Online Connectivity-aware Dynamic Distribution for Heterogeneous Multi-Robot Systems

Chendi Lin

CMU-RI-TR-20-03

May 2020

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Katia Sycara, Chair
Maxim Likhachev
Wenhao Luo

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Keywords: Multi-Robot Systems, Networked Robots, Task Allocation

For my family, my friends, my dream, and myself.

Listen not to the rain beating against the trees.
Why don't you slowly walk and chant at ease?
Better than saddled horse I like sandals and cane.
O I would fain
Spend a straw-cloaked life in mist and rain.

Abstract

In many multi-robot applications, the robot team often needs to execute multiple tasks simultaneously with diverse capabilities and task-prescribed controllers. To ensure effective coordination and collaboration, robots from different tasks not only need to stay collision-free but also connected within each task subgroups as well as across different subgroups. In this thesis, we consider the dynamic task allocation and control problem, where a heterogeneous group of networked robots must be assigned and moved to multiple dynamic task places (not known beforehand) to maximize the overall task performance over time. As each task requires various capabilities from the assigned robots and has different weights of importance, the objective is to find the optimal combination of robots assigned to each task and the controllers to drive the robots towards the task sites, while enforcing safety and connectivity (global and subgroup) guarantee. In particular, we propose a unified optimization-based framework for the robots to compute the real-time task assignments with bounded optimality and the controllers that ensure inter-robot collision avoidance and connectivity maintenance. Such a framework allows for autonomous task re-assignment and redistribution of robots to improve task fulfillments during execution in the presence of dynamic task requirements. Simulation and numerical results are provided to demonstrate the effectiveness of the proposed approaches.

Acknowledgments

Firstly, I would love to express my sincere gratitude to my advisor Prof. Katia Sycara. She has been very supportive through my master study. Her enthusiasm towards the work, enormous experiences, and massive knowledge greatly inspired me during these two years. Her guidance and advice made this thesis possible.

I would also like to thank Prof. Maxim Likhachev as one of my thesis committee members. His enlightening comments and suggestions furnish this thesis work further.

Additional thanks to Wenhao Luo, who is also one of my thesis committee members from our lab. All the insightful conversations with him are not only the catalysis of this work, but also great directions for my future career.

Last but not least, I would like to thank my family and all my friends inside and outside the lab who encourage me and support me the whole time.

Contents

List of Figures

Chapter 1

Introduction

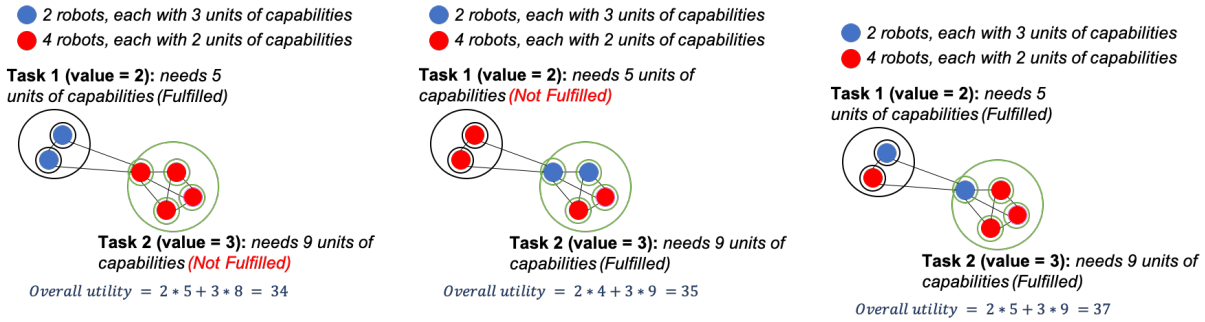
1.1 Motivation

Multi-robot systems are powerful in their ability to perform different tasks in parallel. Typically, the robots are distributed to different tasks based on the their capabilities, the importance, and the needs of the tasks. A wise and dynamic task allocation are essential in various applications. For instance, in a surveillance task, different buildings need different numbers of robots to monitor, based on the size of the building and the mobility of the robots. The importance and the requirements of each building can be dynamically updated according to the users' judgements and the uncertainty of the environment. Also, in a robot soccer game, the players need to be dynamically reassigned between attackers and defenders as the game develops. Rapid and adaptive redistribution is needed to be carried out any time. Such problems are naturally formulated into combinatorial problems. In a combinatorial problem, the same task demands can be achieved by various solutions, which escalates the complexity. For example, a task may involve a target that needs 8 units of capability. It can be supplied by 4 robots, each of which has 2 units of capability, or by 3 robots, where two have 3 units of capability each and one that has 2 units of capability.

In this work we develop a framework that can adaptively distribute the robot in a dynamic environment, while taking connectivity constraints into considerations. In this problem, n robots and m are present. Each task has its value and required units of capabilities, and each robot has its units of capabilities. The overall system utility in this work is calculated as the summation of the capabilities provided by the robots to each task weighted by the task's value. Correspondingly, the remaining system utility is the summation the remaining unfulfilled requirements of each task weighted by its value. Notice that since we are dealing a changing environment in this work, the number of target m is fixed over the whole time horizon. The importance and requirements may also be unknown beforehand, which requires at least one robot to explore the target. Moreover, one task may involve a target where the robots would be concentrated in the middle of the target for better monitoring, whereas another task may require the robots to encircle the target. Therefore, different controllers would also be needed. In this work, we discuss our approach using the illustrative example of covering targets that may appear dynamically. However, the method applies to multiple other tasks. Figure 1.1 shows an example of such problems. In this example, two types of robots are allocated to two tasks with known locations, areas, and

needs. We mark robots with different capabilities by different colors. Allocating more than the required capabilities (over-allocating) does not bring any more utility. Figure 1.1a and Fig. 1.1b lead to a lack of 1 unit in one of the tasks whereas the allocation in Fig. 1.1c fulfills the requirements of both tasks. Since the number of tasks and their requirements are not known a priori but are discovered as the robots move, the robots may need to be redistributed as new tasks are discovered. Since the robots are connected, if a "front" robot sees a new target, it must notify the others, so that a reallocation may be done. In this reallocation the traveling time of the robots that are candidates to be re-assigned to the new target must also be considered.

To achieve collaboration, robots are expected to exchange messages and share information, which requires the robot members to stay connected within at least one neighbor's communication range. Additionally, maintaining connectivity of the whole multi-robot system is important to ensure that the various robot groups that are assigned to different tasks can rejoin with one another after they have finished their tasks, so as to be re-allocated as new tasks arise. For example, the robots can be asked to form various formations or surround multiple targets simultaneously. To achieve such behaviors, multiple subgroups of the multi-robot system must be involved, and the connectivity both within the subgroups and within the whole systems should be guaranteed. However, such connectivity constraints add additional complexity into the problem because they can impede the multi-robot system from following the planned allocation. Figure 1.2 is an illustration of the issue that arises from the connectivity constraints. In all three scenarios, the capabilities provided by the allocated robots surpass the demands of the tasks. Nonetheless, some robots might never be able to reach their assigned target areas because of the connectivity constraints as shown in Fig. 1.2a and Fig. 1.2b. In those situations, we want the robots to be effectively redistributed to the configuration shown in Fig. 1.2c.

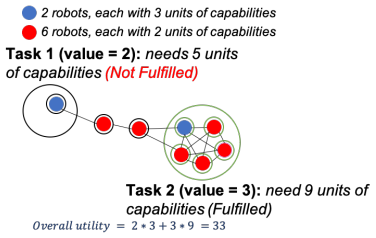


(a) Task 1 is fulfilled but Task 2 is not.

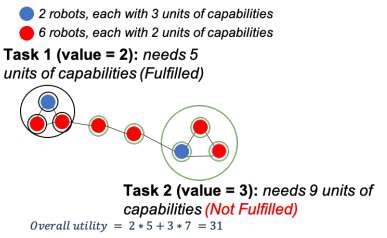
(b) Task 2 is fulfilled but Task 1 is not.

(c) Both tasks are fulfilled.

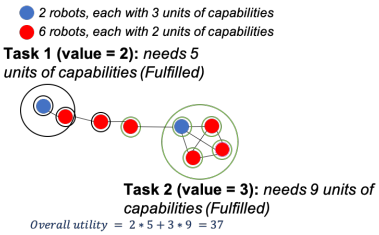
Figure 1.1: Illustration of combinatorial task allocation that two types of robots allocated to two different tasks: 2 blue robots with 3 units of capabilities each and 4 red robots with 2 units of capabilities each. Task 1 requires 5 units of capabilities, and Task 2 needs 9 units. With the assignment in (a), Task 1 is over-allocated by 1 extra unit by two blue robots while Task 2 lacks 1 unit. In (b) Task 2 is over-allocated whereas Task 1 is under-allocated by 1 unit. The combination shown in (c) can fully meet the requirements of both tasks.



