Decentralized Persistent Connectivity Deployment in Robot Swarms

by

Adhavan Jayabalan

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Masters of Science

in

Robotics Department

by

J. Adhavan

May 2018

APPROVED:

Dr. Carlo Pinciroli, Advisor

Dr. Jane Li, Committee Member

Dr. William R. Michalson, Committee Member

Abstract

Robot swarms are often considered suitable for tasks that are large-scale and long-term. Large-scale missions force the robots to spread spatially. In these type of tasks, actively maintaining connectivity allows the swarm to coordinate. Similarly, long-term nature of the task requires robots to work for a long time. This is affected by the limited energy level of the robot. However current studies normally focus only on connectivity or energy awareness. Therefore, in this work, we propose an approach to tackle the problem of maintaining global connectivity (swarm-level property) considering finite battery life (individual property). We are specifically focusing on growing the communication network and keeping it alive for a long period. We construct a logical tree over the connectivity graph. The logical tree is constructed by using a subset of robots from the swarm. The tree is grown by adding robots as necessary. The tree is also periodically reconfigured to cope with dynamic robot motion. This enables the swarm to grow the tree efficiently. In addition, robots exchange their roles based on their available energy levels. This allows robots with low energy levels to navigate to dedicated charging stations for recharging thus allowing the swarm to maintain the communication network. We evaluate our approach in a wide set of experiments with a realistic robot simulator named ARGoS.

Acknowledgment

I would like to express my gratitude to my advisor Prof. Carlo Pinciroli for his guidance, direction, and support for the duration of this work. I would like to thank Prof. Jane Li and Prof. William R. Michalson for being part of the committee. I would like to thank Nathalie Majcherczyk for her guidance and support throughout the project. Lastly, I would like to thank my family and friends for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

In the field of robotics, coordinating a large number of robots to accomplish different tasks is a research problem that has gained attention recently. Swarm robotics [1] is a field that approaches this problem in a decentralized manner. Swarm robotics draws inspiration from the collective behaviors of various species such as ant-colonies, fish schools, and bird flocks. Simple local interactions between individuals are responsible for the emergence of collective behaviors. Robot swarms can be used for large-scale tasks such as search and rescue, construction, demining, exploration and mapping where using a single robot is time-consuming and inefficient.

Figure 1.1: Examples of natural swarm

However, preferring only local interactions causes robots in the swarm to be unaware of the global information. Robot swarms can perform basic tasks like aggregation [2] without

explicit information exchange. However, complex tasks such as construction or demining require frequent information exchange for efficient task completion. Another important aspect that affects the performance of the swarm is energy. The useful mission time of any robot is limited by how efficiently it uses its available energy. Robots are forced to abandon a mission when energy levels are critical low.

1.2 Project Brief

The problem statement we considered in this thesis is to deploy and maintain a communication network in the robot swarm while accounting for the energy limitation of the individual robots. The robot swarm is required to reach a number of distant task locations without any disconnection in the network. Once the connections are deployed, there should be a path for the information to flow between any two robots. The network should be persistently maintained despite the fact that robots lose energy over time.

The problem considered in this work is interesting because it combines two types of constraints that are mutually contradictory. Connectivity is a swarm-level property, whose maintenance requires the long-term cooperation of as many individuals as possible in the swarm. Conversely, energy limitations force individual robots to abandon their duties to recharge. Under this light, it is apparent that achieving long-term connectivity is a difficult problem, whose solution would enable swarm applications that are today impossible.

When compared to recent studies in connectivity maintenance, two aspects set our work apart from others. First, we used only part of the swarm to achieve global connectivity. By doing this, we can use the remaining robots for other tasks. Second, considering two conflicting constraints connectivity and energy limitation together.

1.2.1 Objectives

Objectives of this thesis are:

- Develop a strategy to enable long-term swarm survival despite battery depletion
- Manage charging stations to recharge low energy robots.

1.3 Outline

The rest of this thesis is organized as follows: In Chapter 2, we discuss recent works associated with connectivity maintenance, and energy-awareness in robot swarm. In Chapter 3, we formalize the problem description. In Chapter 4, we present our approach to the problem. In Chapter 5, we display the experimental setup and the performance of our approach under different conditions. In Chapter 6, we conclude the thesis and contemplate the future work.

Chapter 2

Related Work

In this chapter, we present recent works about deploying and maintaining connectivity in robot swarms. Additionally, we review current literature pertaining to energy awareness in multi-robot systems. Both topics will be tied together in the following chapters.

2.1 Connectivity

Connectivity is the ability of the robots to exchange information whenever required. In robot swarms, connectivity is essential to enable coordination between robots. By exchanging information, robots will be aware of the situation. Different approaches have been explored to solve the problem of connectivity.

Williams et al. [3] estimated the algebraic connectivity in a decentralized way and defined a control law that aims at maximizing connectivity. They tested the algorithm when the swarm is performing aggregation and leader following.

Cornejo et al. [4] developed an algorithm that modifies the plans provided by any motion planner to preserve connectivity in the swarm. Their approach is split into three phases: the collection, proposal and adjustment phases. In the collection phase, the robots query their current position and plan. In the proposal phase, the swarm comes to consensus on what edges need to be preserved to maintain connectivity. Finally, in the adjustment phase, robots check if they can reach the target location and if so, they execute the motion. Hsieh et al. [5], developed a reactive controller for maintaining communication links. They estimated the link quality using radio signal strength and data throughput. Each robot

will react to the changes in the observed communication link quality around its neighbor. Sabattini et al. [6] estimated algebraic connectivity using power iteration method [7] for maintaining global connectivity. In [4–6] uses the idea of preserving only selected edges of the graph to maintain global connectivity.

Due to uncertainties in the environment, the swarm might fail to preserve connectivity. Such failures lead to degradation in performance, losing robot in the environment and even failure of the tasks assigned to the swarm. Ghedini et al. [8, 9] improved robustness in connectivity under failures. They proposed a mechanism which allows the individual robots to estimate the probability of being in a harmful configuration. Further, they defined a local control law strategy that is robust to connectivity failures. The control law defines proactive and reactive mechanisms which prevent robots to the vulnerable topology and accommodates the ongoing failures respectively. Battagello et al. [10, 11] estimated algebraic connectivity to analyze the effects of communication failure and communication delay in the distributed multi-robot system. They used the control strategy developed by Sabattini et al. [12]

Summary

Authors	Approach	Strengths	Limitations
Williams et al. [3]	Fiedler value estimation	- Collision avoidance - Tested for aggregation and leader following	- Tested with ten robots
Cornejo et al. [4]	Local connectivity by preserving edges	- Works with any motion planner	- Not scalable
Hsieh et al. [5]	Used radio signal strength & data throughput to maintain connectivity	-Real robot validation	- Tested with four robots - Not scalable
Sabattini et al. [6]	Fielder value estimation using power iteration	- Preserved only selected edges	- Tested with eight robots
Ghedini et al. [8, 9]	Probabilistic estimation of harmful configuration	- Improved robustness - Tested with 100 robots	- Simulation
Battagello et al. [10, 11]	Fielder value estimation	- Analyzed effects of communicationh failure	- Simulation

Table 2.1: Summary: Literature related to connectivity

2.2 Energy Awareness

Recently, there has been a wide interest in studying energy constraints in robot swarms. By considering this energy limitation, the robot swarms are able to revise their decision to mitigate the failures caused by limited battery life. Literature presented in this section can be categorized into two groups.

