further, $b(R)$ is a fixed deployment cost and $\beta(R)$ the maximum battery life. The matrix $\boldsymbol{D} \in \mathbb{R}_{\geq 0}^{|V|\times|V|}$ describes the traversal time of each edge of the graph for a robot of type R. This does not only encode different speeds for a robot in different parts of the environment, but can also capture the traversability of edges. For instance, when some edges are only usable by drones, a ground robot would have infinite traversal time for these. A fleet is a collection of robots $F = \{r_1, r_2, \dots\}$ where each robot r is of some type R. A fleet can use multiple robots of the same type, making F a multi-set. To service tasks, each robot executes a tour τ, visiting a sequence of task vertices. We do not consider inter-robot collision. Given a fleet F, an MRTA solver π finds a set of tours $Q_\pi(F) = \{\tau_1, \tau_2, \dots\}$ for all robots $r_1, r_2, \dots$ in F that optimizes some measure for the quality of service for tasks in $\mathcal{T}$. In this paper, we consider the objective to be the number of tasks serviced before their deadline, denoted by the functional $\rho(Q_\pi(F), \mathcal{T})$. Finally, the length of each tour $l(\tau_i)$ must remain below the robot's battery limit $\beta(r_i)$. The fleet design problem is then formulated as follows:

Problem 1 (Budgeted fleet design)**.** Given an environment described by $G = (V, E)$, a set of tasks $\mathcal{T}$, an infinite supply of different robot types $\mathcal{R}$ and a budget B, find a fleet of robots F^* and an MRTA solver π, that solve

$$\begin{aligned} &\max_F \rho(Q_\pi(F), \mathcal{T}) \\ &\quad s.t. \sum_{r_i \in F} b(r_i) \leq B \\ &\qquad l(\tau_i) \leq \beta(r_i), \text{ for all } i = 1, \dots, |F| \\ &\qquad r_i \in \mathcal{R} \text{ for all } r_i \in F. \end{aligned} \tag{1}$$

In essence, the problem at hand seeks to find the best fleet given the available robot types to service a set of tasks, while the budget for robots is limited.

III. Method

First, we formulate the problem as an optimization of a set function in order to relate it to well-known combinatorial optimization problems, establishing hardness results, and to present a mixed integer linear program. We then briefly study the greedy algorithm before presenting our proposed method based on a Large Neighbourhood Search (LNS).

A. Approach

We approach Problem 1 by formulating it as a subset selection problem. Given the robot types, let $\mathcal{F}$ be a multi-set containing $\lceil B/b_i \rceil$ robots of each type R_i. By construction, any feasible solution to (1) is a subset of $\mathcal{F}$. This allows us to treat the problem as a heterogeneous variant of the team orienteering problem (TOP) [27], [28] with robots $\mathcal{F}$ where only some robots execute a tour of non-zero length, *i.e. activated*. Thus, let $Q = \{\tau_1, \tau_2, \dots\}$ be the set of tours for all robots $r_1, r_2, \dots$ in the base set $\mathcal{F}$. Further, let q_i be an indicator taking value 1 if $\tau_i \neq \emptyset$ and 0 otherwise. Given a budget B, tours Q are *feasible* when the cost of all robots

with non-empty tours do not exceed the budget, *i.e.,*

$$\sum_{i=1}^{|Q|} q_i b(r_i) \leq B. \tag{2}$$

a) Computational Hardness: It might not be surprising that Problem 1 is intractable since it contains MRTA as a subproblem. Nonetheless, we briefly study the hardness of the problem in more detail to highlight two important details.

We begin with stating the decision version of the problem: *Given the inputs of Problem 1, does there exist a fleet such that the reward jointly collected by the optimal tours for each robot is larger or equal to some given constant?* We make two observations: First, in order to be a member of NP, a certificate for Problem 1 consists of not only a fleet but also a set of tours for the robots. Otherwise, the correctness of a solution could not be verified without solving the underlying vehicle routing problem. Second, the problem is also NP-hard when the vehicle routing part is trivial. To show this, we provide a reduction from the $0/1$ knapsack problem.

Lemma 1 (NP-hardness)**.** The decision version of Problem 1 is NP-hard.