(a) Task 1 is fulfilled but Task 2 is not.



(b) Task 2 is fulfilled but Task 1 is not.



(c) Both tasks are fulfilled.

Figure 1.2: Illustration of issues brought by preserving connectivity within the robot team: 2 blue robots with 3 units of capabilities each and 6 red robots with 2 units of capabilities each are allocated to two tasks. Task 1 requires 5 units of capabilities, and Task 2 needs 9 units. Both requirements should be overly fulfilled by 2 units. However, because of the connectivity constraints, in (a) Task 1 is under-allocated by 2 units, and (b) Task 2 is under-allocated by 2 units. Only the configuration shown in (c) can satisfy both demands.

1.2 Thesis Contribution

Most of the existing works deal with static task allocation or dynamic assignment at an abstract level [1], [2], where the detailed configuration of the robot team is usually not addressed. In this thesis, we propose an online connectivity-aware and dynamic redistribution method to accommodate the changing environment systematically and address the limitations in fulfilling the requirements of tasks brought by connectivity constraints. The problem is formulated into an optimization problem which yields the sub-optimal assignment and controllers of the robots, constrained by connectivity, so that both the unfulfilled requirements of the tasks prioritized by their values and the control efforts are minimized. Since the tasks are discovered during execution, at each time step, the combinatorial allocation problem is solved via a greedy algorithm with provable optimality bound and polynomial computation complexity. With the task and the primary controller assigned to each robot, we solve a quadratic programming (QP) problem by minimizing the difference between its current control and the desired control, weighted by the importance and the unfulfilled requirements of its assigned task, with connectivity maintenance and collision avoidance. The collision avoidance and connectivity maintenance are expressed using control barrier functions [3, 4] and the minimally disruptive global and subgroup connectivity maintenance is achieved using our previous work in Behavior Mixing [4].

The contributions of our work are: (1) it presents a systematic framework that combines combinatorial task allocation with the low-level control to generate different robot configurations with connectivity constraints that maximizes the overall system utility; (2) it proposes a problem with dynamic and uncertain environment that requires exploration and it solves the problem by applying our framework; (3) it provides the sub-optimality proof of the greedy assignment, along with the simulation results and the numerical analysis.

1.3 Organization

As discussed in Section 1.2, this thesis presents a systematic framework that achieve a dynamic combinatorial task allocation strategy that can be easily generalized and adaptive control that execute the assignment to the greatest extent while reserving the connectivity constraints. The rest of the thesis is organized as the following.

In Chapter 2 we present the related work in efficient task allocation and dynamic robots redistribution. Various formulations and approaches that solve dynamic distribution of robots are briefly discussed.

Chapter 3 shows the definition of the problem. In Section 3.1, the notations in this thesis are introduced. In Section 3.2, we describe the problem that we are solving and the mathematical optimization formulation of the problem.

Chapter 4 presents the methods and algorithms to embed the dynamic redistribution at the control level. In Section 4.1, a greedy method is presented to greedily allocate the robots in a dynamic environment. In Section 4.2, we analyze and prove the optimality bound of this algorithm. In Section 4.3, the enforcement of connectivity maintenance in control rendering is described. In Section 4.4, we present a redistribution method that drives the multi-robot system to a more desirable configuration and generates the control outputs for the robots to execute at every time step.

In Chapter 5 we show the simulation results and numerical results. In Section 5.1, we illustrate some experiments in different scenarios and demonstrate how the multi-robot system configures itself adaptively in a changing environment. In Section 5.2, we present the numerical outcomes, comparisons, and analysis of the experiments.

Chapter 6 concludes the evaluation and future work.

Chapter 2

Related Work

For multi-robot systems, task allocation is a crucial and extensively studied topic [5], [6]. In a Multi-Robot Task Allocation (MRTA) problem [6], multiple tasks with different priorities require various capabilities from heterogeneous robots which is also known as a complex task problem following Zlot's definition of task types [7]. Instead of considering the required number of robots, we want to provide an ideal combination of capabilities (e.g. sensor payloads) to fulfill the requirement of each task. The problem is formulated as an NP-hard combinatorial optimization problem [8]. Such problems can commonly be solved by mixed integer nonlinear programming (MINLP) [9], or be formulated into a distributed constraint optimization problems (DCOPs) [10] that can be solved by self-adjusting algorithms like MGM [11] and the Distributed Stochastic Algorithm (DSA) [12], [13]. These distributed methods can solve the problem rapidly with good scalability. Combinatorial Auction-based algorithms are also widely applied, for example, on allocation of virtual machine instances [14].

Along with the algorithms that solve the task allocation problems for heterogeneous groups, many papers present works on the dynamic redistribution to adapt the failures of the robots or the modifications of the knowledge about missions in a changing environment. In [15] and [16], the authors utilized constraint-based optimization to dynamically allocate heterogeneous robots along with the task execution controls. In [1] and [17], the reallocation is characterized as transition rates for the robot groups based on their trait distribution. [18] presents a method to quickly reconfigure the network of the heterogeneous multi-robot system when their shared resources fail. In [19], the authors presented an approach to dynamically reassign the homogeneous robots to multiple locations in accordance with the changes in the environment. These dynamic redistribution strategies mostly neglect the actual arrangement and controls of the robot team and the existence of connectivity constraints. In contrast, this thesis studies the case when multiple robots are executing different tasks with diverse importance and requirements and have connectivity constraints. Both the allocation plan and control execution are produced adaptively to meet the combinatorial task requirements.

To exchange messages and achieve collective behaviors, connectivity maintenance is essential to a multi-robot system. Many researchers have proposed various connectivity control methods in three main categories: local connectivity [20, 21, 22], global algebraic connectivity [23, 24, 25], and control barrier function based connectivity [4, 26, 27, 28]. In this thesis, to ensure the connectivity among multiple subgroups performing different behaviors, the connec-

tivity is constructed using Minimum Connectivity Constraint Spanning Tree (MCCST) [4, 27], and the controls are rendered by Behavior Mixing [4]. In Behavior Mixing, the goal is to minimize the difference between the output revised controller and the original controller subject to the constraints using the control barrier functions of connectivity reservation [4] and collision avoidance [3]. This problem is formulated as a quadratic programming (QP), which has the advantages in computational efficiency and convergence [29]. With the help of this framework, the combinatorial task allocation problem is solved adaptively at every time step and the distribution of the robots is adjusted at the control level according to the current state without paying extra computational costs.

Chapter 3

Problem Formulation

3.1 Notation

Consider a multi-robot system with n heterogeneous robots that operates in a planar space. Denote the robot set as $\mathcal{A} = \{1, 2, ..., n\}$. Each robot i has its capability quantified by c_i, and its location is defined by a 2-dimensional vector $\boldsymbol{x}_i$. In the problem, m tasks are given and the task set is encoded as $\mathcal{J} = \{1, 2, ..., m\}$ [1]. The location of each task j is denoted as $\boldsymbol{y}_j$, which is also a 2-dimensional vector. The task assignment is defined as a set $\mathcal{S}$ containing robot-task pairs. It is equivalent to a binary matrix S of size $n \times m$, i.e., robot i is assigned to task j if and only if $S_{ij} = 1$. Since each robot can only be allocated to at most one task, $\forall i \in \mathcal{A}$, $\sum_{j=1}^{m} S_{ij} \leq 1$. Naturally, we denote the set of robots assigned to task j as $\mathcal{A}_j$. Notice that, at the beginning the robots are all assigned to task 0 to indicate that they are available. The importance of each task j is evaluated as a non-negative constant v_j, and the number of its required capability units is given as w_j. At each time step, we assess the remaining required capability of task j as r_j, which is always larger than or equal to 0. The robots are expected to execute the task the moment they enter within range L from the task.

3.2 Problem Statement

Our objective is to achieve an adaptive distribution so that the overall utility of the robot team can be maximized and allocation to the newly discovered targets can be solved while, in addition to the required capabilities and target importance, we also consider traveling costs and connectivity constraints.

