

Heterogeneous Coalition Formation and Scheduling with Multi-Skilled Robots

Ashay Aswale and Carlo Pinciroli

Abstract—We present an approach to task scheduling in heterogeneous multi-robot systems. In our setting, the tasks to complete require diverse skills. We assume that each robot is multi-skilled, i.e., each robot offers a subset of the possible skills. This makes the formation of heterogeneous teams (*coalitions*) a requirement for task completion. We present two centralized algorithms to schedule robots across tasks and to form suitable coalitions, assuming stochastic travel times across tasks. The coalitions are dynamic, in that the robots form and disband coalitions as the schedule is executed. The first algorithm we propose guarantees optimality, but its run-time is acceptable only for small problem instances. The second algorithm we propose can tackle large problems with short run-times, and is based on a heuristic approach that typically reaches 1x-2x of the optimal solution cost.

I. INTRODUCTION

The parallelism offered by multi-robot systems is a natural fit for missions in which large numbers of tasks must be achieved as quickly as possible [1]. In realistic settings, each task requires robots with specific *skills*, such as specific sensors, actuators, or computational resources. However, as the diversity of the tasks increases and the set of required skills grows, it becomes infeasible to envision swarms of identical robots that can interchangeably perform any task. Rather, specialization and redundancy become desirable due to better scale economy and expected long-term resilience [2], [3].

The goal of this paper is to contribute to realizing this vision. In our setting, a heterogeneous swarm of multi-skilled robots must perform a set of tasks as quickly as possible. We assume that, due to the diversity of the tasks, the robot must form *coalitions*, i.e., teams of robots that, combined, offer the required skills for each task to be completed [4]–[6]. In addition to heterogeneity, our problem setting has two crucial features: (i) the diversity of the tasks also requires the coalitions to be *dynamically* formed and disbanded on a per-task basis; and (ii) all the required robots in a coalition must be present at the same time for the task to progress. These two features imply that, along with the combinatorial problem of forming coalitions, the robots must also *schedule* the optimal task agenda in a coherent manner.

The key difference between our work and existing works is that we consider *simultaneously* multi-skill coalition formation and multi-robot task scheduling for a complete coverage problem. With reference to the Korsah *et al.* taxonomy [7], this problem is an instance of cross-schedule dependencies

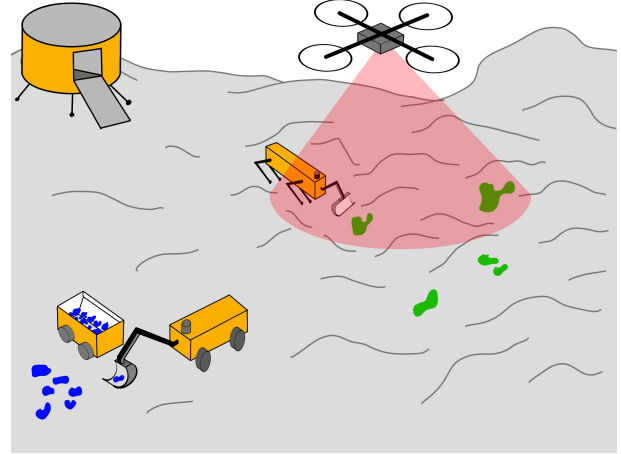


Fig. 1: A heterogeneous swarm of multi-skilled robots forming coalitions. The tasks in this diagram are excavation of blue resources (bottom left), and green material sampling (center). The excavation task requires a resource sensor, an excavating arm, and a bucket to carry resources. Material sampling requires a scanner to find and locate the sample, and an arm on a legged robot.

(XD), single-task robots (ST), multi-robot tasks (MR), and time-extended assignment (TA). In addition, to make our problem setting more realistic, we enrich our formulation with *stochastic travel times* across tasks [8].

We study two centralized algorithms to solve this problem. The first algorithm is optimal, but it scales poorly with the number of tasks, robots, and skills. In contrast, the second method scales to hundreds of tasks, robots, and skills. Even though the latter method offers no optimality guarantees, we empirically show that its performance is within a factor of 2 with respect to the optimum.

The rest of this paper is structured as follows. In Sec. 2 we survey related work on coalition formation and multi robot scheduling. In Sec. 3 we discuss the problem formulation for both of our approaches. In Sec. 4 we analyze the results of both the methods and compare them with each other. We conclude the paper in Sec. 5.

II. RELATED WORK

The problems of task scheduling and coalition formation have received wide attention in the literature. We identified several axes to categorize relevant work in Table I.

Several works study coalition formation without scheduling. Rahwan *et al.* [9] and Guo *et al.* [10] focus on *homogeneous* coalition formation. Rahwan *et al.* [9] considers

All the authors are with the Dept. of Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA, USA (email: {asaswale, cpinciroli}@wpi.edu).

static coalitions in which coalitions, once formed, are kept constant throughout the experiment. Guo *et al.* [10] compare *static* and *dynamic* coalition formation, in which the robots are allowed to change coalition from task to task.

Recently, several researchers considered coalition formation of heterogeneously skilled robots [11]–[13]. However, they focus on coverage and connectivity problems, neglecting scheduling aspects. Similarly, Barton *et al.* [4] study coalition formation in a network of *heterogeneously skilled* agents. At the opposite end of the spectrum, Parker *et al.* [14] and Visser *et al.* [15] study search-and-rescue scenarios with a heterogeneous set of robots and focus on scheduling them without coalition formation.