Proof. We prove hardness via reduction from a $0/1$ knapsack problem [29]. Given an instance of Knapsack consisting of a set of n items, their weights and profits and a budget (all integer values), we construct a graph with a central depot and exactly one vertex for each item. For each vertex, we create several identical tasks with a deadline ≥ 1 and a requirement unique to the item. The number of task copies for each item equals the profit of the item. Further, let there be n robot types $r_1, \dots r_n$ with disjoint capabilities, each able to service exactly one task. The robot costs $b(r_i)$ equal the weight of the i-th item. Finally, all robot travel times equal 1, and the battery life $\beta = 2$. Thus, each robot can visit exactly one vertex, service all task copies located there, and return to the depot. Finally, the budget B equals the budget of the knapsack. A solution to the fleet design problem then contains a set of of tours for robots. This can easily be converted into a solution to the Knapsack problem: Looping over the tours Q, we identify the tasks being serviced, which directly correspond to items for the knapsack problem. □

We observe that vehicle routing is trivial in the instance created in the reduction: Each robot can only visit exactly one vertex and then returns to the depot. This highlights that the fleet design problem is computationally intractable, regardless of the hardness of the underlying MRTA problem.

b) MILP Formulation: We present an MILP formulation, adapting notation from MILPs for the TOP with time windows [30]. Given the inputs of the problem we can construct a complete graph $\bar{G}$ where vertices correspond to all N task locations and the central depot. For each robot type, the distance matrix $\bar{\boldsymbol{D}}$ then describes the shortest distances on the original graph G. First, we create a copy of the depot vertex where each robot's tour will end such that the graph has vertices $0, 1, \dots, N+1$. Indices i, j refer to vertices, index k identifies a robot ranging from 1 to K where $K = |\mathcal{F}|$. We use three binary decision variables. i) $x_{ij}^k = 1$ indicates that robot k traverses edge (i, j), ii) $y_i^k = 1$ indicates robot k service task i, and iii) $z^k = 1$ indicates if robot k is used. We use two auxiliary variables: s_i^k denotes the time robot k visits vertex i and c_i^k is a binary variable indicating if robot k can service the task i. Lastly, let M be some large constant. The MILP is then given by:

$$\max \sum_{i=1}^{N} \sum_{k=1}^{K} y_i^k \tag{3a}$$

$$s.t. \sum_{j=1}^{N} x_{0j}^k = \sum_{j=1}^{N} x_{j,N+1}^k = z^k \tag{3b}$$

$$\sum_{i=0, i\neq l}^{N+1} x_{il}^k = \sum_{j=0, j\neq l}^{N+1} x_{lj}^k = y_l^k \tag{3c}$$

$$s_i^k + \bar{D}_{ij}^k - s_j^k \leq M(1 - x_{ij}^k) \tag{3d}$$

$$\sum_{k=1}^{K} y_i^k \leq 1 \tag{3e}$$

$$y_i^k \leq c_i^k \tag{3f}$$

$$s_i^k \leq y_i^k \cdot t_i^d \tag{3g}$$

$$\sum_{i=0}^{N} +1 \left(\sum_{j=0}^{N} +1 x_{ij}^k \bar{D}_{ij}^k \right) \leq \beta^k \tag{3h}$$

$$\sum_{k=1}^{K} z^k b^k \leq B \tag{3i}$$

$$s_i^k \geq 0 \tag{3j}$$

$$x_{ij}^k, y_i^k \in \{0, 1\} \; \forall i, j = 0, \dots, N+1; \; k = 1, \dots, K. \tag{3k}$$

The objective (3a) counts the number of serviced tasks. (3b) ensures that only activated robots leave the depot. (3c) and (3d) establish connectivity and record the time a robot visits a task. (3e-g) ensure that only one robot services each task, a robot has the capabilities to service a task, and the task is serviced before the deadline. (3h) and (3i) enforce the battery life and budget constraints. Finally, (3j) and (3k) ensure positive arrival times and initialize the variables.

B. Greedy Algorithm

To the best of our knowledge there are no state-of-the-art methods for Problem 1. Thus, we introduce a greedy algorithm to establish a baseline. Beginning with an empty fleet $F = \emptyset$, `Greedy` iteratively adds the robot r^* that solves:

$$r^* = \arg\max_{r_i \in R} \frac{\rho(Q_\pi(F \cup \{r_i\}), \mathcal{T}) - \rho(Q_\pi(F), \mathcal{T})}{b_i}. \tag{4}$$

The process is repeated until the budget is exhausted. However, MRTA itself is NP-hard and thus the greedy step cannot be solved within polynomial time. Moreover, approximation algorithms for MRTA are often not available under complex constraints such as task deadlines. Thus, `Greedy` does not have an approximation guarantee.