In the first group, the swarm has a limited energy resource and studies involve optimizing energy spent. Setter et a.l [13, 14] divided the rendezvous problem into two parts. First, the robot team decides the rendezvous based on the individual's energy levels. Second, each robot realizes an optimal motion that minimizes power consumption. Liu et al. [15] devised an algorithm that allows the individual robot in the swarm to optimize the swarm energy in the foraging task. Food collected by the robots represent a limited energy that robot can use to gather more food. Each robot uses cues that are internal (successful food retrieval), social (teammates success in food retrieval) and environmental (collisions with teammates while searching for food) to change their role between foraging and resting.

In the second group, charging stations are present as a part of the problem. This group of literature focuses on continuing the tasks assigned to the swarm persistently. Kevin et al. [16] developed an automated persistent surveillance system. They positioned charging stations across the environment to recharge robots regularly. They generated a control policy based on the constraints defined using bounded linear temporal logic (BLTL) to accommodate charging time and deadlines for visiting a region. Guannan et al. [17] developed an energy-aware decentralized algorithm to form a desired pattern using robot swarms. The low energy robots exchange their position with their neighbors until they reach the charging stations in the formed pattern. The potential application of pattern formation includes infrastructure support during search and rescue and exploration. It should also be noted that these approaches were tested with only 10 robots in the swarm. The authors have not discussed the scalability of the algorithm to a larger group of swarms.

Bjerknes et al. [18] performed a case study on different types of failures by the robots in the swarm. The failures include agent deaths and sensor/actuator failures. These failures can be due to low energy levels in the robots. In this study, robot swarms are required to reach a goal location. It is understood that actuator failure prevents the swarm from reaching its goal because of the anchoring effect exerted by the failed robots. This shows

the importance of the charging stations in the environment to prevent these failures.

Summary

Authors	Approach	Strengths	Limitations
Setter et.al [13, 14]	energy constrained optimization for coordination	- Tested for rendezvous	- Tested with three robots
Liu et al. [15]	role adaptation based on cues	- Robust to environmental changes	- Tested with ten robots
Kevin et al. [16]	Temporal Logic for surveillance	- Persistent surveillance - Real robot validation	- Tested with three robots
Guannan et al. [17]	Pattern formation using point cloud data	- Considered network coverage	- Simulation
Bjerknes et al. [18]	Case study on failures	- Tested with real robots	- Tested only for flocking

Table 2.2: Summary: Literature related energy awareness

Chapter 3

Problem Statement

This chapter frames the problem studied by discussing our main objectives and assumptions. It includes a formal description of the system dynamics, robot battery model and robot communication model.

3.1 Objectives

We aim to design a strategy to deploy and grow a connected network of robots with realistic energy limitations. We consider a scenario in which the robot swarm has to reach and service multiple tasks. These tasks are spread out in space. An important aspect of our problem is to ensure that the system persists over time despite depleting batteries on individual robots. Therefore, our objectives can be stated as follows:

- Study and help characterize the performance of the deployment strategy presented in [19];
- Develop an extension of the deployment strategy to enable long-term swarm survival despite battery depletion;
- Manage charging stations to recharge low energy robots in accordance to the above mentioned strategy.

3.1.1 Assumptions

We make the following assumptions throughout our work:

- **2D Space**: Robots are interacting in the 2D Euclidean space and their position and orientations are represented by x, y and θ.
- **Initial Condition**: Robots are initially deployed in a tight cluster.
- **Task Allocation**: Assigning tasks for the each robot is done by some external modules.
- **Communication**: Each robot is equipped with a range bearing sensor that measures the relative position and orientation of the other robots and transmits information to neighboring robots within the communication range C.
- **Battery**: The battery charges and discharges at a defined rate across the experiments.
- **Obstacle free environment**: The robot swarm operates in an environment without any obstacles.

3.1.2 Constraints

We consider the following constraints:

- **Decentralized**: The algorithm should be completely decentralized
- **Energy**: Experiments are considered failure if any of the robot runs out of energy, even if they are not part of the communication graph

3.2 Robot Dynamics

We considered N robots modeled as,

$$\dot{x}_i = u_i$$

where, $x_i \in \mathbb{R}^M$ is the position of i^{th} agent in the swarm. $u_i \in \mathbb{R}^M$, is the control input given to the robot i. In this work, we tested our approach (discussed in Chapter 4) in a 2D space hence the state vector x_i is defined as $[p_x, p_y, \theta]$. Modeling agents as single integrators simplifies the vehicle's dynamics by treating them as a point-mass object.

3.3 Battery Model

To model the energy constraints of a robot swarm, we created a battery model with an energy level ranging from zero to one. Robots in the swarm dissipates energy as they move (i.e., change in position and orientation). To account for factors like sensing and communication, the robot loses a a constant amount of energy at every time step as shown in Eq. 3.1.

$$e_{new} = e_{old} - \lambda_t - \lambda_{pos}(\Delta_{pos}) - \lambda_{orient}(\Delta_{orient}) \tag{3.1}$$

where e_{new}, e_{old} are the battery levels at current and previous time steps. $\lambda_t, \lambda_{pos}, \lambda_{orient}$ are the time, position, and orientation factors respectively. $\Delta_{pos}, \Delta_{orient}$ are the changes in position and orientation respectively.

Whenever the robots are within a certain distance from the charging station, the batteries are charged at a constant rate as shown in Eq. 3.2.

$$e_{new} = e_{old} + \eta \tag{3.2}$$

where η is the charging rate per time step. Table 5.1 lists the values chosen for the parameters in Eq. 3.1 and Eq. 3.2.

3.4 Robot Communication and Interactions

3.4.1 Situated Communication

We assume robots can perform situated communication (refer Fig. 3.1). This type of communication allows robots to broadcast messages to their neighbors within the communication range C. Robots receiving messages can locate the relative position of the senders. The advantage of situated communication is that messages are no longer need to contain metadata like sender's id and location.

The communication region of each robot is a disc shape. We divided the communication region into different sub-regions based on the radial distance from the robot. They are avoid region, safe region, and target region as shown in Fig. 3.2. The type of interaction between each robot in the neighborhood is described in Table 3.1.