Significant effort has been devoted to combining coalition formation with task scheduling [5], [6], [16]–[18]. However, all of these works assume the robots to be homogeneously skilled.

Some of the closest research to this paper studies coalition formation with heterogeneous robots that also produces an optimal plan with cross-scheduling dependencies [19]–[22]. However, these works do not consider *multi-skilled* robots; rather, each robot is a specialist of a specific skill.

Prorok *et al.* [23], [24] and Kosak *et al.* [25] address multi-skilled robots. In the work of Prorok *et al.*, [23], [24], the robots offer a subset of the possible skills, whereas Kosak *et al.* [25] allow “multipotent” robots to modify themselves and adapt to the task at hand. However, all these works focus on coalition formation without a scheduling component.

Amador *et al.* [26], [27] addressed a comparable issue, the ‘Law enforcement problem’ (*LEP*), which assigns police officers to tasks with unknown locations, arrival times, and importance levels. Although *LEP* considers cross-scheduling dependencies for multi-skilled agents, their problem statement differs significantly from ours. Their approach concerns time-sensitive tasks with various importance levels and agents who can abandon or ignore tasks if they do not offer a higher utility. In contrast, our problem assumes all tasks are equally important and not time-sensitive. Moreover, our agents cannot abandon any tasks, and all tasks must be completed.

We are also interested in the presence of stochastic aspects in the problem statement. The works considered so far lack such a component, but recent research has started to include it. Nam *et al.* [8], for example, include stochastic travel times in multi-agent task scheduling; however, their work does not require coalitions formation. In the literature on coalitions, stochastic aspects concern resilience and reconfiguration [2], [28], [29], without a scheduling component.

III. APPROACH

A. Preliminaries

We consider an environment in which a set of m tasks is scattered. Each task requires a specific set of skills. A task can require any number of skills. In total, across all tasks, l skills are required; some might be in higher demand than others. The mapping between tasks and required skills

TABLE I: Comparison of the related work to the present work. Legend: Coalt: Coalition, Heter: Heterogeneous set of robots, Sched: Scheduling, Stoch: Stochasticity of any nature, M-Skill: Multi-skilled robots.

Work	Coalt	Heter	Sched	Stoch	M-Skill
[9], [10]	✓				
[4], [11]–[13]	✓	✓			
[5], [6], [16]–[18]	✓		✓		
[14], [15]		✓	✓		
[19]–[22]	✓	✓	✓		
[2], [28], [29]	✓	✓		✓	
[8]			✓	✓	
[23]–[25]	✓	✓			✓
[26], [27]	✓	✓	✓		✓
<i>this work</i>	✓	✓	✓	✓	✓

is captured by the binary matrix R , whose element r_{js} is 1 if task j requires skill s and 0 otherwise.

A swarm of n robots is deployed in the environment and must perform the tasks as quickly as possible. The robots may start from the same ‘depot’ or be scattered throughout the environment. We assume that each robot offers a subset of the possible skills, and the robots combined offer all the required skills for the tasks to be completed. We restrict each robot to have a maximum of $l/2$ skills. The binary matrix Q encodes the mapping between robots and skills. An element q_{is} is 1 if robot i possesses skill s and 0 otherwise.

B. Optimal Formulation

We first discuss a Linear Program that produces the optimal solution to the problem. We use a binary assignment tensor (X) for the formulation. In this tensor, if an element x_{ijk} is 1, then robot i attends task k right after task j ; otherwise the element is 0. We refer to the robot’s location at the start and the end of the experiment as “task 0” and “task $m + 1$ ” respectively.

We denote with T the cost tensor that stores the travel time from one task to another. Because the robots might start from different locations, the travel cost from task 0 to the other tasks differs from robot to robot. For this reason, t_{ijk} denotes the travel time for robot i to navigate from task j to task k . Travel times at this stage are deterministic; we will explain how to handle stochastic times in Sec. III-B.4.

1) *Valid schedule generation*: We now discuss the constraints necessary to generate valid schedules for the robots. Every robot i must start from task 0 (initial location) exactly once (Eq. (1)) and finish its schedule at task $(m + 1)$ (final location) exactly once (Eq. (2)). Therefore, task 0 is exit-only (Eq. (3)) and task $(m + 1)$ is entry-only (Eq. (4)).

TABLE II: Table of parameters and variables

Symbol	Description
l	Total number of skills in a configuration
m	Total number of tasks in a configuration
n	Total number of robots in a configuration
Q	Binary matrix for mapping between robots and skills
R	Binary matrix for mapping between tasks and requirements
T	Cost tensor
T^s	Matrix for the stochastic buffer time for travel
T^e	Vector for execution time of the tasks
X	Binary assignment tensor
Y	Matrix for arrival times of the robots at tasks
Y^{\max}	Vector for the tasks' execution start times
Z	Matrix for how many robots offer each skill
Z^b	Matrix to denote excess skills at the tasks

$$\forall i \sum_{k=1}^{m+1} x_{i0k} = 1 \quad (1)$$

$$\forall i \sum_{j=0}^m x_{ij(m+1)} = 1 \quad (2)$$

$$\forall i \sum_{j=1}^{m+1} x_{ij0} = 0 \quad (3)$$

$$\forall i \sum_{k=0}^m x_{i(m+1)k} = 0 \quad (4)$$

For what concerns the other tasks $[1, m]$, we impose that a task cannot appear twice in a robot's schedule. To this effect, a robot can enter a task k at most once (Eq. (5)) and exit it at most once (Eq. (6)). In addition, if a robot has visited a task j , it must leave it, and it cannot leave it without first visiting it (Eq. (7)). Finally, robots cannot dwell at a task after visiting it (Eq. (8)).