Consider a team of n heterogeneous robots with their capabilities given as a vector $\boldsymbol{c}$ of size n. Before the exploration, the values of tasks along with their requirements are unknown. As robots start to move, they discover new targets. Target discovery requires at least one robot to visit the target j to acquire the information on the targets importance v_j and required capabilities w_j. At every time step, after an assignment has been made that covers some of the target requirements, the remaining requirement of task j, r_j, is calculated as the original requirement w_j minus the

[1] Since we solve the assignment problem at each time step, the number m of tasks that are present is known but changing as execution proceeds.

sum of capabilities of all the robots that are executing the mission and is lower bounded by 0. The discovered target's position is is given as a 2-dimensional vector $\boldsymbol{y}_j$. *Notice that the number of targets m is known at every time step when solving the allocation problem, but it is not a fixed constant for the whole problem horizon because new tasks can appear in the middle of mission executions.* The goal is to minimize the weighted sum of the total remaining requirements along with the traveling distance of each robot. To achieve that, an optimal assignment matrix S^* is produced and controls $\boldsymbol{u}^* \in R^{2n}$ are provided to distribute the robots. The objective is expressed as

$$S^*, \boldsymbol{u}^* = \arg\min_{S,\boldsymbol{u}} \sum_{j=1}^{m} \left(v_j r_j - \alpha \sum_{i \in \mathcal{A}_j} \frac{1}{1 + \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2} \right) \tag{3.1}$$

$$S^*, \boldsymbol{u}^* = \arg\min_{S,\boldsymbol{u}} \sum_{j=1}^{m} \left(v_j(\max(0, w_j - \sum_{i \in \mathcal{A}_j} c_i)) - \alpha \sum_{i \in \mathcal{A}_j} \frac{1}{1 + \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2} \right) \tag{3.2}$$

$$S^*, \boldsymbol{u}^* = \arg\max_{S,\boldsymbol{u}} \sum_{j=1}^{m} \left(v_j(\min(w_j, \sum_{i \in \mathcal{A}_j} c_i)) + \alpha \sum_{i \in \mathcal{A}_j} \frac{1}{1 + \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2} \right) \tag{3.3}$$

$$S^*, \boldsymbol{u}^* = \arg\max_{S,\boldsymbol{u}} \left(\boldsymbol{v}^T \min(\boldsymbol{w}, S^T \boldsymbol{c}) + \alpha \sum_{j=1}^{m} \sum_{i \in \mathcal{A}_j} \frac{1}{1 + \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2} \right) \tag{3.4}$$

The first term of the objective function is the effective utilities of the robots, and the second term is the inverse of the traveling costs with normalization, in which α is a scaling constant that determines how important the traveling cost is in our problem.

Chapter 4

Methods

In this chapter, our framework to obtain an optimal allocation plan and controls for the problem defined in Section 3.2 is described as following: In Section 4.1, we design a greedy algorithm that can quickly and adaptively generate a sub-optimal assignment at each time step, which is passed to the control-level optimization for further improvement. By proving the submodularity and monotonicity of our objective function in Section 4.2, we can deploy a greedy algorithm that has known suboptimality bound. In Section 4.3, we review a connectivity maintenance technique MCCST [4], which is utilized to produce minimum connectivity constraints of our optimization problem. Because of the connectivity constraints however, during most of the executions, robots cannot reach the ideal configurations. To address this issue, in Section 4.4, we present a control optimization framework, which extends the Behavior Mixing scheme with task allocation to achieve the desired configurations. In Section 4.5, we generalize the formulation to the multi-robot systems with multiple capabilities.

4.1 Greedy Dynamic Task Allocation

At every time step, we want to first obtain a allocation plan served as a baseline for the control rendering. Since task allocation is a strongly NP-hard problem [8], greedy methods are efficient and flexible choices with provable optimality [30, 31, 32] and polynomial computational complexity. In our work, the greedy algorithm is constructed to provide both the initial assignment and the adaptive re-assignments and is as following:

From Line 2 to Line 12, the rewards of all possible robot-task pairs are determined. The reward of matching task j to robot i is calculated as the summation of three terms gain g_{ij}, loss l_{ij}, and cost h_{ij} (Line 4 - 10), where j' is the original task that robot i is assigned to. If robot i is available, $j' = 0$ and its loss term is 0. Therefore, the reward p_{ij} is calculated as

$$p_{ij} = g_{ij} - l_{ij} + \alpha h_{ij} \tag{4.1}$$

where α is a scaling constant as stated in the explanations of the objective function.

In Line 13-16, the maximal reward among all pairs is found. Notice that the lower bound of p_i is α. If the maximal reward is equal to α, then the best strategy for all the remaining robots in set $\mathcal{A}$ is to stay with their original tasks and the assigning process can be early terminated (Line

Algorithm 1: Greedy Algorithm

Result: The assignment matrix S

while $\mathcal{A}$ *is not empty* **do**
  for *robot* $i \in \mathcal{A}$ **do**
    for *task* j *= 0 :* m **do**
      $g_{ij} = v_j \min(c_i, \max(0, w_j))$;
      if $j' = 0$ **then**
        $l_{ij} = 0$;
      else
        $l_{ij} = v_{j'} \max(0, \min(w_{j'} + c_i, c_i))$;
      end
      $h_{ij} = \frac{1}{1+\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2}$;
      $p_{ij} = g_{ij} - l_{ij} + \alpha h_{ij}$;
    end
    Find the maximal price p_i from task j_{best} to robot i or say marginal gain;
  end
  Collecting the reward p_i from all the robots;
  Choose robot i^* with the largest reward among all;
  if $p_{i^*} == \alpha$ **then**
    break ;
  else
    $j^* = j_{\text{best}}$;
  end
  $S_{i^*j'} = 0$;
  $S_{i^*j^*} = 1$;
  $w_{j^*} = w_{j^*} - c_{i^*}$;
  $w_{j}{}' = w_{j'} + c_{i^*}$;
  $\mathcal{A} \setminus i^*$;
  $w_0 = 0$;
end

18). A robot will be assigned to a task different from the one it was assigned earlier only when the improvement in requirement fulfillment is greater than the traveling cost (Line 20). After that, the assignment matrix S is updated, and so is the requirement vector $\boldsymbol{w}$. The selected robot i^* is removed from the robot set $\mathcal{A}$, and the demand of task 0 (stay still) is reset back to 0 (Line 22-27). The assignment process takes at most n rounds and for each round it requires at most mn reward evaluations. Thus, it can be easily proven that the computational complexity of the algorithm is $O(mn^2)$ in the worst case. However, in most scenarios since the algorithm is early terminated, the complexity is $O(1)$ when there is no update about tasks' information.

4.2 Algorithm Performance Analysis

In this section, we show that using this greedy algorithm, the solution quality is bounded by proving the submodularity and the monotonicity of our objective function. Let $\mathcal{V}$ be a finite set, which is also the grounded set of our problem. Denote the optimal assignment as $\mathcal{S}_{\text{opt}} \subseteq \mathcal{V}$. The objective function as defined before is

$$f(\mathcal{S}^*) = F(S^*) = \boldsymbol{v}^T(min(\boldsymbol{w}, {S^*}^T \boldsymbol{c})) + \alpha \sum_i h_i \tag{4.2}$$

which is always non-negative.

Corollary 1 *The solution $\mathcal{S}$ obtained from our greedy algorithm is guaranteed that $f(\mathcal{S}^*) \geq \frac{1}{2} f(\mathcal{S}_{opt})$*

To prove this corollary, we can apply the conclusion from Fisher's work [30] that if a function $f : 2^{\mathcal{V}} \rightarrow R$ is submodular and monotone, the greedy algorithm (taking the largest marginal gain at each step) is guaranteed a constant optimality bound as 1/2. To prove them, we need to define submodularity and monotonicity firstly.