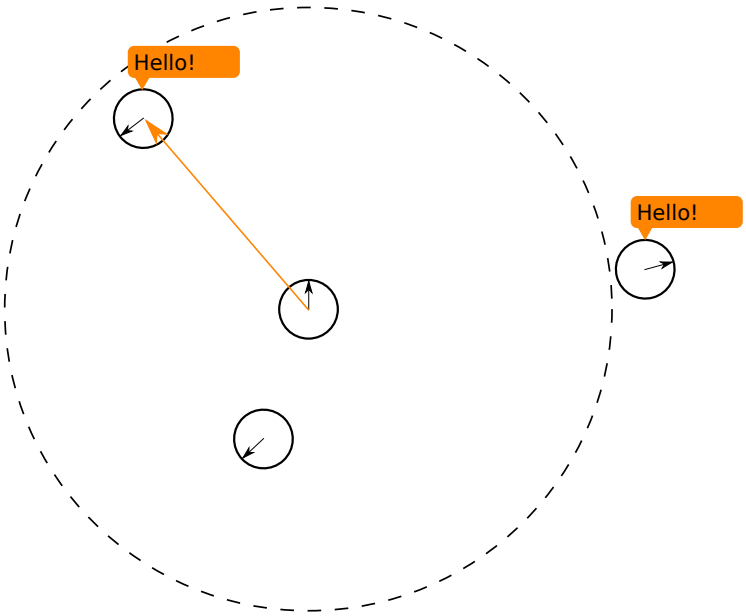

Figure 3.1: Situated Communication

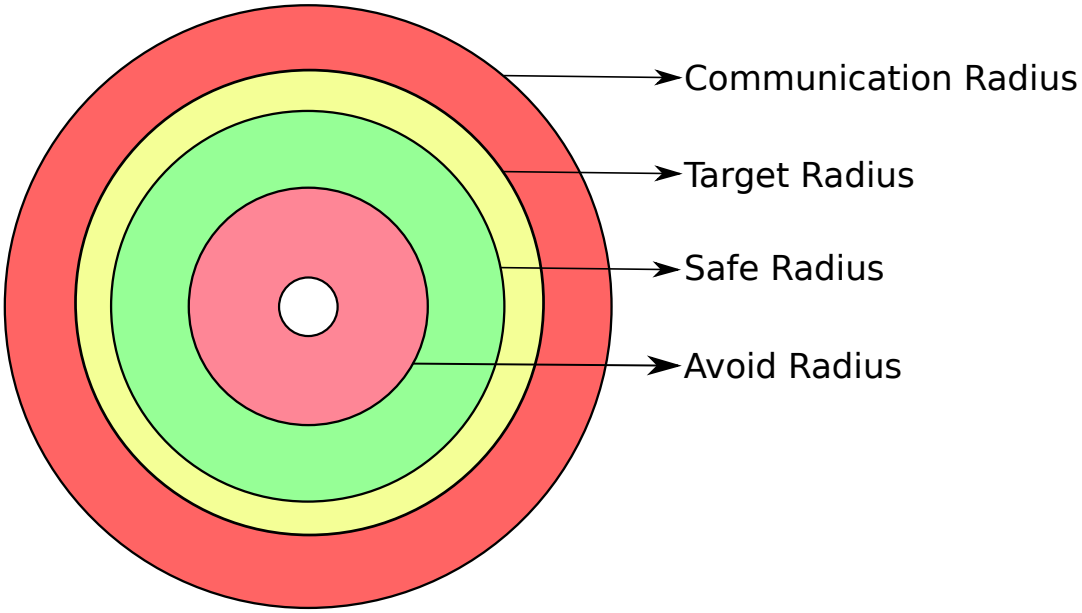

Figure 3.2: Communication model

3.4.2 Lennard-Jones Potential

The magnitude of interactions between two robots is obtained using the Lennard-Jones potential. Lennard-Jones potential (refer Eq. 3.3) is a simple model for particle interaction where the magnitude and type of force depends on the distance between them. It consists

Region	Interaction
Avoid	Robot moves away from parent (repulsion)
Safe	Robot circulates the parent (centripetal)
Target	Robot moves towards parent (attraction)
Beyond Target	

Table 3.1: Robot Interaction

of a steep repulsive force and a smooth attractive force as a function of a distance between two objects as shown in Fig. 3.3.

$$V(x) = \epsilon\left(\left(\frac{\delta}{x}\right)^4 - 2\left(\frac{\delta}{x}\right)^2\right) \tag{3.3}$$

where, ϵ is the depth of the potential well, δ is the distance at which particle interaction is zero and x is the distance between two interacting robots. By differentiating Eq. 3.3 we obtain the magnitude of the virtual force acting between the robots in the swarm.

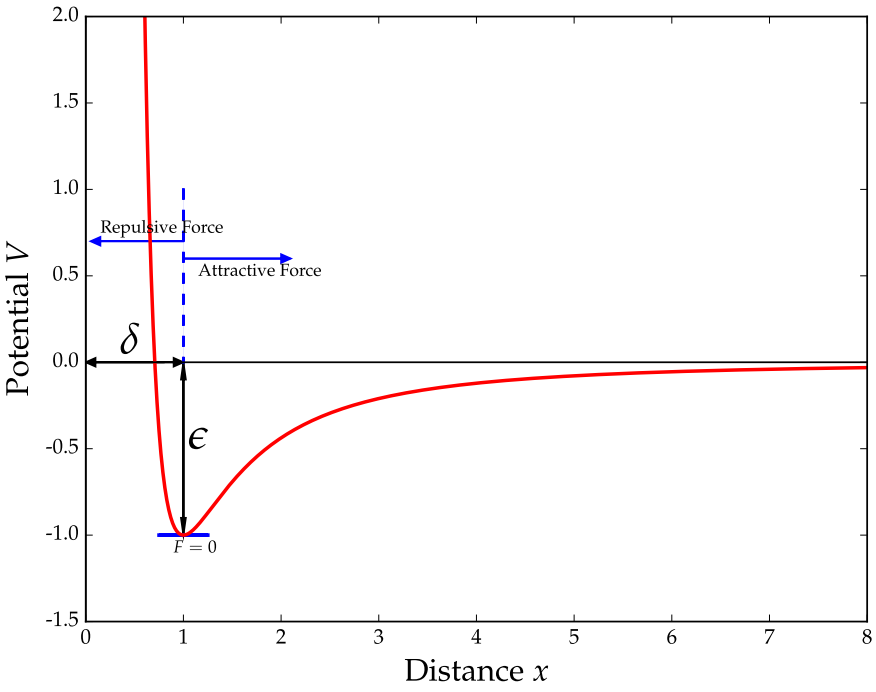

Figure 3.3: Lennard-Jones Potential

3.4.3 Communication Graph

The communication topology of the robot swarm can be represented as a graph $G = (V, E)$, where V is the set of nodes representing robots in the swarm and E represents edges connecting the robots. At any given time t, an edge (i, j) exists only if the distance between the robots connected by that edge is within the communication range C.