$$\forall i \forall k \setminus \{0, m+1\} \sum_{j=0}^m x_{ijk} \leq 1 \quad (5)$$

$$\forall i \forall j \setminus \{0, m+1\} \sum_{k=1}^{m+1} x_{ijk} \leq 1 \quad (6)$$

$$\forall i \forall j \setminus \{0, m+1\} \left(\sum_{k=0}^m x_{ikj} = \sum_{k=1}^{m+1} x_{ijk} \right) \quad (7)$$

$$\forall i \forall j \quad x_{ijj} = 0 \quad (8)$$

These constraints could result in schedules where a robot travels along multiple different paths at the same time. There might be one valid path, starting from task 0 and ending at task $m+1$, and an invalid path, looping between three or more tasks. To solve this, we add "lazy constraints" that reject a candidate solution if it contains such loops. The algorithm to detect loops is reported in Alg. 1. Intuitively, the algorithm checks the number of tasks covered in the valid path from 0 to $m+1$. It then compares this number with the total number of tasks covered in the whole schedule. Dissimilarity in these two numbers indicates the existence of an invalid path in the schedule.

Algorithm 1 Detecting loops in a candidate solution

```

for each robot  $i$  do
   $next \leftarrow 0$ 
   $count \leftarrow 0$ 
  while  $next$  is not  $m+1$  do
     $next \leftarrow \arg \max_k (x_{i,next,k})$ 
     $count \leftarrow count + 1$ 
  end while
   $visited \leftarrow \sum_{j=0}^{m+1} \sum_{k=0}^{m+1} x_{ijk}$ 
  if  $count \neq visited$  then
    return solution is not valid
  end if
end for
return solution is valid

```

2) *Skill allocation*: To satisfy the skills required by the tasks, a robot i must possess at least one of the required skills to attend a task $k \in [1, m]$.

$$\forall i \forall k \setminus \{0, m+1\} \sum_{j=0}^m x_{ijk} \leq \sum_{s=0}^l q_{is} r_{ks}. \quad (9)$$

For a schedule to be valid, each task must have robots with the required skills. To achieve this, we introduce matrix Z , where z_{ks} indicates the number of robots that offer skill s required for task k (Eq. (10)). We ensure that each element in Z is greater than or equal to the corresponding element in the skill requirement matrix R (Eq. (11)).

$$\forall s \forall k \setminus \{0, m+1\} \quad z_{ks} = \sum_{i=0}^n \sum_{j=0}^m x_{ijk} q_{is} \quad (10)$$

$$\forall s \forall k \setminus \{0, m+1\} \quad z_{ks} \geq r_{ks}. \quad (11)$$

The above constraints theoretically allow for a task to have more skills than required. In general this is unavoidable, because the robots contributing to a task might have overlapping skills while also contributing unique ones. However, there is a benefit in avoiding schedules where certain tasks are attended by superfluous robots, i.e., robots that have some of the required skills, but none of them is unique within the coalition. The benefit is that rejecting superfluous robots makes the search space much smaller, significantly reducing run-times as we empirically observed in the experiments we ran during early phases of this work.

To identify superfluous robots, we use the binary matrix Z^b (Eq. (12)) where z_{ks}^b equals 1 if skill s is excessive for task k , and 0 otherwise.

$$\forall s \forall k \setminus \{0, m+1\} \quad z_{ks}^b = \begin{cases} 0 & \text{if } z_{ks} \leq r_{ks} \\ 1 & \text{otherwise.} \end{cases} \quad (12)$$

We then impose that, if a robot i attends a task k , the robot must have at least one skill that is not in excess. In Eq. (13) $z_{ks}^b q_{is}$ is 1 when a skill s of robot i is in excess for task k , and $\sum_s z_{ks}^b q_{is}$ is the number of redundant skills robot i has for task k . Due to constraint (9), if robot i attends task k then

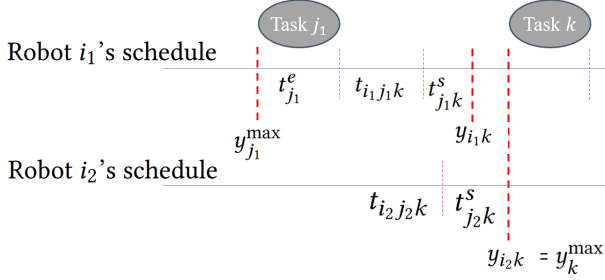


Fig. 2: Arrival times of robots i_1 and i_2 . Robot i_1 attends task k after j_1 , whereas the robot i_2 attends the same task after j_2 . The duration of execution, travel, and stochastic buffer time are denoted by t^e , t , and t^s , respectively. Task k 's starting time is given by y_k^{\max} . Robot i_2 is the last to arrive at the task, so y_k^{\max} is equal to its time of arrival (y_{i_2k}).