Definition 1 *(Submodularity [33]) A function $f : 2^{\mathcal{V}} \rightarrow R$ is submodular if and only if for any $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{V}$, $\forall e \in \mathcal{V} \setminus \mathcal{Y}$, the following inequality holds*

$$\Delta_f(e|\mathcal{X}) \geq \Delta_f(e|\mathcal{Y}) \tag{4.3}$$

where $\Delta_f(e|\mathcal{S})$ is defined as

$$\Delta_f(e|\mathcal{S}) = f(\mathcal{S} \cup \{e\}) - f(\mathcal{S}) \tag{4.4}$$

Definition 2 *(Monotonicity [33]) A function $f : 2^{\mathcal{V}} \rightarrow R$ being monotonic is equivalent to show that*

$$\forall \mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{V}, f(\mathcal{Y}) - f(\mathcal{X}) \geq 0 \tag{4.5}$$

We prove the monotonicity as following.

$$f(\mathcal{Y}) = \sum_{j=1}^{m} v_j(\min(w_j, \sum_{i\in\mathcal{Y}_j} c_i)) + \sum_{i\in\mathcal{Y}_j} h_i \tag{4.6}$$

$$\begin{aligned} &= \sum_{j=1}^{m} v_j(\min(w_j, \sum_{i\in\mathcal{X}_j} c_i)) + \sum_{i\in\mathcal{X}_j} h_i + \sum_{i\in(\mathcal{Y}\setminus\mathcal{X})} h_i \\ &+ \sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i\in\mathcal{X}_j} c_i), \sum_{i\in(\mathcal{Y}\setminus\mathcal{X})_j} c_i)) \end{aligned} \tag{4.7}$$

where $\mathcal{S}_j$ refers to the set containing all the robots assigned to task j in assignment $\mathcal{S}$. So we can rewrite it as

$$\begin{aligned} &f(\mathcal{Y}) - f(\mathcal{X}) \\ =&\sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i\in\mathcal{X}_j} c_i), \sum_{i\in(\mathcal{Y}\setminus\mathcal{X})_j} c_i)) + \sum_{i\in(\mathcal{Y}\setminus\mathcal{X})} h_i \end{aligned} \tag{4.8}$$

Notice that here $f(\mathcal{Y}) - f(\mathcal{X}) \leq f(\mathcal{Y} \setminus \mathcal{X})$ because when the requirement of the task is overly satisfied, designating more robots to it cannot lead to further improvements. The term $\min(\max(0, w_j - \sum_{i\in\mathcal{X}_j} c_i), \sum_{i\in(\mathcal{Y}\setminus\mathcal{X})_j} c_i)$ in Eqn. 4.8 summarizes the marginal gain of $\mathcal{Y}$ compared to $\mathcal{X}$ in all scenarios, which are listed out as below:

1. When neither assignment $\mathcal{Y}$ nor $\mathcal{X}$ can fulfill the requirement of task j, the improvement of $\mathcal{Y}$ compared to $\mathcal{X}$ with respect to task j is $\sum_{i\in(\mathcal{Y}\setminus\mathcal{X})_j} c_i$.

2. When task j is overly satisfied with assignment $\mathcal{Y}$ but not with assignment $\mathcal{X}$, the enhancement of $\mathcal{Y}$ as to $\mathcal{X}$ is $w_j - \sum_{i\in\mathcal{X}_j} c_i$.

3. When the requirement of task j is met by both assignment $\mathcal{Y}$ and $\mathcal{X}$, the improvement of $\mathcal{Y}$ compared to $\mathcal{X}$ regarding task j is 0.

Since all the terms above are non-negative,

$$f(\mathcal{Y}) - f(\mathcal{X}) \geq 0 \tag{4.9}$$

Thus, our objective function f is monotonic.

To prove the submodularity, by applying Eqn. 4.4, we can write Eqn. 4.3 as

$$\Delta_f(e|\mathcal{Y}) - \Delta_f(e|\mathcal{X}) \leq 0 \tag{4.10}$$

$$f(\mathcal{Y} \cup \{e\}) - f(\mathcal{Y}) - f(\mathcal{X} \cup \{e\}) + f(\mathcal{X}) \leq 0 \tag{4.11}$$

$$(f(\mathcal{Y} \cup \{e\}) - f(\mathcal{Y})) - (f(\mathcal{X} \cup \{e\}) - f(\mathcal{X})) \leq 0 \tag{4.12}$$

To prove that, we can rewrite the left hand side using Eqn. 4.8,

$$(f(\mathcal{Y} \cup \{e\}) - f(\mathcal{Y})) - (f(\mathcal{X} \cup \{e\}) - f(\mathcal{X})) \tag{4.13}$$

$$= \sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i), e_{ij}c_i)) + h_{e_i} - \sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i), e_{ij}c_i)) - h_{e_i} \tag{4.14}$$

$$= \sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i), e_{ij}c_i)) - \sum_{j=1}^{m} v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i), e_{ij}c_i)) \tag{4.15}$$

$$= \sum_{j=1}^{m} v_j \left(\min(\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i), e_{ij}c_i) - \min(\max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i), e_{ij}c_i) \right) \tag{4.16}$$

where $e_{ij} = 1$ if and only robot i is assigned to task j in assignment e. Otherwise, $e_{ij} = 0$. Since $\mathcal{X}$ is a subset of $\mathcal{Y}$, we know that there are fewer assignment pairs existing in $\mathcal{X}$, so $\forall j$ the following inequality always holds

$$\sum_{i \in \mathcal{Y}_j} c_i \geq \sum_{i \in \mathcal{X}_j} c_i \tag{4.17}$$

$$w_j - \sum_{i \in \mathcal{Y}_j} c_i \leq w_j - \sum_{i \in \mathcal{X}_j} c_i \tag{4.18}$$

$$\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i) \leq \max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i) \tag{4.19}$$

$$\min(\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i), e_{ij}c_j) \leq \min(\max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i), e_{ij}c_j) \tag{4.20}$$

$$v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{Y}_j} c_i), e_{ij}c_j)) \leq v_j(\min(\max(0, w_j - \sum_{i \in \mathcal{X}_j} c_i), e_{ij}c_j)) \tag{4.21}$$

Therefore, by plugging Eqn. 4.21 back to Eqn. 4.16, we can see that Eqn. 4.12 holds, and thus, the objective function f is proven to be submodular.

With both properties proven, applying the conclusion from [33], the optimality bound of this algorithm is proven to be $f(\mathcal{S}^*) \geq \frac{1}{2} f(\mathcal{S}_{\text{opt}})$ as stated in Corollary 1.

4.3 Minimum Global and Subgroup Connectivity Maintenance

To collaborate, a multi-robot system is required to stay connected so that the robots can send peer to peer messages to each other. We define the constant communication range for all robots as R_c. The framework of Behavior Mixing [4] is employed to enable the multi-robot system to split into connected subgroups so that each subgroup can cover dynamically appearing targets, while

the global connectivity is preserved so that, after each subgroup has finished its task, it can rejoin the other groups. To secure connectivity among robots, a communication graph of the multi-robot system is maintained. The graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which each vertex $v \in \mathcal{V}$ represents a robot and each undirected edge $(v_i, v_j) \in \mathcal{E}$ represents the communication link. The communication link is established between robot i and robot j if the distance between these two robots is equal or less than R_c. At each time step, based on the current configuration, a minimum connectivity constraint spanning tree (MCCST) is built [4] so that the constructed Minimum Spanning Tree defines the optimal connectivity topology to preserve that satisfies global and subgroup connectivity while are least likely to be violated given the robots' original controllers at the moment. With that, we can define the feasible set of $\boldsymbol{x}$ for which the connectivity can always be preserved. In the communication graph, the communication link between robot i and robot j is preserved using the following expressions:

$$h^c_{i,j}(\boldsymbol{x}) = R_c^2 - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \tag{4.22}$$

$$\mathcal{H}^c_{i,j} = \{\boldsymbol{x} \in R^{2n} : h_{i,j}(x)^c \geq 0\} \tag{4.23}$$

With the MCCST graph $\mathcal{G}^c = (\mathcal{V}, \mathcal{E}^c)$, we can create a feasible set for $\boldsymbol{x}$ so that any connected pairs, i.e. communication links, in $\mathcal{G}^c$ should be in the set $\mathcal{H}^c(\mathcal{G}^c)$, which is expressed as

$$\mathcal{H}^c(\mathcal{G}^c) = \bigcap_{(v_i,v_j)\in\mathcal{E}^c} \mathcal{H}^c_{i,j} \tag{4.24}$$

Similarly, the safe set for inter-robot collision avoidance between robot i and j is defined as

$$h^s_{i,j}(\boldsymbol{x}) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 - R_s^2 \tag{4.25}$$

$$\mathcal{H}^s_{i,j} = \{\boldsymbol{x} \in R^{2n} : h^s_{i,j} \geq 0\} \tag{4.26}$$

$$\mathcal{H}^s = \bigcap_{(v_i,v_j)\in\mathcal{V}:i>j} \mathcal{H}^s_{i,j} \tag{4.27}$$