Algorithm 1: Fleet LNS

Input: Graph G, depot s, tasks $\mathcal{T}$, robot classes R, budget B, integer K.
Output: Robot fleet F and tours Q

1 Initialize tours $Q = \{\emptyset, \emptyset, \dots\}$ for all $r_i \in \mathcal{F}$
2 $Q \leftarrow$ Randomly generate feasible tours
3 $Q^{\text{best}} \leftarrow Q$
4 **for** $k = 1$ *to* K **do**
5 $\quad$ `mode` $\leftarrow$ `Select_Removal_Mode` // tasks or robots
6 $\quad$ $Q', \mathcal{T}' \leftarrow$ `Removal`$(Q, \texttt{mode})$
7 $\quad$ $Q^{\text{new}} \leftarrow$ `Repair`$(Q', G, \mathcal{T}', B)$
8 $\quad$ **if** $\rho(Q^{new}, \mathcal{T}) > \rho(Q, \mathcal{T})$ **then**
9 $\quad\quad$ $Q^{\text{best}} \leftarrow Q^{\text{new}}$
10 $\quad\quad$ $Q \leftarrow Q^{\text{new}}$
11 $\quad$ **if** `Accept`(Q^{new}, k) **then**
12 $\quad\quad$ $Q \leftarrow Q^{\text{new}}$
13 $F \leftarrow$ robots with non-empty tours in Q^{best}
14 **return** F, Q^{best}

Algorithm 2: `Repair` (Insertion heuristic)

Input: Graph G, current tours Q, tasks $\mathcal{T}$, unassigned tasks $\mathcal{T}'$, budegt B.
Output: New tours Q'

1 $Q' \leftarrow Q$
2 **while** $\mathcal{T}'$ *is not empty* **do**
3 $\quad$ $T \leftarrow$ `pop_random_task`$(\mathcal{T}', Q')$
4 $\quad$ **for** r_i *in* $\mathcal{F}$ **do**
5 $\quad\quad$ $\tau_i' \leftarrow$ best insertion of T in tour τ_i
6 $\quad\quad$ $Q^i \leftarrow Q' \cup \tau_i' \setminus \tau_i$
7 $\quad\quad$ Compute utility z^i
8 $\quad$ **if** $\max_i \ z^i > 0$ **then**
9 $\quad\quad$ $Q' \leftarrow$ feasible Q^i with largest z^i
10 **return** Q'

C. Fleet Optimization via Large Neighbourhood Search

We now present our large neighbourhood search (LNS) approach to Problem 1. Beginning with an arbitrary initial solution, LNS algorithms repeatedly remove parts of the solution, *e.g.,* vertices from a TSP tour, and then reinsert these elements. This allows LNS approaches to solve large instances of complex problems.

a) Algorithm Overview: We propose an integrated LNS algorithm that simultaneously optimizes which robots are part of the fleet and what route each robot takes. Thus, our method uses two removal heuristics, allowing it to randomly switch between a) changing which robots are currently part of the fleet and b) improving the current tours. The main procedure is summarized in Algorithm 1. We begin with a randomly generated feasible solution Q. Over K iterations, we select a mode for the removal heuristic and then remove a subset of all tasks from the tours Q (line 6). We then re-enter removed tasks $\mathcal{T}'$ into the subtours Q' (line 7), and update the current solution Q and best solution found thus far Q^{best}(lines 8-9). Further, we use simulated annealing and potentially accept a suboptimal new solution to allow for more exploration in early iterations of the algorithm (lines 11-12). Next, we will present the two proposed removal heuristics and the insertion heuristic in detail.

b) Removal Heuristics: We propose two removal heuristics: `Robot-removal` and `Task-removal` to enable replacing robots in the fleet and improving current tours.

`Robot-removal` selects a random subset of all robots and deletes their tours entirely, effectively removing these robots from the fleet. A tuning parameter n^{R} defines an upper limit for the size of the subset of robots that are removed. This allows for replacing parts of the fleet with different robots in the subsequent `Repair` step.