$\sum_s q_{is} r_{ks} \geq 1$, i.e., at least one of robot i 's skills is required by task k . We can then impose the following constraint:

$$\forall i \forall k \setminus \{0, m+1\} \sum_{j=0}^m x_{ijk} = 1 \implies \sum_{s=0}^l z_{ks}^b q_{is} \leq \left(\sum_{s=0}^l q_{is} r_{ks} \right) - 1 \quad (13)$$

3) *Arrival times*: One of the core requirements in a coalition is the simultaneous presence of all its members. In this paper, we assume that the absence of even a single robot makes it impossible for a task to progress. Hence, a task can start when the last required robot has joined the coalition at the location.

To express these requirements, we consider the arrival times of each robot. We introduce matrix Y whose elements y_{ik} store the arrival time of robot i at task k . If a robot does not visit a task, its corresponding arrival time is set to 0:

$$\forall i, k \setminus \{0\} \sum_{j=0}^m x_{ijk} = 0 \implies y_{ik} = 0 \quad (14)$$

Task j starts at the arrival time of the last robot to join the coalition, denoted by y_j^{\max} .

$$\forall j \setminus \{0\} y_j^{\max} = \max_i (y_{ij}) \quad (15)$$

To calculate the arrival time y_{ik} of robot i at a task k , it is sufficient to sum the time of completion of the previous task j with the travel time from task j to task k (denoted by t_{ijk}). The constraint is then

$$\forall i \forall j \setminus \{m+1\} \forall k \setminus \{0\} x_{ijk} = 1 \implies y_{ik} = y_j^{\max} + t_j^e + t_{ijk} + t_{jk}^s \quad (16)$$

where t_j^e is the execution time of task j and y_j^{\max} indicates the starting time of the same task. The stochastic buffer time between the task j and k is given by t_{jk}^s which will be covered in Sec. III-B.4. A pictorial representation of this calculation is reported in Fig. 2.

4) *Stochastic travel times*: In the quest for a problem formulation that incorporates as many realistic aspects as possible, we include the possibility for travel times to be known only probabilistically. To model travel times as stochastic processes, we assume that the delay can be captured as Gaussian noise $\mathcal{G}(\mu, \sigma)$. More specifically, if we denote with t the ideal travel time between two tasks, then

$$t^* = t + \mathcal{G}(\mu, \sigma). \quad (17)$$

For each robot in a coalition, we can express the need to arrive at the task as

$$P(t < \bar{t}) \geq \epsilon \quad (18)$$

where \bar{t} is the hypothetical starting time of the task. We can develop Eq. (18) as follows:

$$\begin{aligned} t^* &= t + \mathcal{G}(\mu, \sigma) \leq \bar{t} \\ \mathcal{G}(\mu, \sigma) &\leq \bar{t} - t \\ \sigma^2 \mathcal{G}(0, 1) + \mu &\leq \bar{t} - t \\ P(\sigma^2 \mathcal{G}(0, 1) + \mu &\leq \bar{t} - t) \geq \epsilon \\ P\left(\mathcal{G}(0, 1) \leq \frac{\bar{t} - t - \mu}{\sigma^2}\right) &\geq \epsilon \\ \Phi\left(\frac{\bar{t} - t - \mu}{\sigma^2}\right) &\geq \epsilon \end{aligned}$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of $\mathcal{G}(0, 1)$. Therefore, indicating with $\Phi^{\text{inv}}(\cdot)$ the inverse of $\Phi(\cdot)$, we can write

$$\begin{aligned} \left(\frac{\bar{t} - t - \mu}{\sigma^2}\right) &\geq \Phi^{\text{inv}}(\epsilon) \\ \bar{t} - t &\geq \mu + \sigma^2 \Phi^{\text{inv}}(\epsilon) \end{aligned}$$

This calculation allows us to introduce the symbol t^s defined as follows:

$$t^s = \mu + \sigma^2 \Phi^{\text{inv}}(\epsilon) \quad (19)$$

which indicates a “safety margin” to arrive on time at a task with probability ϵ given the mean and standard deviation μ, σ of the road to that task.

5) *Objective*: The cost function we aim to minimize is the total time taken by the robots to complete the tasks. This corresponds to the arrival time of the last robot at task $(m+1)$. The objective is therefore

$$\min y_{m+1}^{\max} \quad (20)$$

6) *Solving*: We use Gurobi [30] to solve the optimization problem. This software is well-known to efficiently produce optimal solutions for convex problems. However, our problem is non-convex, and the objective function (Eq. (20)) hints that multiple equally good solutions will exist.

C. Greedy formulation

The previously discussed method produces an optimal result, but experimental evaluation reveals that it takes a long time to reach a solution. This motivates the need for another method that can solve the same problem quickly, although at the cost of optimality. We propose a simple, but effective greedy solver that produces a quick but sub-optimal result.