A graph is considered to be connected if there exists a path between any two robots in the swarm. The graph connectivity can be quantified by the second smallest eigenvalue of laplacian matrix L. Laplacian matrix L is defined as $L = D - A$ where A is the adjacency matrix and D is the degree matrix defined as follows:

$$A_{ij} = \begin{cases} 1 & \textit{if } (i,j) \in E \\ 0 & \textit{otherwise} \end{cases} \tag{3.4}$$

$$D_{ij} = \begin{cases} \sum A_{ij} & \textit{if } i = j \\ 0 & \textit{otherwise} \end{cases} \tag{3.5}$$

If the eigenvalues of the matrix L are arranged in increasing order, $\lambda_1 \leq \lambda_2 \leq \lambda_3 \cdots \leq \lambda_n$

- Smallest eigenvalue λ_1 is always zero
- Second smallest eigenvalue λ_2 is greater than zero only if the graph is connected

The second smallest eigenvalue λ_2 is known as algebraic connectivity or Fielder value.

Chapter 4

Methodology

In this chapter, we present our work in designing a swarm behavior that deploys a connected network of robots while recharging robots with critically low energy levels. The connected deployment strategy is a result of the work presented in [19] (see Appendix A) and we describe it in section Sec. 4.2. We discuss the extended behavior including realistic energy constraints in section Sec. 4.3.

4.1 Roles

Each robot in the swarm is assigned to either of the two roles: working or spare robot. Working robots are part of the connected tree. Whereas, spare robots are not involved in the tree. Further, working robots can take one of the three roles: root, worker or connector as shown in Fig. 4.1. The root connects multiple branches together and at any given time only one root can exist. Worker robots are the tree leaves that move towards the target location, forcing the tree growth. The connector robot are the tree nodes between the tree leaves and the root. These robots leave the pool of spares to join the tree helping the swarm in growing the tree. Spare robots are the remaining robots in the swarm that are not part of the tree. They become a connector or replacing a working robot whenever required. Apart from these robots, one special type of robot that is not part of the swarm is the charging station. These are robots that recharge the robots in the swarm whenever required. We assume that workers and charging stations have unlimited energy.

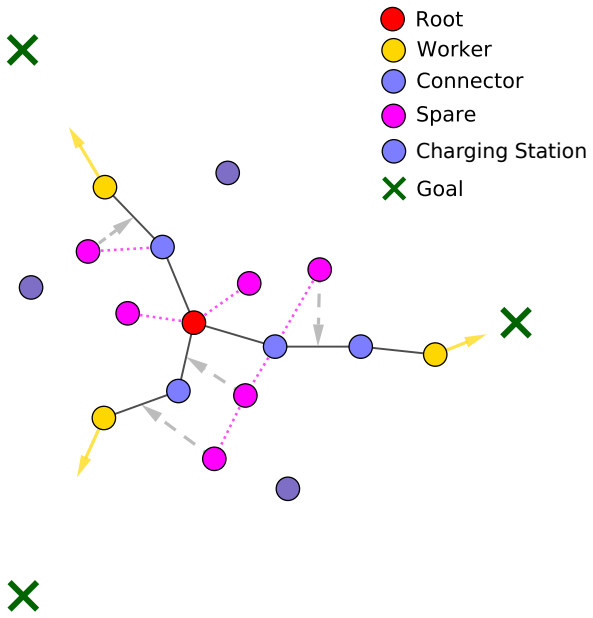

Figure 4.1: Different Roles

4.2 Connectivity

In this section, we present our approach to deploy and maintain connectivity without considering energy parameter. In Sec. 4.2.1 we discuss the high level behaviors of the swarm and in Sec. 4.2.2 we present the local interaction between robots.

4.2.1 High-level Behavior

We start with a tight cluster of robots. The root and workers are assigned by an external module (e.g., task allocation algorithm). The INIT state is used to serve this purpose. The closest robot to the centroid of the cluster is selected as the root. The robots closer to the target locations are assigned the role of worker. Remaining robots become spare by default. The root robot signals the swarm to proceed with the START TREE state, informing the swarm to form a new tree. The purpose of forming a new tree is to have an updated tree that is suitable to the current robot positions. When robots receive the trigger, they compute their relative distances to the root.

Once the relative distance to the root is estimated, robots proceed to the SELECT PARENT state to select a new parent. The parent should be within communication range C and closer to root. The parent selection process is started by workers and end with the root.

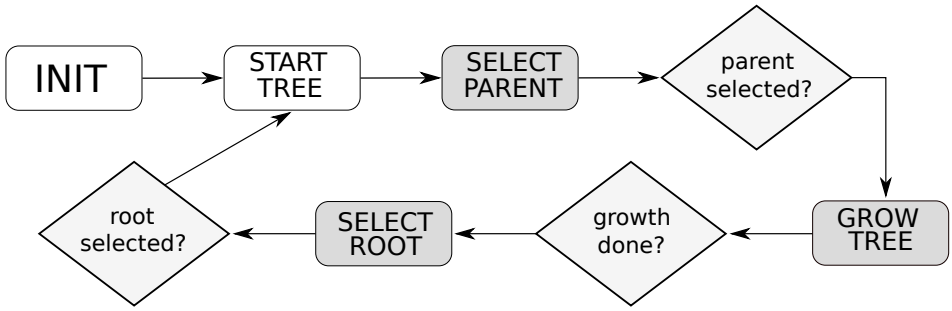

Figure 4.2: State Machine

In this process, robots form a shorter path between workers and root when compared to the previously grown tree. Robots coexist in both old and new tree until the new tree is completely formed. Once robots switch to the new tree, spare robots chooses the closest working robot as their parent.

Robots proceed to the GROW TREE state after choosing their parent. In this state, spare robots can join the tree to help worker robots reach their target location. After a predefined period, the root triggers state transition to allow robots to switch to the SELECT ROOT state, allowing the swarm to select a new root. This makes the tree topology evenly distributed. Finally, the newly assigned root signals swarm to proceed with START TREE state. The root selection process is discussed detail in Sec. 4.2.3.

State transitions to START TREE, GROW TREE and SELECT ROOT states happen simultaneously in robots only when the root satisfies those conditions. This synchronization is archived through a barrier-trigger mechanism, where a barrier is a "wait state" preventing robots transitioning to next state until the trigger from the root is received.

4.2.2 Spare Management

Robots in the swarm move only when they are in the GROW TREE state, this is to allow robots to do distance estimation and parent selection process properly. The state machine diagram shown in Fig. 4.3 describes the interaction between working and spare in the GROW TREE state.