To avoid collision, we apply the barrier certificate function described in [3]

$$\mathcal{B}^s(\boldsymbol{x}) = \{\boldsymbol{u} \in R^{2n} : \dot{h}^s_{i,j}(\boldsymbol{x}) + \gamma h^s_{i,j}(\boldsymbol{x}) \geq 0, \forall i > j\} \tag{4.28}$$

Following the barrier certificate function used for collision avoidance, we can write a barrier function for connectivity maintenance as

$$\mathcal{B}^c(\boldsymbol{x}, \mathcal{G}^c) = \{\boldsymbol{u} \in R^{2n} : \dot{h}^c_{i,j}(\boldsymbol{x}) + \gamma h^c_{i,j}(\boldsymbol{x}) \geq 0, \forall (v_i, v_j) \in \mathcal{E}^c\} \tag{4.29}$$

where γ is a user-defined parameter to enclose the available set. It is proven that both $\mathcal{H}^c$ and $\mathcal{H}^s$ are forward invariant sets if the joint control input $\boldsymbol{u}$ stays in $\mathcal{B}^c \cap \mathcal{B}^s$ [26], [3]. Equation 4.29 will serve as the connectivity constraint in our optimization for controls, which is discussed in the next section, while Eqn. 4.28 serves as the collision avoidance constraint in the optimization. By using behavior mixing with global and subgroup connectivity, we enforce MCCST here to preserve an optimal subset of communication link with connectivity guarantee.

4.4 Redistribution with Heterogeneous Robots

As we mentioned in Section 1.1, because of the connectivity constraints, even given a good allocation, the missions still cannot be carried out as desired. For example, when a new task appears, one of robots will be assigned to explore it. Yet, constrained by all of its neighbors, it is impossible for this robot to examine the new task, unless there is a mechanism to help the robot drag its neighbors to the new goal while ensuring that the majority of the system is still performing their tasks. Here, we develop a method to achieve a minor adjustment in the distribution of the robots at the control level.

Without the connectivity constraints, assuming all the robots can reach their assigned targets, we calculate the remaining requirement of each task as

$$r_j = \max(0, w_j - \sum_{i \in A_j} c_i) \tag{4.30}$$

Then we can calculate vector $\boldsymbol{r}$ as

$$\boldsymbol{r} = \max(\boldsymbol{0}, \boldsymbol{w} - S^T \boldsymbol{c}) \tag{4.31}$$

As mentioned in Section 3.1, here we assume the mission is executed when robots rendezvous within range L around the task. Also, as a result of the connectivity constraints, not all robots are able to arrive at their allocated areas. In other words, the true remaining requirements $\hat{\boldsymbol{r}}$ can be different from the ideal remaining requirements $\boldsymbol{r}$ shown in Eqn. 4.31. The true remaining requirement of task j is calculated as

$$\hat{r}_j = \max\left(0, w_j - \sum_{i \in A_j} c_i H(L - d_{ij})\right) \tag{4.32}$$

where d_{ij} is the l^2- norm of the distance between the current position of robot i and the position of its assigned task j, calculated as $d_{ij} = \|\boldsymbol{x_i} - \boldsymbol{y_j}\|_2$. H is a Heaviside Step Function [34], which is defined as a function of x:

$$H(x) = \begin{cases} 1, \text{if } x \geq 0 \\ 0, \text{if } x < 0 \end{cases} \tag{4.33}$$

With that in mind, we calculate controls by employing the quadratic programming (QP) as following,

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \sum_{i=1}^{n} \|\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i\|^2 \tag{4.34}$$

$$\text{s.t.} \quad \boldsymbol{u} \in \mathcal{B}^s(\boldsymbol{x}) \cap \mathcal{B}^c(\boldsymbol{x}, \mathcal{G}^c) \tag{4.35}$$

where $\hat{\boldsymbol{u}}_i$ is the primary controller for robot i's assigned task and $\boldsymbol{u}^* \in R^{2n}$ contains the control inputs of all robots. For simplicity without losing generality, single integrator dynamics is used

for the task-related controller of each robot, i.e., $\hat{\boldsymbol{u}}_i = -K_p(\boldsymbol{x}_i - \boldsymbol{y}_j)$, and as defined in the previous section, $\boldsymbol{y}_j$ is the location of task j assigned to robot i.

Here, we want to to maximize the utility function, which is equivalent to reducing the weighted remaining unfulfilled requirements by assigning higher weights to the robots whose assigned tasks are of higher priority and less fulfilled. For each robot i that is assigned to task j, its coefficient a_i in the QP is calculated as

$$a_i = c_i v_j \hat{r}_j \tag{4.36}$$

In matrix form, we can write it as

$$\boldsymbol{a} = \boldsymbol{c} \odot (S(\boldsymbol{v} \odot \hat{\boldsymbol{r}})) \tag{4.37}$$

where $\odot$ is the element-wise vector multiplication.

Hence, how much the robot's primary controller is preserved depends on the value and the remaining unfulfilled requirements of its assigned task, and also the capability of the robot, which indicates how much improvement the robot can contribute to the task. Thus, the more capable robot will be encouraged to fulfill the gap of the more important and less fulfilled task, and the others will tend to serve as connectivity nodes.

When solving the QP, it is important that the objective function is differentiable. Now we have a step function H in the coefficient, and thus the robots might bounce back and forth near the boundaries. To avoid that, we want to replace it to be a differentiable function. A scaled and shifted sigmoid function σ is used here to replace the Heaviside step function. So we have

$$\begin{aligned} a_i &= c_i v_j \hat{r}_j & (4.38)\\ &= c_i v_j (\max(0, w_j - \sum_{i \in A_j} c_i \sigma(k(L - d_i) + b))) & (4.39) \end{aligned}$$

where k is a scale factor, and b is a shift constant. At the control level, to further endorse the robots moving towards more demanding tasks, we want to solve the QP as

$$\begin{aligned} \boldsymbol{u}^* &= \arg\min_{\boldsymbol{u}} \sum_{i=1}^{n} (a_i + 1) \|\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i\|^2 & (4.40)\\ &\text{s.t.} \quad \boldsymbol{u} \in \mathcal{B}^s(\boldsymbol{x}) \cap \mathcal{B}^c(\boldsymbol{x}, \mathcal{G}^c) & (4.41) \end{aligned}$$

A constant "1" is added to the coefficient a_i to avoid the lack of ranks in quadratic programming when all the requirements are fully satisfied.

4.4.1 Scaling in Quadratic Programming

Since the values and requirements of the tasks can be arbitrarily large as defined by users, the numerical stability of the quadratic programming solver can be drastically affected. To address

this issue, we need to scale the coefficients $\boldsymbol{a}$ to a proper magnitude. The coefficients are scaled as following,

$$\boldsymbol{a} = \boldsymbol{a}/(\max(\boldsymbol{a}) + \epsilon) * 10 \tag{4.42}$$

where ϵ is a small constant added in the denominator to avoid division by zero when all the needs are satisfied. Therefore, the coefficients $a_i + 1$ are scaled to be about the order of magnitude 1, which provides solutions more stably as shown in [35, 36].

4.5 Multi-Capability Generalization

One of the biggest advantages of this framework is that it is easy to be generalized with the multi-robot systems that have various capabilities. For instance, in a rescue mission, we may need to allocate different types of robots to different jobs like exploring, carrying loads, and defending against the enemies, with respect to their capabilities in various fields like sensing, mobility, load capacity, and etc. In this section, the generalization of the framework is presented to encompass more general multi-robot systems.

4.5.1 Generalized Problem Formulation

To accommodate the multi-capability generalization, we need to slightly revise the formulation. Suppose we are considering o capabilities in total. Now instead of a scalar, the required capability units of task j is noted by a vector $\boldsymbol{w}_j$ of size o. Similarly, the remaining requirements of task j is denoted as a vector $\boldsymbol{r}_j$ of size o. The other notations should remain the same. Now the objective function becomes

$$f(\mathcal{S}^*) = F(S^*) = \sum_{j=1}^{m} \left(v_j \left(\sum_{s=1}^{o} (\min(\boldsymbol{w}_{js}, \sum_{i \in \mathcal{A}_j} \boldsymbol{c}_{is})) \right) + \alpha \sum_{i \in \mathcal{A}_j} \frac{1}{1 + \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2} \right) \tag{4.43}$$

It is trivial to show that this revised objective function still owns the properties of submodularity and monotonicity. Thus, the quality of the solution from a greedy algorithm is still bounded as $f(\mathcal{S}^*) \geq \frac{1}{2} f(\mathcal{S}_{\text{opt}})$.