Algorithm 2 The proposed greedy algorithm

```

1: while Any task is unsatisfied do
2:    $(i_1, k_1), \dots \leftarrow$  Robot-task pairs with max contribution
3:    $(i^c, k^c) \leftarrow$  The earliest robot-task pair from  $(i_1, k_1), \dots$ 
4:   Assign task  $k^c$  to the robot  $i^c$  using Algorithm 3
5:   while Unaddressed skill at task  $k^c$  do
6:      $i_1^d, \dots \leftarrow$  Robots with max contributions from remaining
       skills at  $k^c$ 
7:      $i^d \leftarrow$  The earliest robot from  $i_1^d, \dots$ 
8:     Assign task  $k^c$  to the robot  $i^d$  using Algorithm 3
9:   end while
10:   $y_{k^c}^{\max} \leftarrow \max_i y_{ik^c}$ 
11: end while

```

Algorithm 3 Assign task k to robot i

```

1:  $j \leftarrow$  The current task of robot  $i$ 
2:  $x_{ijk} = 1$ 
3:  $y_{i,k} \leftarrow y_j^{\max} + t_j^e + t_{ijk} + t_{jk}^s$ 
4: Update the list of unaddressed skills at task  $k$ 

```

1) *Methodology*: In this work, we assume that task execution can only start when all the required skills are fulfilled simultaneously. Thus, a coalition might cause its robots to wait idly until the last robot in the coalition arrives. It is thus desirable to have as small coalitions as possible with robots that cover as many skills as possible. On the other hand, only seeking a solution with small coalitions might require few, powerful robots to spend significant time travelling across the environment to attend the assigned tasks. In such a scenario, the generated paths for the robots are not optimal due to the absence of any mechanism to shorten the robots' travel path. Motivated by these observations, we propose a greedy algorithm that promotes forming small coalitions while also minimizing the distance traveled by the robots.

Our algorithm first finds the robots that can contribute the most to a task and arrive the soonest. We define a robot's 'contribution' as the number of previously unoffered skills it can bring to a task. We identify all the robot-task pairs that maximize the robots' contributions to the tasks (Alg. 2, line 2). If multiple robots contribute equally, we choose the one that can reach the task location first (Alg. 2, line 3). This estimated time of arrival is calculated with the same logic as in Sec. III-B.3.

We use Alg. 3 to add the task to the robot's schedule (line 2), update its arrival time (line 3), and update the task's requirements (line 4) to account for the skills provided by the attending robot.

We now choose a robot coalition to fulfill the skills required for task k^c . If the task still requires additional skills

to start (Alg. 2, line 5) we select the robots that can offer the highest number of the remaining skills (Alg. 2, line 6). If multiple robots are tied, we choose the one that can reach the task location first (Alg. 2, line 7). We then use Algorithm 3 to add the task to the robot's schedule (Alg. 2, line 8). We repeat this process until all of the task requirements have been fulfilled (Alg. 2, line 5). We then update the task start time for the chosen task k^c according to the attending coalition (Alg. 2, line 10).

2) *Correctness of the algorithm*: We assert that our algorithm yields a feasible solution in which all tasks are allocated to suitable robots. To establish this claim, we demonstrate that the algorithm assigns each task to a set of appropriate robots. Suppose there is an unassigned task k with unfulfilled requirements, which means the sum of its requirements is greater than 0. The solver must continue until this task is assigned a group of robots that can fulfil all of its requirements. Hence, eventually a robot will choose this task, even if it can only provide a single skill. Once the solver has found a robot for task k , it will search for other robots to fulfil any remaining requirements. A feasible solution requires at least one robot to contribute at least one skill to the remaining requirements. As long as such a robot exists, it will be assigned to task k . Moreover, the solver will not choose a robot that cannot contribute to the task as long as there is a robot that can contribute at least one skill. Therefore, there can be no redundant robot assigned to any task. We conclude that our algorithm always terminates with a feasible solution. Therefore, we can conclude that the proposed greedy algorithm for task allocation is correct and produces a feasible solution.

IV. EXPERIMENTAL EVALUATION

We conducted experiments with 4 robots and 8 tasks, testing 3 configurations with 2, 4, and 8 skills. Each configuration comprised of 30 unique setups that differed in the location of the tasks, their skill requirements, and the allocation of skills among the robots.

A. Experimental Setup

In our experiments, we define an effective area of 200×200 units, and we assume that each robot can travel 1 unit of distance per time unit. For each configuration setup, we randomly assign the locations and skill requirements for each task. The task execution time of each task is set uniformly at random from the range $[0, 100]$. We also allocate skills to the robots uniformly at random. We set the starting locations of the robots to be evenly distributed around the center of the experiment area. Specifically, the starting location (p_x^i, p_y^i) of the i^{th} robot is calculated using the value $r = 15$ units as follows:

$$(p_x^i, p_y^i) = \left(r \sin\left(\frac{i\pi}{n}\right), r \cos\left(\frac{i\pi}{n}\right) \right). \quad (21)$$

We verify the validity of each generated experiment setup by checking the following conditions:

- 1) Each robot is not allocated more than $l/2$ skills;

- 2) Every skill is present at least once in the robot pool;
- 3) Every robot possesses at least one skill.

a) *Stochasticity parameters:* The value of the mean travel time, denoted by μ , was set to 10% of the time it takes for the robot to travel between tasks. The value of the standard deviation, denoted by σ , was set as a random fraction of μ . Specifically, a value was chosen uniformly at random from the interval $[0.05, 0.50]$ and multiplied by μ to obtain the final value of σ . This ensured that the amount of variability in the travel times was proportional to the mean travel time. To make the experiment results repeatable, the standard deviation value for each path is assigned at the time of setup generation. This ensures that the same standard deviation values are used throughout all the experiment runs. Finally, the probability of a robot arriving at a task within a given time window, denoted as ϵ , was set to 0.95 to allow for some flexibility in task scheduling.

b) *Computer specifications:* The experiments were run on a computing cluster with the following configuration allocation: AMD EPYC 7543 processors, 22 CPU cores, 156 GB of RAM.