Working robots start in the NO NEED state, they can extend the edge with their parent without an additional robot. They switch to the NEED state whenever any of their children

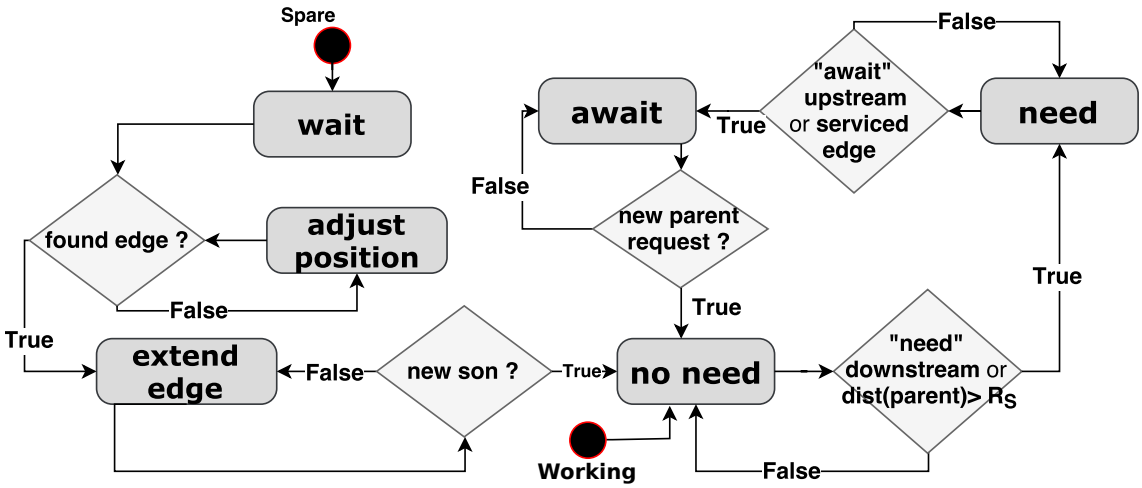

Figure 4.3: Sub State Machine

are in the NEED state or distance to their parent is more than the safe radius S. When robots are in the NEED state, they check if the spare robot is joining the edge they are present in or if the parent is in the AWAIT state. Robots switch to the AWAIT state whenever either of the conditions are satisfied. Robots in the AWAIT state switches to NO NEED state once the spare robot joins the tree.

Spare robots start in the WAIT state, looking for an edge to extend in the neighborhood. If the spare robot finds an edge, it switches to EXTEND state to extend the tree. If they do no find an edge, spare robots enter the ADJUST state, to circumvent around the parent if it is within safe radius else move straight towards the parent.

4.2.3 Root Selection

Selecting a new root for the tree allows us to balance the uneven distribution of robots in the swarm. Balancing the tree helps to reduce the number of robots between worker and root. This is critical when comes to time complexity because some states described in the Sec. 4.2.1 uses diffusion or aggregation of information whose time complexity is linear to the depth of the tree. Processes that involve diffusion of information includes distance estimation and barrier-trigger mechanism. Processes that include aggregation of information includes centroid calculation and counting algorithm (discussed later in this section). If the depth of the tree is high, it takes more time for the robots away from the root to obtain/process the information. Thus, by choosing a new root, we are reducing the

Algorithm 1 Distributed centroid estimation algorithm executed by robot i: $\mathbf{a}_i$ denotes an accumulator value; $\mathbf{q}_i$ denotes the contribution of robot i to the estimation algorithm; c_i and d_i denote the number of robots in the swarm estimated by robot i and the tree depth of robot i, respectively; and $\mathbf{p}_i^{\text{parent}}$ is the vector from robot i to its parent.

$\mathbf{a}_i = 0$
for all child j **do**
    $\mathbf{q}_j^i$ = express $\mathbf{q}_j$ in i's reference frame
    $\mathbf{a}_i = \mathbf{a}_i + \mathbf{q}_j^i$
end for
if robot i has a parent **then**
    $\mathbf{q}_i = \mathbf{a}_i - (\underbrace{c_i - d_i}_{\text{nb descendants}} + 1) \cdot \mathbf{p}_i^{\text{parent}}$
end if
if robot i is the root **then**
    $\mathbf{q}_i = \mathbf{a}_i / \underbrace{c_i}_{\text{robot count}}$
end if

tree depth and time to complete the diffusion or aggregation process.

Based on the above discussion, we can say that the robot closer to the centroid of the swarm will be an ideal candidate for the root role. The new root is selected based on the centroid and count obtained in the SELECT PARENT state. The Alg. 1 describes how the centroid is estimated. As the robots are only aware of their relative position, each robot accumulates the contribution of their children and themselves and passes it to their parent as a vector in their own reference frame. By repeating this, the root obtains the total contributions made by the swarm. Now to calculate the centroid, the root needs the count of robots in the swarm as shown in the Alg. 2. Algorithm defined in Alg. 2 counts the number of robots in the tree, where each robot count the number of children and passes that to their parents. In SELECT ROOT state, the current root estimates the distance to the centroid and if it the closets to the centroid, the swarm proceeds with the START TREE state. If any other robot is closer to the centroid than the current root, the root elects this robot as the new root. The newly selected root acknowledges it and repeats this process until the closest robot is chosen as the root.

Algorithm 2 Tree-based count algorithm for robot i. The depth of robot i in the tree is denoted as d_i. The depth of the tree root is set to 1. The count calculated by robot j is denoted as c_j.

switch number of children **do**
    case 0
        return d_i
    case 1
        return c_{child}
    default
        return $\sum_{\text{neighbors } j}(c_j - d_i) + d_i$
end switch

4.3 Energy awareness

In this section, we consider energy constraint to our deployment strategy discussed in the previous section. First, we present how the charging stations work and in the later part, we discuss the behaviors that are modified to satisfy the energy constraint in our connectivity maintenance problem.

4.3.1 Charging Stations

Charging stations are the special robots that are not part of the swarm and are assumed to have unlimited energy. These charging robots recharge the robots in the swarm whenever needed. The charging stations are deployed in the area between the tasks as shown in Fig. 4.4.

Charging stations have three states: PATROLLING, MOVING towards incoming robots and CHARGING as shown in the Fig. 4.5. At the start of the experiment, charging stations start in the PATROLLING state as they move between predefined checkpoints. By patrolling, charging stations make themselves available to a large number of robots in the swarm. When the incoming spare robots are within the communication range of the charging station, they move towards the energy-based centroid of all the incoming robots. Once all the robots are close enough, the charging station will switch to charging state till all the robots are charged.