`Task-removal` removes a random subset of the tasks of each current tour. Similar to the first heuristic, a parameter n^{T} describes the maximum percent of tasks that can be deleted. The subsequent `Repair` step then may improve the tours without changing which robots are *active* (*i.e.,* in the fleet). Finally, the function `Select_Removal_Mode` randomly chooses one of the heuristics following some bias $p^{\texttt{removal}}$.

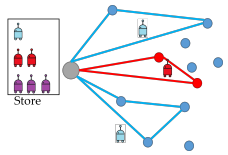
c) Insertion Heuristic: We now present our `Repair` heuristic. Consider a robot r_i with current tour τ_i, and some task T. A maximum reward insertion then finds position in the tour τ_i such that the reward of adding T at that position is maximized. We notice that this is independent of the other tours. Let Q be the current set of tours, and let Q^i denote the set of tours when T is added to robot r_i. One could then assign T to the robot r_i where the marginal gain $\rho(Q^i, \mathcal{T}) - \rho(Q, \mathcal{T})$ is largest. However, such a strategy does not consider the cost of adding a new robot to the fleet. Thus, we construct a utility function. First, we check if Q^i is feasible with respect to the budget B (equation (2)) and each robots battery-life constraint. If Q^i is infeasible, the utility is zero. If the insertion is feasible, we compose the utility using the marginal gain, a discount factor δ^i and a noise term η:

$$z^i = (1 + \eta)\ \delta^i\ (\rho(Q^i, \mathcal{T}) - \rho(Q, \mathcal{T})). \tag{5}$$

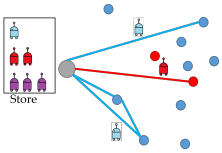
The discount factor penalizes adding T to a robot which currently has an empty tour. Thus, $\delta^i = 1$ if τ_i is not empty. Otherwise, δ^i is drawn randomly from $\{1, {}^1\!/_{\beta(r_i)}\}$ with some probability $p^{\texttt{discount}}$, *i.e.,* randomly discounts the marginal gain with the robots deployment cost. Lastly, $\eta =\in [0, .1]$ is a small uniformly random noise term, increasing exploration.

Given currently unassigned tasks $\mathcal{T}'$, we select a task randomly (line 3) and then find the maximum reward insertion into the current tour τ_i for every robot r_i in the base fleet $\mathcal{F}$ (lines 4-6). We compute a utility z^i for the insertion Q^i (lines 7) and update the new set of tours Q' (lines 8-9). This is repeated until the set of unassigned tasks $\mathcal{T}'$ is empty.

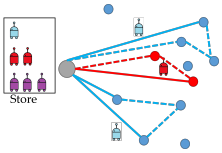
d) Example Illustration: We provide an example of the LNS algorithm in Figure 2. Here the budget constraint allows for using at most three robots of any type. An initial solution deploys two blue and one red robot, allowing the fleet to service 7 of 11 tasks. In iteration 1, a `Task-removal`



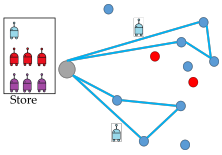
(a) Initial solution.



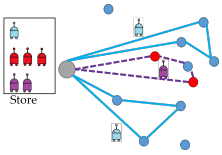
(b) Iteration 1: `Task-removal`.



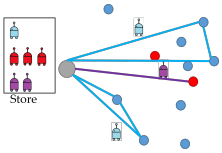
(c) Iteration 1: `Repair`.



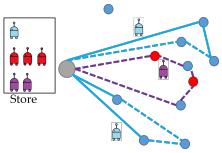
(d) Iteration 2: `Robot-removal`.



(e) Iteration 2: `Repair`.



(f) Iteration 3: `Task-removal`.



(g) Iteration 3: `Repair`.

Fig. 2: Illustration of Algorithm 1 over three iterations. The grey dot is the depot, coloured dots are task locations. Blue and red robots can only service tasks of their respective colour, purple robots can service any task. Solid lines indicate current robot tours, dashed lines show edges added during `Repair`. Robots in the 'store' are currently not in the fleet and thus do not count towards the budget.

enables the upper blue robot to change its tour, increasing the number of serviced tasks to 8 (subfigure b and c). This is the best attainable solution for the current fleet. In the second iteration, the algorithm performs a `Robot-removal` and subsequently replaces the red robot with a purple one, which is able to service yet another additional tasks (subfigure d and e). Finally, by performing `Task-removal` again, the purple robot takes over a task from the lower blue robot, allowing the latter to service another previously neglected task. Thus, in the final solution 10 out of 11 tasks are being serviced.