B. Discussion

To evaluate the performance of the two methods, we analyze two key aspects of the solution. The first one is the final cost of the solution produced. This tells us the quality of the produced solution. The second aspect of interest pertains to the wall clock time required to solve the problem. This allows us to assess the efficiency of the two algorithms in terms of the computational resources and time complexity. A sample of optimal solution for a 2-skill, 4-robot, 8-task setup is reported in Figure 3.

1) *Optimal solver:* Figure 4 presents the results for the optimal solver. As we double the total number of skills required, both the solution cost and the wall clock solving time (WCST) increase. However, the notable increase is in the WCST of the 8-skilled setups. This indicates a significant increase in the computational resources required to solve the problem at higher scale. In this configuration, some setups required about 2,000 seconds and one of the setup required 5,000 seconds to declare the final solution as optimal. Based on these results, it would not be realistic to solve a problem larger than the setup presented, as the computational resources required would be prohibitively high.

2) *Greedy solver:* To analyze the greedy solver, we compared its performance with the optimal solver. Figure 5 compares the performance of the greedy solver to that of the optimal one. In Figure 5a we can see that for the configuration with 2 skills, multiple points lie near 1. This means that most of the solutions were very close to the optimum. The median relative cost performance of the greedy solver was 1.15.

The greedy solver performs very well in terms of computational efficiency, as shown in Figure 5b. In 2-skills setups, the median \log_{10} relative run-time is -3.57, indicating that the greedy solver is more than three orders of magnitude faster than the optimal solver. These results showcase the

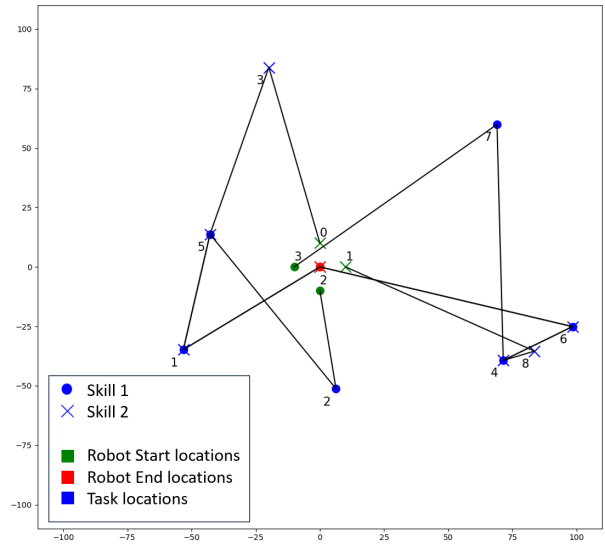
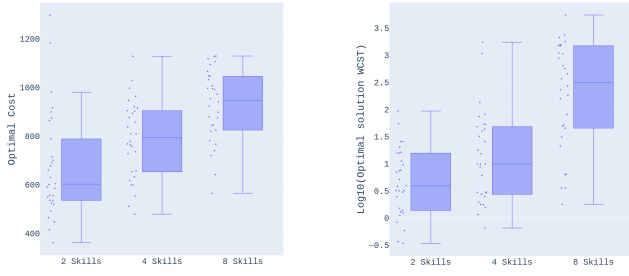


Fig. 3: An example solution for a setup with 2 skills, 4 robots, and 8 tasks. The two types of skills are represented by a *cross* and a *circle*. The robots’ starting locations are shown by green icons near the center of the area, and their end locations are shown by red icons at the center. If a robot with a *cross* skill starts at a location, a green cross is shown at that location. Blue skill symbols indicate the skills needed for each task at that location, and the tasks are randomly placed in the simulation area. The solid lines show the schedule for each robot, from its starting location to its end location.

fast nature of the greedy solver and its potential for use in scenarios where real-time decision-making is required.

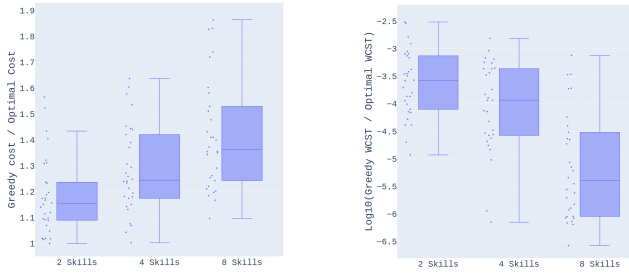
As we scale up the problem, the performance of the greedy solver in terms of cost slightly degrades. For the configuration with 8 skills, none of the experiments produced a solution close to the optimum. As indicated by the median cost performance of 1.36, the greedy solutions are off the optimal solution by a significant margin. However, as shown in Figure 5b, most solutions generated using the greedy solver are produced within a factor of 10^{-5} of the time it took to solve the same problem using the optimal method, demonstrating the efficiency of the greedy solver. Although the quality of the solutions generated by the greedy solver may not be very good for larger problem sizes, the method is extremely fast and can be useful for scenarios where quick, “good enough” decision-making is prioritized over solution optimality.