We will discuss about the revision for the algorithms in the next sections.

4.5.2 Adaptive Greedy Method for Multi-Capability Heterogeneous Robots

To accommodate the multi-capability formulation, the greedy algorithm is revised as in Algorithm 2.

A main amendment is the calculation of the gain g_{ij} and the loss l_{ij}, as they are the summation of the gains and losses of all the capabilities in every field. Also, when the requirements are updated, the whole vector is modified instead of one scalar.

Algorithm 2: Revised Greedy Algorithm

Result: The assignment matrix S

while $\mathcal{A}$ *is not empty* **do**
  for *robot* $i \in \mathcal{A}$ **do**
    for *task* j *= 0 :* m **do**
      $g_{ij} = v_j \left(\sum_{s=1}^{o}(\min(\boldsymbol{c}_{is}, \max(\mathbf{0}, \boldsymbol{w}_{js})))\right)$;
      if $j' = 0$ **then**
        $l_{ij} = 0$;
      else
        $l_{ij} = v_{j'} \left(\sum_{s=1}^{o}(\max(\mathbf{0}, \min(\boldsymbol{w}_{j's} + \boldsymbol{c}_{is}, \boldsymbol{c}_{is})))\right)$;
      end
      $h_{ij} = \frac{1}{1+\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2}$;
      $p_{ij} = g_{ij} - l_{ij} + \alpha h_{ij}$;
    end
    Find the maximal price p_i from task j_{best} to robot i or say marginal gain;
  end
  Collecting the reward p_i from all the robots;
  Choose robot i^* with the largest reward among all;
  if $p_{i^*} == \alpha$ **then**
    break ;
  else
    $j^* = j_{\text{best}}$;
  end
  $S_{i^*j'} = 0$;
  $S_{i^*j^*} = 1$;
  $\boldsymbol{w}_{j^*} = \boldsymbol{w}_{j^*} - \boldsymbol{c}_{i^*}$;
  $\boldsymbol{w}_{j'} = \boldsymbol{w}_{j'} + \boldsymbol{c}_{i^*}$;
  $\mathcal{A} \setminus i^*$;
  $\boldsymbol{w}_0 = \mathbf{0}$;
end

4.5.3 Redistribution among Multi-Capability Heterogeneous Robots

Recall that to generate controls, we solve the following Quadratic Programming Problem,

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \sum_{i=1}^{n} (a_i + 1) \|\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i\|^2 \tag{4.44}$$

$$\text{s.t.} \quad \boldsymbol{u} \in \mathcal{B}^s(\boldsymbol{x}) \cap \mathcal{B}^c(\boldsymbol{x}, \mathcal{G}^c) \tag{4.45}$$

This optimization formulation stays the same for multi-capability systems. Nonetheless, we calculate the coefficients a_i now as

$$a_i = v_j \sum_{s=1}^{o} \boldsymbol{c}_{is} \hat{\boldsymbol{r}}_{js} \tag{4.46}$$

$$= v_j \left(\sum_{s=1}^{o} \boldsymbol{c}_{is} (\max(0, \boldsymbol{w}_{js} - \sum_{i \in A_j} \boldsymbol{c}_{is} \sigma(k(L - d_i) + b))) \right) \tag{4.47}$$

Since here we have more capabilities, potentially the calculated coefficients grow larger. Thus, the scaling technique mentioned in Section 4.4.1 becomes more crucial.

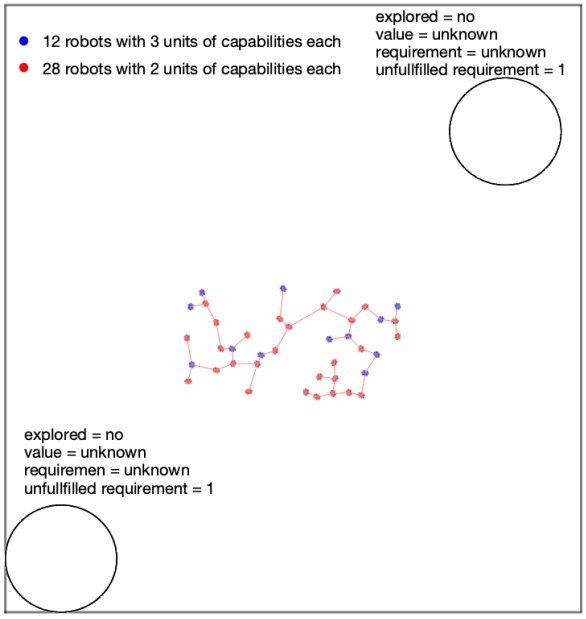
Chapter 5

Results

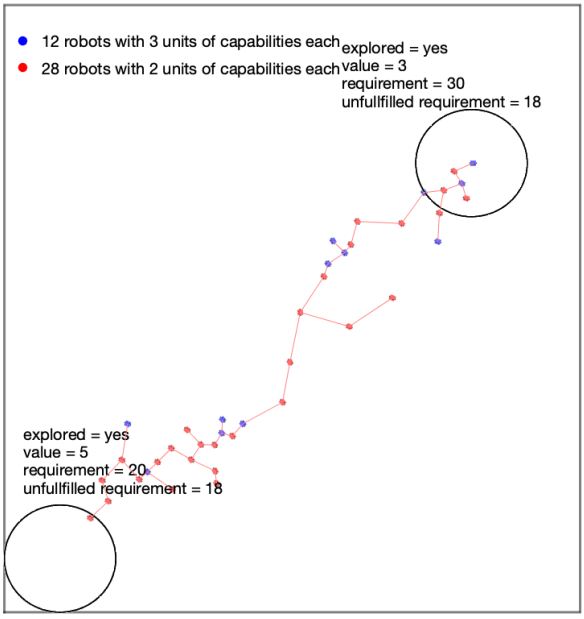
5.1 Simulation Result

Simulation experiments are shown here to illustrate how the multi-robot system can reconfigure itself adaptively along with the changing environment with this framework. For a more clear visualization without losing generality, in all experiments, only one capability is considered.

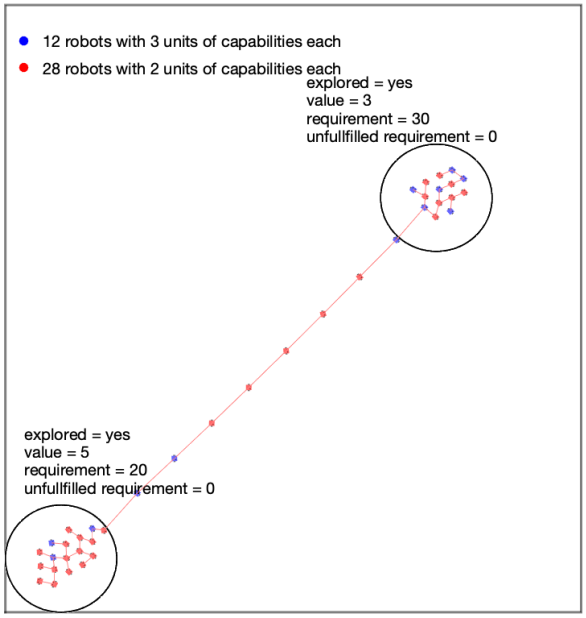
An example of simulation tests is demonstrated in Fig. 5.1. In this test case, the value of exploration is chosen to be 50, and the weight on traveling costs $\alpha = 0.5$. Two targets are present at the beginning. The top right target moves towards the center and the robots assigned to it should follow it in order to monitor the target. The task values and requirements are unknown and must be explored, as shown in Fig. 5.1a. We have 12 blue robots with 3 units of capability each and 28 red robots with 2 units of capability each, of which two robots are dispatched to explore the tasks. In the state represented in Fig. 5.1b, the information of both targets is revealed because at least one robot reaches the targets' visibility ranges. Task 1 needs 30 units of capability and its value is 3. Task 2 needs 20 units of capability with 5 as its value. The robot team is quickly reallocated based on the updated information to fulfill all the requirements. Because the number of robots is more than sufficient for both missions, both requirements are satisfied nicely as presented in Fig. 5.1c. In the next state shown in Fig. 5.1d, a new task with hidden importance and requirements pops up at the bottom right and one robot is reassigned to explore it. In spite of the connectivity constraints, the attraction of the exploration is strong enough to drag the robot along with its neighbors to the goal, as demonstrated in Fig. 5.1e. As the explorer acquires the knowledge of the new task, which requires 25 units of capabilities and is valued as 7, the system is rapidly redistributed and eventually fulfills all the requirements as presented in Fig. 5.1f.