IV. Simulations

We evaluate our work in a series of simulation experiments, comparing against two baselines.

A. Experiment Setup

a) Scenarios: We consider three scenarios: *Experiment 1* has a simple setup with only three ground robots and two task types. In *Experiment 2* tasks have no deadlines, features more task requirements and offers a larger variety of robots. In *Experiment 3* we consider ground robots and aerial robots that jointly have to take different measurements, yet not all measurements can be taken by either robot type. Further,

ID	capabilities	speed	battery	cost	type
1	$\{1\}$	100%	200	20	AGV
2	$\{2\}$	100%	200	20	AGV
3	$\{1,2\}$	150%	500	25	AGV

(a) Experiment 1.

ID	capabilities	speed	battery	cost	type
1	$\{1\}$	100%	300	20	AGV
2	$\{2\}$	100%	300	20	AGV
3	$\{3\}$	100%	300	20	AGV
4	$\{1,2,3\}$	150%	300	30	AGV
5	$\{1,2\}$	100%	250	25	AGV

(b) Experiment 2.

ID	capabilities	speed	battery	cost	type
1	$\{1\}$	100%	300	20	AGV
2	$\{1,2\}$	150%	300	25	AGV
3	$\{1,3\}$	200%	300	20	UAV
4	$\{3\}$	200%	250	10	UAV
5	$\{1\}$	300%	250	15	UAV

(c) Experiment 3.

TABLE I: Different types of robots used in the Experiments.

ground robots have to avoid obstacles while aerial robots can travel directly between task locations.

b) Baselines: We consider two baselines: Selecting robots randomly until the budget is exhausted (`Random`), and the greedy approach (`Greedy`) described in Section III-B. To solve the underlying MRTA problem for computing (4) and the final set of tours, we use another large neighbourhood search that optimizes the tours for a fixed fleet. Using the MILP as a baseline is impractical due to very high runtime for even small instances. Our approach is labelled as `LNS`.

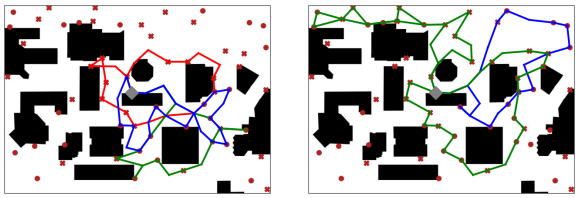
c) Algorithm parameter: The algorithm parameters are set up as follows: Iteration budget $K = 1000$, bias for random selection of removal heuristic $p^{\texttt{removal}} = 1/3$, max. % of robots removed by `Robot-removal` $n^{\texttt{R}} = 25$, max. % of tasks removed by `Task-removal` $n^{\texttt{T}} = 50$, and bias to neglect deployment cost in `Repair` $p^{\texttt{discount}} = 1/10$.

d) Environment: We use a schematic real-world campus map, shown in Figure 3. Given uniformly sampled task locations, we create a complete meta-graph, where vertices correspond only to the depot and task locations, and edge lengths are given by the shortest paths on a PRM. All task deadlines are $t^d = 150$. We repeat experiments for 20 trials.

B. Results

a) Experiment 1: In the first experiment, we consider two different task types and only three robot types with varying capabilities and cost, summarized in Table Ia.

Figure 3 illustrates example solutions for `Greedy` and `LNS` for $N = 60$ tasks and a budget of $B = 70$. `Greedy` iteratively adds one robot with ID 1, 2, and 3, resulting in a fleet that services 27 tasks before their deadlines. The `LNS` approach assembles a fleet with one robot of type 2 and two robots of type 3, allowing the fleet to service 38 tasks. The example highlights the main shortcoming of a greedy approach: In early iterations, low budget robots have a large marginal gain and are thus selected. Yet, in later iterations the remaining



(a) Greedy solution. (b) LNS solution.