3) *Optimal solver’s first solution:* It is interesting to analyze the time taken by the optimal solver to reach its first solution, neglecting the time needed to verify whether it is optimal. As shown in Figure 6a, for the configuration with 8 skills, the solver rapidly produced a feasible but sub-optimal solution. However, it took a significantly longer time to reach the optimal solution and even more time to prove its optimality. This is further illustrated in Figure 6b, which displays the time taken by the solver to obtain the first feasible solution. The optimal solver consistently produces



(a) Optimal costs for each configuration (b) Wall clock solving time (WCST) for each configurations

Fig. 4: Optimal solution cost and solving times for three different set of configurations with varying number of skills. We can see that as the number of skills increase, both the solution cost and the solving time increases.



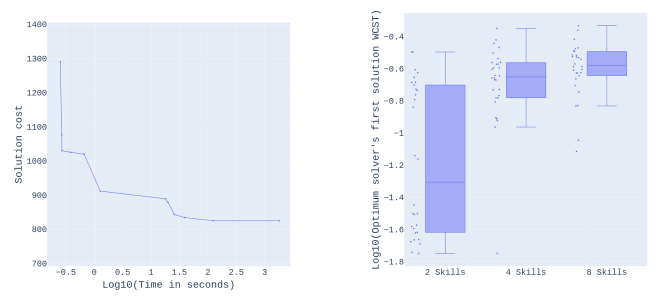
(a) Solution quality of the greedy solver (b) Relative speed of the greedy solver

Fig. 5: Performance of the greedy solver as compared with the optimal solver. We can see that the greedy solution gets further away from the optimum solution as we increase the number of skills. But it takes the greedy solver orders of magnitude lesser amount of time, as compared with the optimum solver

the first solution quickly, but it takes a long time to reach the optimum and prove its optimality (Figure 4b).

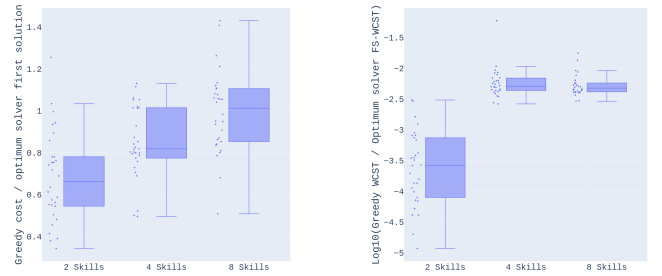
We compared the performance of the greedy solver with the first solution offered by the optimal solver in Figure 7. As shown in Figure 7a, most of the solutions produced by the greedy solver are better than the first solutions provided by the optimal solver. While the optimal solver's solutions are superior half of the times with the 8-skill configurations, the greedy solver is still faster. The greedy solver is consistently more than two orders of magnitude faster than the optimal solver, as shown in Figure 7b. The data clearly shows the significant speed advantage of the greedy solver over the optimal solver.

4) *Large-scale experiments:* To further investigate the speed and scalability of the proposed methods, we conducted a series of experiments on larger-scale configurations. We set the number of robots at 32 and the number of skills at 64, and generated 30 setups for each of four task counts: 128, 256, 512, and 1,024. For such large scales, the optimal solver failed to produce even the first solution after running for three hours. Hence, we analyze only the greedy solver's performance in what follows.



(a) Solution cost progression by the optimum solver for a typical setup of 8 skills (b) Time taken to produce the first feasible solution by the optimum solver

Fig. 6: Progression of the solution cost produced by the optimum solver. The solver finds a sub-optimal solution quickly, but then spends long time improving and proving the solution as the optimal.



(a) Comparing greedy solver's and optimum solver's first solution costs (b) Comparing greedy solver's and the optimum solver's first solution (FS) WCST

Fig. 7: Performance of the greedy solver as compared with the optimal solver's first solution. We can see that the greedy solutions are mostly better than the first solutions offered by the optimum solver. The greedy solver is also orders of magnitude faster, as compared with the optimum solver.

Figure 8 displays the \log_{10} of the wall clock times required by the greedy solver to solve each of these configurations. The results reveal that the \log_{10} of the WCST required to solve the larger-scale configurations increases by approximately 0.6 for each doubling of the number of tasks. In other words, as the number of tasks doubles, the solve times increase by a factor of approximately four. Despite this increase, we consistently obtained solutions for the configuration of 1,024 tasks within 45-50 seconds. These results suggest that the greedy solver can handle large-scale instances efficiently, making it a promising approach for real-world scenarios with a large number of tasks.

V. CONCLUSION

In this work, we presented an approach to task allocation in heterogeneous multi-robot systems. Our problem combines coalition and scheduling of a heterogeneous swarm of multi-skilled robots. Our problem formulation also includes stochastic aspects of travel between any two tasks. We proposed two methods to solve this problem. The first

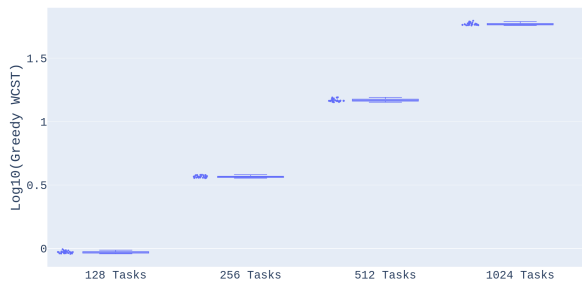


Fig. 8: Wall clock solving times (WCST) required by the greedy solver for large set of tasks

produces an optimal solution at the expense of long run-times. This method is only suitable for small-scale problems where optimality is required. Our second proposed method uses a greedy approach and it quickly produces sub-optimal solutions.