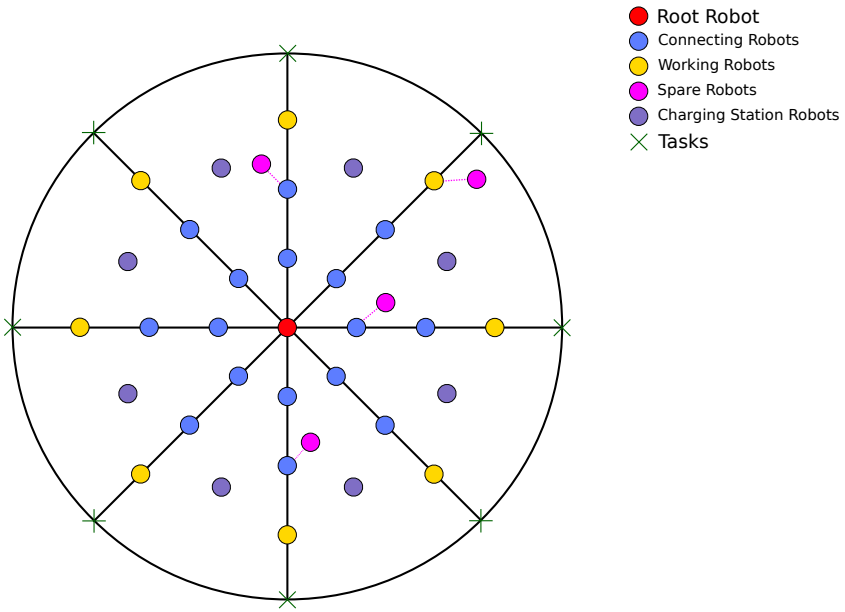

Figure 4.4: Initial position of charging stations

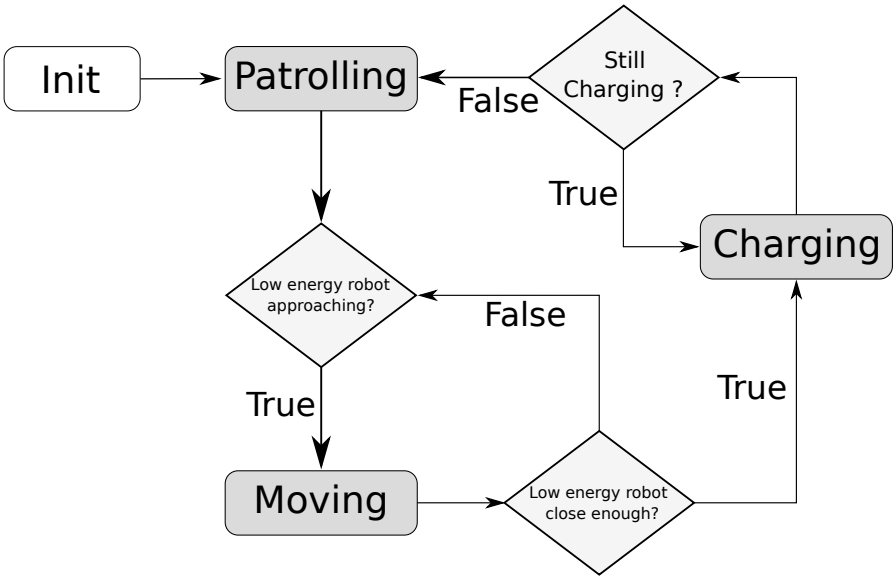

Figure 4.5: Charging Station State Machine

4.3.2 High-Level Behavior

To maintain the tree connectivity for a long period, robots in the tree should have high energy levels. We need to include the robot energy levels as a factor in the parent selection and root selection processes. This allows building a tree that can be connected as well as maintained for a longer period. In the parent selection process, the robots that are closest

to the root are taken and their energy levels are compared to respective communication regions shown in Fig. 3.2.Similarly, in root selection process, the robot closer to the centroid is chosen only if it has higher energy level than the current root.

4.3.3 Replacing phase

In this phase, low energy working (L.E.W) robot, children of L.E.W of low energy working robot and a spare robot are involved. The sequence of actions taken by these robots is shown in Fig. 4.7 as a sequence diagram.

Whenever the energy of a working robot is below a certain threshold, it broadcasts messages (like the SOS signal) to the neighboring spare robots requesting for a replacement. Spare robots respond with their availability upon receiving the message. The low energy working robot gets to choose one of the available spare robots to replace it. As the replacing spare robots are close enough, low energy working robots inform their children and parent about exchanging roles. After informing they leave the tree allowing new working robot (replacing spare robot) to join. Once the new working robot joins the tree, the low energy robot heads towards an approximate charger location. The charger location will be corrected when the robot see one.

Low Energy Working Robot
Whenever the available spare robots are in the neighborhood of a low energy robot, they choose a spare with a higher energy level. Choosing the robot with high energy level for substitution avoids the new working robot to become a low energy robot. Once the low energy working robot selects the spare robot, it broadcasts a message to that spare robot requesting for substitution. If the spare robot did not acknowledge this message within a certain time, the low energy working robot selects another available spare robot.

When the selected spare robot comes closer, the low energy working robot will move in the opposite direction to give space for the replacing spare robot. Just before the low energy working starts to move, it sends information like parent id, number of children, and the role it was playing within the tree to the replacing spare robot. Also, it informs children about the exchange of roles. As soon as the parent and children connect to the new replacing robot, the low energy working robot becomes a spare robot and establishes a flexible connection with any of the robots in the swarm.

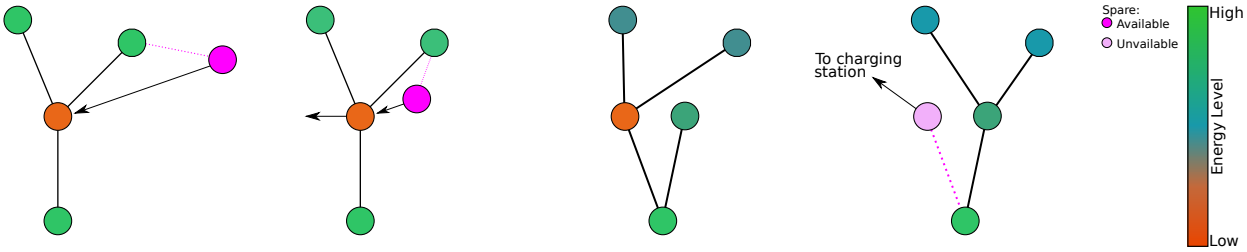

Figure 4.6: Replacing Phase

Replacing Spare Robot
When the spare robots receive the SOS message from the low energy working robot, they respond if they are available for substitution. A spare robot is considered unavailable if it currently replacing another working robot or if it approaching a charging station. If a spare robot is chosen from a pool of neighboring spare robots, the selected spare robot has to acknowledge to the low energy robot. This is to ensure the availability of the chosen spare robot is not changed.

Once they are close enough to the low energy working robot, the replacing spare robot tries to establish a strict connection with the parent of the low energy working robot and becomes a node in the tree.

Children of low energy working robot
Children of the low energy working robot receive the information about the new parent from its current parent. It connects to the new parent relieving the low energy robot.

4.3.4 Spare Management

Apart from the behaviors discussed previously, spare robots also recharge periodically and replace low energy working robots from the tree. Spare robots need to recharge if any of the following conditions are satisfied:

- The robot just left the tree because of low energy
- The robot's energy level is below a certain threshold and it is currently not extending an edge.