Fig. 3: Example for Experiment 1. Different colors indicate different robot types from Table Ia (ID 1 – red, ID 2 – blue, ID 3 – green)

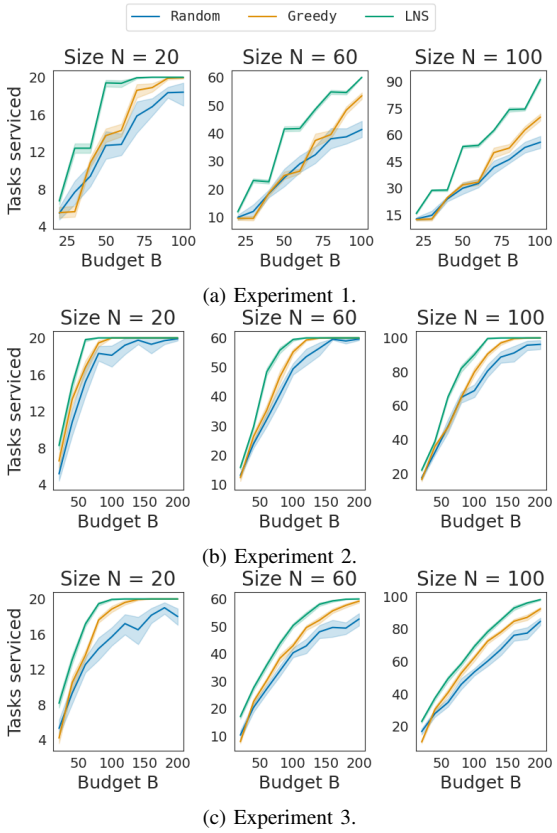
budget does not suffice to add multiple flexible robots that could combine different task types. In contrast, LNS employs multiple of the more expensive but also much more capable type 3 robots, leading to more tasks being serviced.

We repeat the experiment for varying budgets and numbers of tasks, and quantitative results are shown in Figure 4a. With increasing budgets, all methods are able to service more tasks. However, while Greedy performs only slightly better than Random, the proposed method services significantly more tasks under almost all settings. In particular, LNS only requires roughly half of the budget ($B = 50$) to achieve the same performance as the baselines with the full budget ($B = 100$). For $N = 20$, LNS is able to service all tasks as the budget increases. Overall, the relative performance of the baselines becomes comparably poorer for larger N even for the full budget $B = 100$: For $N = 20$ Greedy and LNS service 100% of tasks. For $N = 60$, LNS still achieves 100%, yet Greedy falls to 83% and for $N = 100$ they achieve 90% and 70%, respectively. In summary, in the simple setup with only few robot types the proposed method is able to find substantially stronger solutions than a greedy approach.

b) Experiment 2: Experiment 2 extends the setup to three task requirements and five available robots (see Table Ib), and removes task deadlines. Figure 4b shows a similar trend as in the first experiment, yet the larger budgets allow all for more tasks being completed. LNS requires a substantially smaller budget to match the best performance of Greedy: To achieve $\approx 100\%$ serviced tasks under different values for $N = 20, 60, 100$, LNS needs a budget of $B = 80$, $B = 120$, and 120, while Greedy requires $B = 100$, $B = 140$, and 180, respectively.

c) Experiment 3: The third experiment considers a data collection mission where ground vehicles and drones take measurements. There are four different task types, of which some are exclusive to different robot types, as listed in Table Ic. Similar to the other experiments, Figure 4c shows that LNS consistently outperforms Greedy. For the different values of N, Greedy achieves its highest performance for $B = 160$, $B = 200$ and $B = 200$, respectively. LNS requires only budgets of $B = 100$, $B = 160$ and $B = 160$, respectively, to achieve the same number of serviced tasks. As in Experiment 2, the performance gap is largest for relatively small budgets. This suggests that the performance of Greedy suffers from suboptimal choices in early iterations. The LNS approach is able to avoid these local optima and thus find better solutions.

d) Runtime: Overall, Greedy and LNS perform comparably for small problems ($N = 20$), running on average



(a) Experiment 1.

(b) Experiment 2.

(c) Experiment 3.

Fig. 4: Experimental Results: # serviced tasks for different budgets.

within $15s$ per instance. In Experiment 1, Greedy performs better for large instances ($N = 100$), resulting in $150s$ compared to $200s$ for LNS. However, Experiment 2 and 3 feature more robots, causing the runtime of Greedy to increase drastically to $> 500s$ for the larger instances. In contrast, LNS maintains an average runtime of $200s$. Moreover, using only 100 iterations for LNS reduces its runtime by factor 10, yet LNS still substantially outperforms Greedy in Experiments 1 and 2, and by a small margin in Experiment 3.