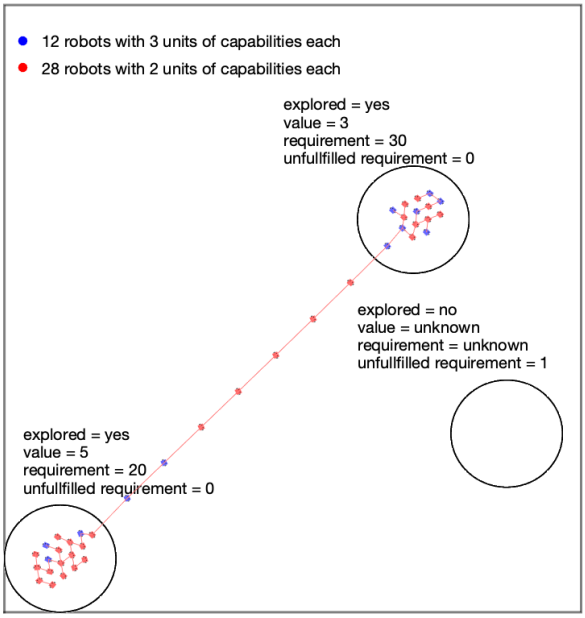
(a) Time Step = 0



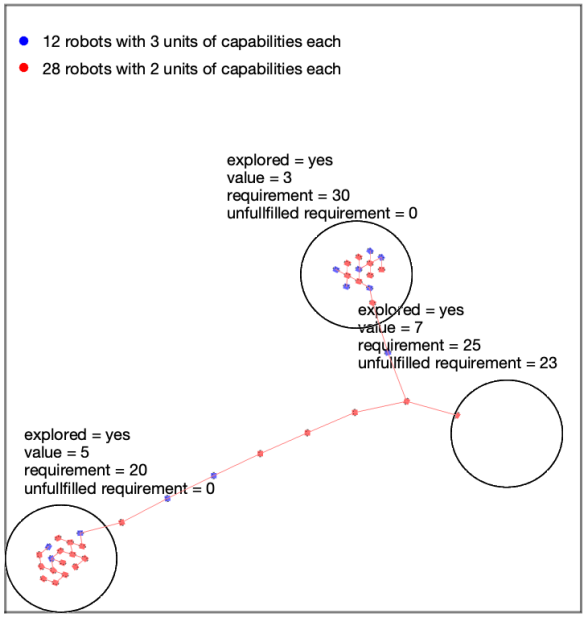
(b) Time Step = 370 (Both tasks explored)



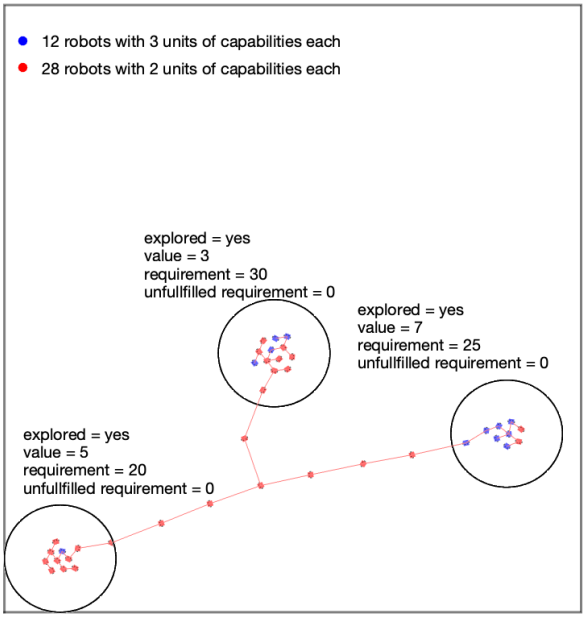
(c) Time Step = 650 (Both fulfilled)



(d) Time Step = 1000 (Third task appears)



(e) Time Step = 1620 (Third task explored)



(f) Time Step = 3270 (Final configuration)

Figure 5.1: Experiment 1: Simulation example of 40 robots of two types allocated to three different tasks: 12 blue robots with 3 units of capabilities each and 28 red robots with 2 units of capabilities each. In (a)-(c), only 2 targets are present. The robots are distributed to explore them firstly and redistributed based on the updated information. There are redundant robots in both tasks dragged by connectivity constraints, and they will be distributed to the new task appearing in the future. Task 3 pops up at time step = 1000 and (d)-(f) demonstrate the process of dispatching robots to explore task 3 and rearranging themselves to meet all the demands as shown in (f).

A naturally raised question would be, what if the number of robots is not large enough to cover all the requirements from the tasks plus satisfying the connectivity constraints? The second experiment here is to demonstrate that in such scenarios, this framework will drive the multi-robot system to the configuration that minimizes the unfulfilled requirements weighted by their importance.

In the second experiment, everything is set up exactly the same as in experiment 1, except the locations of the tasks are changed. Task 1 needs 30 units of capability and its value is 3. Task 2 needs 20 units of capability with 5 as its value. Task 3 requires 25 units of capabilities and is valued as 7. These numbers are unknown beforehand and shall be explored. We still have 12 blue robots with 3 units of capability each and 28 red robots with 2 units of capability each. However, in this experiment, as shown in Fig. 5.2, the three tasks are spread out further away from each other so that to maintain the connectivity, more robots are asked to serve as the connectivity nodes and fewer robots can actually execute their assigned tasks. We can observe from the figure that, in total 8 robots have to stay in the open space to reserve the connectivity. Because of the dynamic redistribution algorithm, the system is reconfigured so that only Task 1, which is the least important task among the three, lacks 2 units of the capability. The robots are allocated and distributed this way to minimize the remaining requirements weighted by tasks'

values, and thus maximizing the overall utility of the multi-robot system.

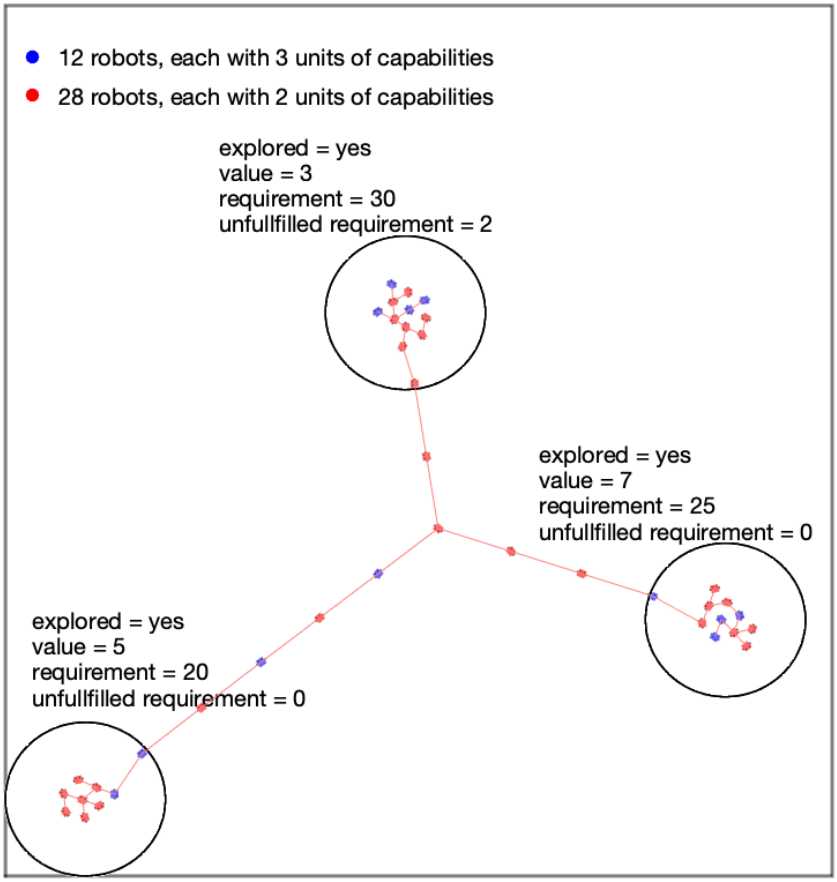
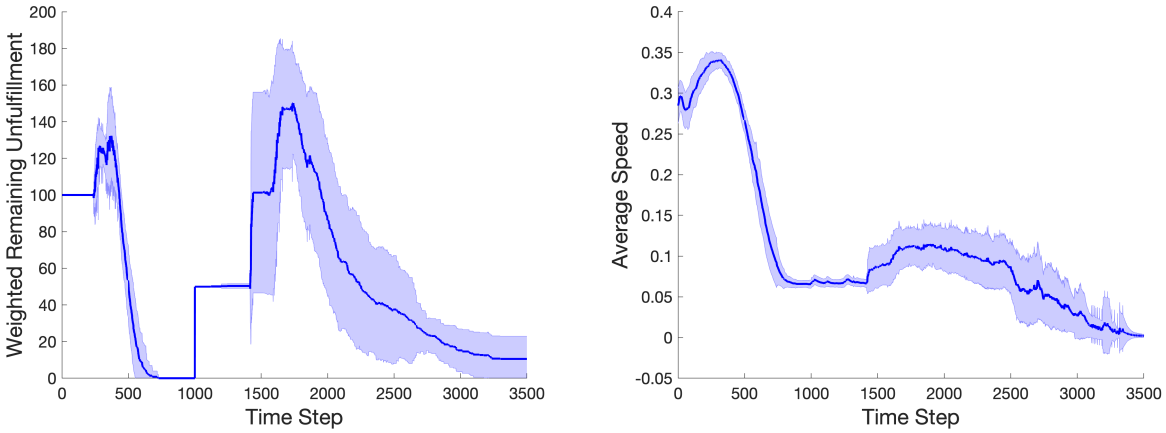