Each Spare robots choose the closest charging station to recharge. However, it is not necessary that charging stations are always in the neighborhood of spare robots. To

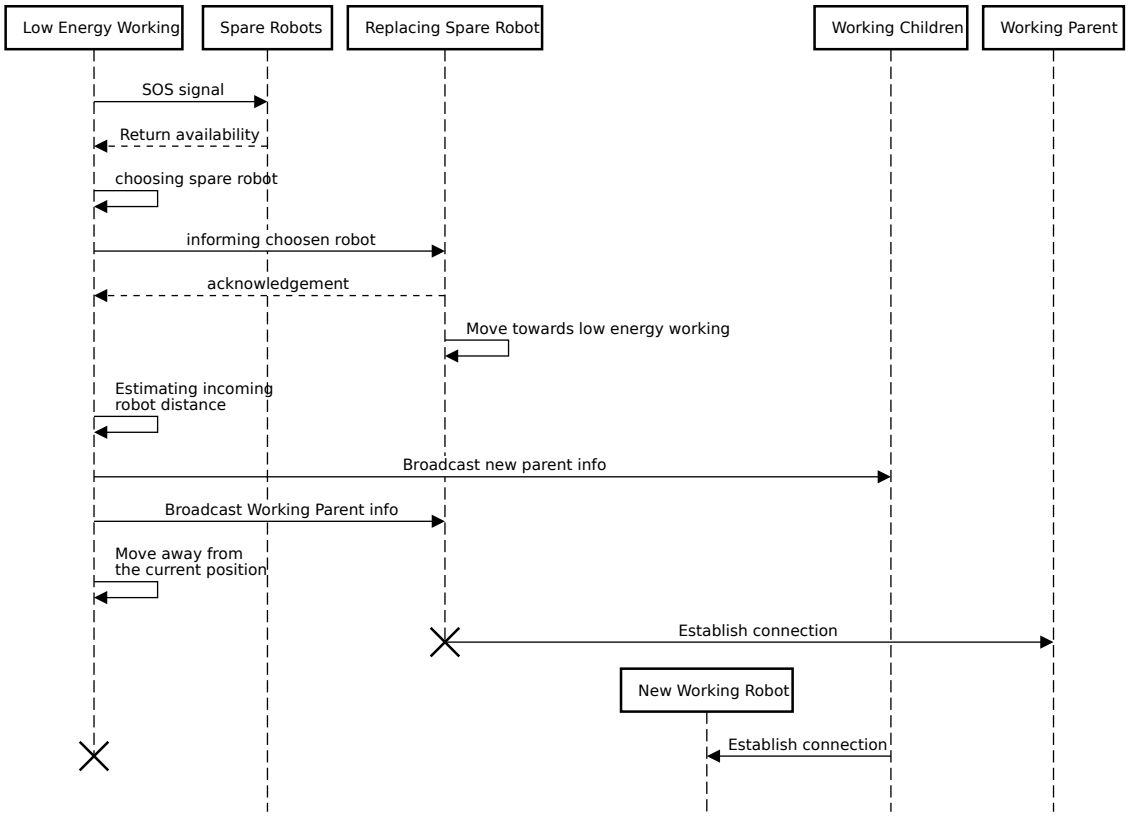

Figure 4.7: Sequence diagram

overcome this shortcoming, robots remember the charging station's location whenever they see one. Once the charging station is selected, the spare robot moves towards an approximate location and corrects them whenever a charging station is within the communication range. Spare robots stop when they are close enough to the charging station for charging. When spare robots are charged, they enter the WAIT state and continue to help to grow the tree or substitute low energy working robots. The state machine shown in Fig. 4.8 is the part of GROWTH TREE state, describing the interactions between working and spare robot. It should be noted that working states NO NEED,NEED, and AWAIT are interrupted whenever the energy level is low. This interrupt will allow the working robots to notify neighbors about the low energy and initiate the replacing phase.

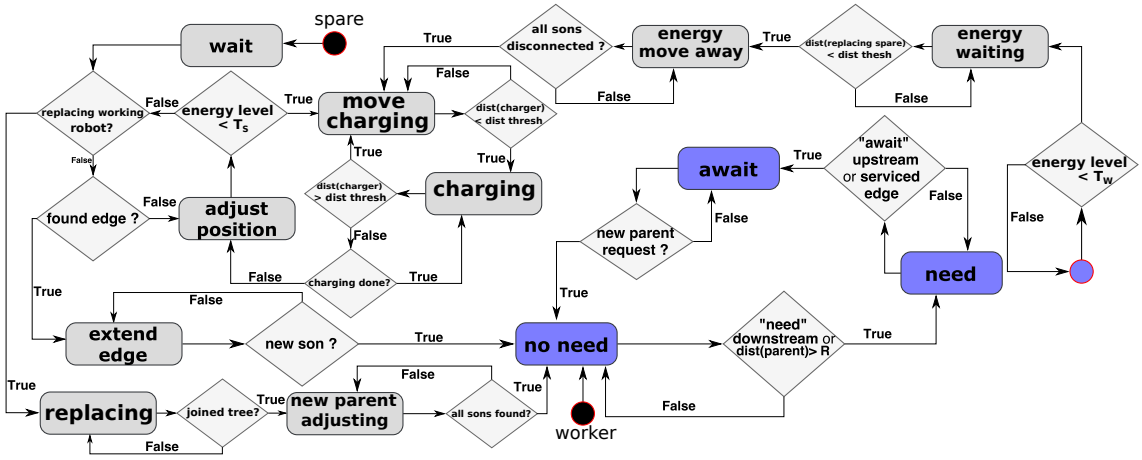

Figure 4.8: Sub State Machine

Chapter 5

Experimental Setups and Results

In this chapter, we first present our experimental setup and how we set values to the parameters that control the algorithm's performance in Sec. 5.1. Later, we define the metrics to evaluate the performance of the swarm and present the results in Sec. **??**.

5.1 Experimental Setup and Parameter Setting

We tested our approach with Khepera IV[1] ground robots using the ARGoS multi-robot simulator [20] as shown in Fig. 5.1. These robots have communication range limited to 2 meters mimicking the actual hardware configurations.

To simulate the battery model, we set the discharge parameters as shown in Table 5.1. We obtained these parameters by assuming that the fully charged robot can travel only three times along the diagonal of the $30\ m \times 30\ m$ arena at the maximum speed of $10\ cm/s$. We also considered the fact that the robots do not travel at full velocity. This is compensated by setting the time spent to 50% more than the ideal time spent (time spent if robot travel a full velocity).

To recharge the low energy robots, we deployed charging stations at the centroid of the area between two task locations as shown in Fig. 4.4. Each charging station is set with three patrolling checkpoints within the assigned area. These checkpoints are defined along the sides of the branches forcing the charging station to stay in their assigned area.