V. Discussion and Future Work

We studied the problem of designing heterogeneous robot fleets for multi-robot task assignment. We provided a MILP formulation and presented an LNS algorithm. In simulation experiments, we demonstrated that the LNS approach consistently finds better solutions than a greedy approach, *i.e.,* requires smaller budgets to service the same number of tasks, while its runtime scales better to large instances.

A limitation of our work is the simple cost model for robot deployment. This could be extended to operation costs that increases with the robot's continued deployment. Further, our experiment relied on synthetic data. Using real-world robot models and realistic travel capabilities would highlight the practical benefits of the proposed method. Lastly, we considered tasks that are serviced independently, *i.e.,* one robot's mission does not affect how another robot can service its tasks. Yet, in some applications such as search-and-rescue, one robot's task completion might benefit other robots. This poses further challenges to the fleet design problem.

REFERENCES

[1] M. Chandarana, D. Hughes, M. Lewis, K. Sycara, and S. Scherer, "Planning and monitoring multi-job type swarm search and service missions," *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–14, 2021.

[2] M. J. Sousa, A. Moutinho, and M. Almeida, "Decentralized distribution of uav fleets based on fuzzy clustering for demand-driven aerial services," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020, pp. 1–8.

[3] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2011.

[4] N. Wilde and J. Alonso-Mora, "Online multi-robot task assignment with stochastic blockages," in *61st IEEE Conference on Decision and Control (CDC)*, 2022, accepted, to appear.

[5] A. Sadeghi and S. L. Smith, "Re-deployment algorithms for multiple service robots to optimize task response," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2356–2363.

[6] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

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

[8] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli, "Analysis and control of autonomous mobility-on-demand systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 633–658, 2022.

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

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

[11] G. Dixit, N. Zerbel, and K. Tumer, "Dirichlet-multinomial counterfactual rewards for heterogeneous multiagent systems," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 209–215.

[12] C. Y. H. Lee, G. Best, and G. A. Hollinger, "Stochastic assignment for deploying multiple marsupial robots," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021, pp. 75–82.

[13] J. L. Kit, A. G. Dharmawan, D. Mateo, S. Foong, G. S. Soh, R. Bouffanais, and K. L. Wood, "Decentralized multi-floor exploration by a swarm of miniature robots teaming with wall-climbing units," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 195–201.

[14] G. Shi, N. Karapetyan, A. Asghar, J. Reddinger, J. Dotterweich, J. Humann, and P. Tokekar, "Risk-aware uav-ugv rendezvous with chance-constrained markov decision process," in *IEEE Conference on Decision and Control (CDC)*, 2022.

[15] M. Wesselhöft, J. Hinckeldeyn, and J. Kreutzfeldt, "Controlling fleets of autonomous mobile robots with reinforcement learning: a brief survey," *Robotics*, vol. 11, no. 5, p. 85, 2022.

[16] M. Cecchi, M. Paiano, A. Mannucci, A. Palleschi, F. Pecora, and L. Pallottino, "Priority-based distributed coordination for heterogeneous multi-robot systems with realistic assumptions," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6131–6138, 2021.

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

[18] W. Xu, Z. Xu, J. Peng, W. Liang, T. Liu, X. Jia, and S. K. Das, "Approximation algorithms for the team orienteering problem," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1389–1398.

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

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

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

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

[23] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3853–3859.

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

[25] K. E. Booth, G. Nejat, and J. C. Beck, "A constraint programming approach to multi-robot task allocation and scheduling in retirement homes," in *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings 22*. Springer, 2016, pp. 539–555.

[26] B. Pan, Z. Zhang, and A. Lim, "Multi-trip time-dependent vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 291, no. 1, pp. 218–231, 2021.

[27] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

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

[29] H. Kellerer, U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy, and D. Pisinger, *Multidimensional knapsack problems*. Springer, 2004.

[30] S.-W. Lin and F. Y. Vincent, "Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing," *Computers & Industrial Engineering*, vol. 114, pp. 195–205, 2017.