Figure 5.2: Experiment 2: The number of robots is not sufficient to cover all the requirements. The redistribution algorithm drives the multi-robot system to this final configuration so that the summation of the remaining requirement from each task weighted by its importance is minimized.

5.2 Numerical Result

To validate the reliability and efficiency of our method, we test the Experiment 1 mentioned in Section 5.1 for 20 times with different initialization of spawn locations. Because two robots are likely to be trapped in deadlock with the existence of collision avoidance and connectivity constraints when they are moving in the opposite direction, the robot team may not converge to the ideal final configurations within 3500 time steps. Figure 5.3a illustrates the trend of remaining requirements weighted by the importance of the tasks. From time step = 0 to 1000, the robot team is allocated to satisfy both requirements. At $t = 1000$, as a result of the appearance of a new task, we see the first bump. Between $t = 1500$ and $t = 1700$, the third task is reached and new demands are exposed, so the second bump is observed. Due to the various timings of arrival,



(a) Weighted remaining vs time

(b) Average speed (norm of velocity) vs time

Figure 5.3: Numerical results summary. In both figures, the solid lines represent the mean values and the shaded areas describe the standard deviations of the 20 experiments. (a) weighted remaining needs calculated by $\sum_{j=1}^{m} v_j r_j$. (b) Average speed of all robots computed by $\frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{u}_i\|_2$.

the variance during this time period is huge. After that, the weighted remaining unfulfilled requirements drop consistently and end with 7.8 averagely at $t = 3500$. Figure 5.3b represents the average speed profile of all robots, which is calculated as the mean of the norms of the robots' velocities. The whole robot team stays connected throughout all the experiments. At the end of the experiment when $t = 3500$, the average speed converges to 0 with significantly small variance, which indicates the whole crew reaches a steady state in most cases.

Chapter 6

Conclusion and Future Work

In this thesis, we present a task allocation method that dynamically distributes robots with different units of some capability in an uncertain environment where tasks appear dynamically, while considering connectivity maintenance, collision avoidance and heterogeneity of controllers. We proposed a greedy algorithm that takes traveling costs into considerations with provable optimality bound and a redistribution mechanism at the control level to mitigate the loss due to the connectivity constraints. We show experimental results to demonstrate the efficiency of our algorithm in maximizing the utility of the robot team and the convergence to steady states.

Future work will include scheduling by adding the temporal dimension in task allocation to further reduce the costs resulted from change of targets during traveling. Also, we can fully decentralize the algorithm so that the whole framework does not solely rely on one central hub and the robustness can be greatly improved.

Bibliography

[1] A. Prorok, M. A. Hsieh, and V. Kumar, "Fast redistribution of a swarm of heterogeneous robots," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, ser. BICT'15. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 249–255. [Online]. Available: http://dx.doi.org/10.4108/eai.3-12-2015.2262349 1.2, 2

[2] J. Liu and R. K. Williams, "Submodular optimization for coupled task allocation and intermittent deployment problems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3169–3176, 2019. 1.2

[3] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68 – 73, 2015, analysis and Design of Hybrid Systems ADHS. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S240589631502412X 1.2, 2, 4.3, 4.3

[4] W. Luo, S. Yi, and K. Sycara, "Behavior mixing with minimum global and subgroup connectivity maintenance for large-scale multi-robot systems," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020. 1.2, 2, 4, 4.3

[5] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004. [Online]. Available: https://doi.org/10.1177/0278364904045564 2

[6] 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. [Online]. Available: https://doi.org/10.1177/0278364913496484 2

[7] R. M. Zlot, "An auction-based approach to complex task allocation for multirobot teams," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, December 2006. 2

[8] B. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*. Springer, 2012, vol. 2. 2, 4.1

[9] J. Lee and S. Leyffer, *Mixed integer nonlinear programming*. Springer Science & Business Media, 2011, vol. 154. 2

[10] M. Yokoo and E. H. Durfee, "Distributed constraint optimization as a formal model of partially adversarial cooperation," 1991. 2

[11] J. P. Pearce and M. Tambe, "Quality guarantees on k-optimal solutions for distributed con-

straint optimization problems." in *IJCAI*, 2007, pp. 1446–1451. 2

[12] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg, "Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks," *Artificial Intelligence*, vol. 161, no. 1, pp. 55 – 87, 2005, distributed Constraint Satisfaction. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370204001481 2

[13] V. Lisy, R. Zivan, K. Sycara, P. M, V. Lisy, R. Zivan, K. Sycara, and P. M, "Deception in networks of mobile sensing agents," in *9th Internationnal Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada*, May 2010, pp. 1031–1038. 2

[14] Z. Duan, W. Li, and Z. Cai, "Distributed auctions for task assignment and scheduling in mobile crowdsensing systems," in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 635–644. 2

[15] G. Notomista and M. Egerstedt, "Constraint-driven coordinated control of multi-robot systems," in *American Control Conference (ACC)*. IEEE, 2019, pp. 1990–1996. 2

[16] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *18th European Control Conference (ECC)*. IEEE, 2019, pp. 2071–2076. 2

[17] A. Prorok, M. A. Hsieh, and V. Kumar, "Formalizing the impact of diversity on performance in a heterogeneous swarm of robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5364–5371. 2

[18] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by reconfiguration: Exploiting heterogeneity in robot teams," *arXiv preprint arXiv:1903.04856*, 2019. 2

[19] A. Halász, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic redistribution of a swarm of robots among multiple sites," in *IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2007, pp. 2320–2325. 2

[20] M. M. Zavlanos, A. Jadbabaie, and G. J. Pappas, "Flocking while preserving network connectivity," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2919–2924. 2

[21] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011. 2

[22] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007. 2

[23] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013. 2

[24] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks,"

Automatica, vol. 46, no. 2, pp. 390–396, 2010. 2

[25] R. K. Williams, A. Gasparri, G. S. Sukhatme, and G. Ulivi, "Global connectivity control for spatially interacting multi-robot systems with unicycle kinematics," in *IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 1255–1261. 2

[26] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 2659–2664. 2, 4.3

[27] W. Luo and K. Sycara, "Voronoi-based coverage control with connectivity maintenance for robotic sensor networks," in *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 148–154. 2

[28] ——, "Minimum k-connectivity maintenance for robust multi-robot systems." 2

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. 2

[30] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, *An analysis of approximations for maximizing submodular set functions—II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 73–87. [Online]. Available: https://doi.org/10.1007/BFb0121195 4.1, 4.2

[31] R. K. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 293–300. 4.1

[32] H.-S. Shin, T. Li, and P. Segui-Gasco, "Sample greedy based task allocation for multiple robot systems," *arXiv preprint arXiv:1901.03258*, 2019. 4.1

[33] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005. 1, 2, 4.2

[34] M. Abramowitz, I. A. Stegun, and R. H. Romer, "Handbook of mathematical functions with formulas, graphs, and mathematical tables," 1988. 4.4

[35] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006. 4.4.1

[36] D. R. Herber and J. T. Allison, "Unified scaling of dynamic optimization design formulations," in *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2017. 4.4.1