We compared the performance of the two methods. We found that the greedy solver is typically between 2x the cost of the optimal solution, but it offers speedups in the order of 10^5 with respect to the optimal solver. Further, the greedy solver can tackle large-scale scenarios (32 robots, 64 skills, and 1,024 tasks) in less than a minute. This makes the greedy solver a viable option for quick but best-effort decision-making.

In future work, we aim to improve the performance of the greedy solver by using better heuristics. We also aim to make the system decentralized to promote parallelism.

REFERENCES

- [1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.
- [2] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by reconfiguration: Exploiting heterogeneity in robot teams," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6518–6525.
- [3] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," *arXiv preprint arXiv:2109.12343*, 2021.
- [4] L. Barton and V. H. Allan, "Adapting to changing resource requirements for coalition formation in self-organized social networks," in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, 2008, pp. 282–285.
- [5] L. Capezzuto, A. Tarapore, and S. Ramchurn, "Anytime and efficient coalition formation with spatial and temporal constraints," in *Multi-Agent Systems and Agreement Technologies*, N. Bassiliades, G. Chalkiadakis, and D. de Jonge, Eds. Cham: Springer International Publishing, 2020, pp. 589–606.
- [6] L. Capezzuto, A. Tarapore, and S. D. Ramchurn, "Multi-agent routing and scheduling through coalition formation," *arXiv preprint arXiv:2105.00451*, 2021.
- [7] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.
- [8] C. Nam and D. A. Shell, "Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 193–200, 2016.
- [9] T. Rahwan, "Algorithms for coalition formation in multi-agent systems," Ph.D. dissertation, University of Southampton, 2007.
- [10] M. Guo, B. Xin, J. Chen, and Y. Wang, "Multi-agent coalition formation by an efficient genetic algorithm with heuristic initialization and repair strategy," *Swarm and Evolutionary Computation*, vol. 55, p. 100686, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S221065021831054X>
- [11] J. Hu, H. Coffin, J. Whitman, M. Travers, and H. Choset, "Large-scale heterogeneous multi-robot coverage via domain decomposition and generative allocation," in *International Workshop on the Algorithmic Foundations of Robotics*, 2022.
- [12] C. Huang and R. Liu, "Inner attention supported adaptive cooperation for heterogeneous multi robots teaming based on multi-agent reinforcement learning," *arXiv preprint arXiv:2002.06024*, 2020.
- [13] C. Lin, W. Luo, and K. Sycara, "Online connectivity-aware dynamic deployment for heterogeneous multi-robot systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8941–8947.
- [14] J. Parker, E. Nunes, J. Godoy, and M. Gini, "Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork," *Journal of Field Robotics*, vol. 33, no. 7, pp. 877–900, 2016.
- [15] A. Visser, L. G. Nardin, and S. Castro, "Integrating the latest artificial intelligence algorithms into the robocup rescue simulation framework," in *Robot World Cup*. Springer, 2018, pp. 476–487.
- [16] J. Guerrero, G. Oliver, and O. Valero, "Multi-robot coalitions formation with deadlines: Complexity analysis and solutions," *PLOS ONE*, vol. 12, no. 1, pp. 1–26, 01 2017. [Online]. Available: <https://doi.org/10.1371/journal.pone.0170659>
- [17] S. D. Ramchurn, M. Polukarov, A. Farinelli, N. Jennings, and C. Trong, "Coalition formation with spatial and temporal constraints," in *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010) (30/04/10)*, 2010, pp. 1181–1188, event Dates: May 2010. [Online]. Available: <https://eprints.soton.ac.uk/268497/>
- [18] M. Koes, I. Nourbakhsh, and K. Sycara, "Constraint optimization co-ordination architecture for search and rescue robotics," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 3977–3982.
- [19] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2022.
- [20] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021, pp. 1017–1023.
- [21] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "xbots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 115–122.
- [22] A. Mansfield, S. Manjanna, D. G. Macharet, and M. Ani Hsieh, "Multi-robot scheduling for environmental monitoring as a team orienteering problem," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6398–6404.
- [23] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.
- [24] —, "Fast redistribution of a swarm of heterogeneous robots," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONET-ICS)*, 2016, pp. 249–255.
- [25] O. Kosak, C. Wanninger, A. Hoffmann, H. Ponsar, and W. Reif, "Multipotent systems: Combining planning, self-organization, and reconfiguration in modular robot ensembles," *Sensors*, vol. 19, no. 1, p. 17, 2018.
- [26] S. Amador, S. Okamoto, and R. Zivan, "Dynamic multi-agent task allocation with spatial and temporal constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.
- [27] I. Tkach and S. Amador, "Towards addressing dynamic multi-agent task allocation in law enforcement," *Autonomous Agents and Multi-Agent Systems*, vol. 35, pp. 1–18, 2021.
- [28] S. Mayya, D. S. D'antonio, D. Saldaña, and V. Kumar, "Resilient task allocation in heterogeneous multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1327–1334, 2021.
- [29] R. K. Ramachandran, P. Pierpaoli, M. Egerstedt, and G. S. Sukhatme, "Resilient monitoring in heterogeneous multi-robot systems through network reconfiguration," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 126–138, 2021.
- [30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: <https://www.gurobi.com>