[1] `https://www.k-team.com/mobile-robotics-products/khepera-iv`

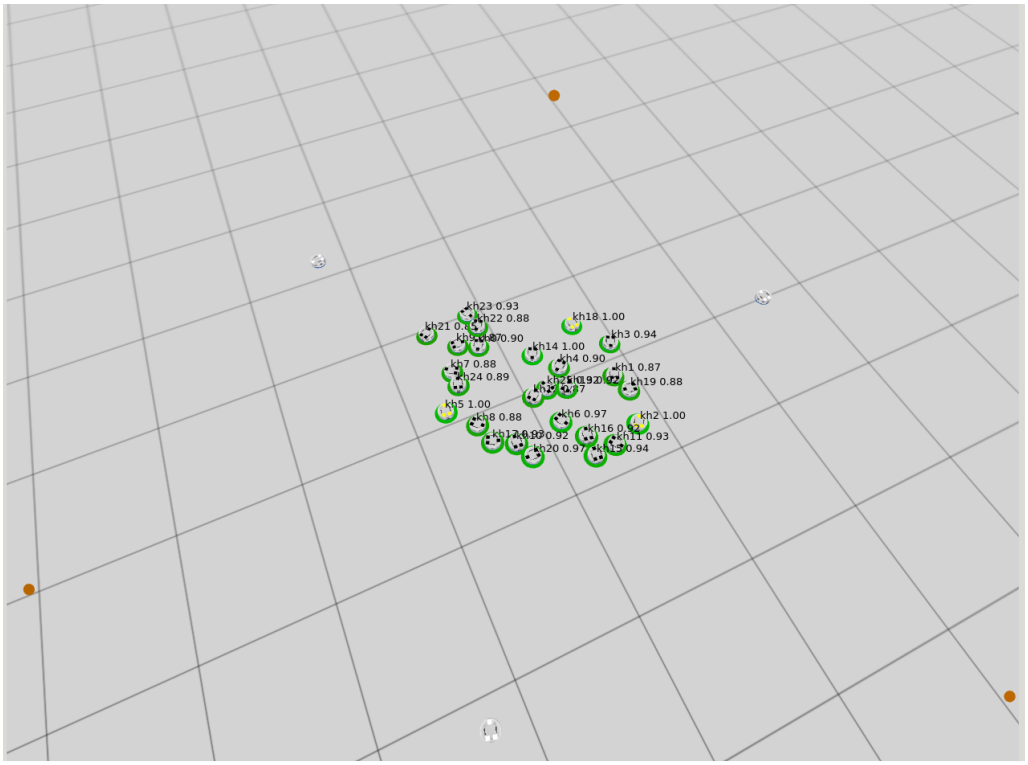

Figure 5.1: Simulation Environment in ARGoS

Parameter	Value	Units
λ_t	$5.6e-4$	per timestep
λ_{pos}	$7e-4$	per cm
λ_{orient}	$4.6e-4$	per rad

Table 5.1: Battery Parameters

We tested our approach in different experimental setups to evaluate the performance. We modified target radius, the number of targets, and redundancy factor. Various targets are placed uniformly on a circle of radius, target radius. Redundancy factor is the ratio of the total number of robots to the number of robots required to complete the task in the ideal condition. We also evaluated the performance based on line-of-sight obstruction for the communication model. Number of robots N_R required for each experiments varies based on the parameters in Table 5.2 and is calculated using Eq. 5.1. To ensure repeatability, the simulation is run with 50 different random seeds for each combination of parameters shown in Table 5.2. The experiments are continued even when the workers reach their target locations. It is considered complete whenever any robot in the swarm has no available energy or if the swarm operated for the maximum experiment time t_{max}. t_{max} is 10 times

the minimum time taken by a robot to reach any task location from the origin.

$$N_R = \alpha \frac{\rho_T}{C} N_T + N_T \tag{5.1}$$

Symbol	Meaning	Value	Unit
α	Redundancy factor	$2, 3, 4$	
ρ_T	Task Radius	$3, 6, 9$	m
N_T	Number of Tasks	$2, 3, 4$	
η	Charging rate (in %)	$0.05, 0.1, 1$	per timestep

Table 5.2: Parameters modifying the experiment

Based on parameters listed in Table 5.3, the algorithm's performance can be modified. Ideal values for those parameters are obtained using a genetic algorithm. The optimization is stopped after 100 generations because the optimizer reached a plateau region in most cases. The scoring function is designed to first optimize parameter for connectivity. Once robots are connected throughout the experiment, time to reach target location is minimized. The table shows the value of the parameter after optimization. Energy threshold parameters T_w and T_s are not optimized using an optimizer because the scoring function promotes only connectivity. Hence, it is determined by trial and error.

5.2 Results and Discussion

In this section, we provide the results we obtained for our approach in different experimental setups. The experiments are divided into three scale: small, medium and large based on the task radius ρ_t.

5.2.1 Liveliness Time

We studied the extent of the liveliness of the network considering the robots lose energy over time. We normalized the time parameter with the maximum simulation time t_{max} to compare the results across the different scenarios. The results are displayed in Fig. 5.2. When comparing across different scale, our approach works better in small scale. This is due to the fact that the charging stations need to cover only a small distance when

Table 5.3: Optimized Design Parameters

Type	Symbol	Meaning	Value	Unit
Motion	S	Safe range between parent and child	135.25581	cm
	A	Non-parent-child avoidance range	40.99	cm
	δ	Ideal distance between parent and child	154.0841	cm
	ϵ	Factor gain in parent-child interaction	10	
	τ	Magnitude of attraction to target	0.2539	
Tree Growth	R	Reconfiguration period	44.0	sec
	I	Information liveness period	0.5	sec
Uncommitted Management	E	Distance threshold for *spare* recruitment	132.1353	cm
	J	Distance threshold to switch to *connector*	6.6395	cm
Energy Parameter	T_w	Energy threshold for *working* to inform neighbors about low energy	32	%
	T_s	Energy threshold for *spare* to recharge	60	%

compared to medium and large scale. Also, this shows that one charging station per task is not sufficient as the number of robots increases based on the task radius. Within small scale, the results for keeping line-of-sight communication true is better than keeping line-of-sight communication false. This result seems contradictory because of the fact that more robots are available for replacing low energy robots when line-of-sight communication is false. However, the reason is that more spare robots are trying to perform the same action in the neighborhood leading to the crowding effect as shown in Fig. 5.3. This sometimes prevents the low energy robots to reach the charging station. In small-scale with line-of-sight communication on, results with higher redundancy factor are better because more spare robots are available for replacing low energy robots.

5.2.2 Mission Time

We studied the time performance of the worker robots to reach the target locations. We compared the performance of the swarm with and without energy constraint to understand the effect of combining two contradictory constraints. The result is shown in Fig. 5.4. In medium and large scale, the workers reached the target location occasionally, this is because robots run out of energy before it reaches the target location. This can be avoided by adding extra charging stations based the swarm size and allowing the robots to leave the swarm to recharge even when the swarm is not in the GROW TREE state. In small-scale

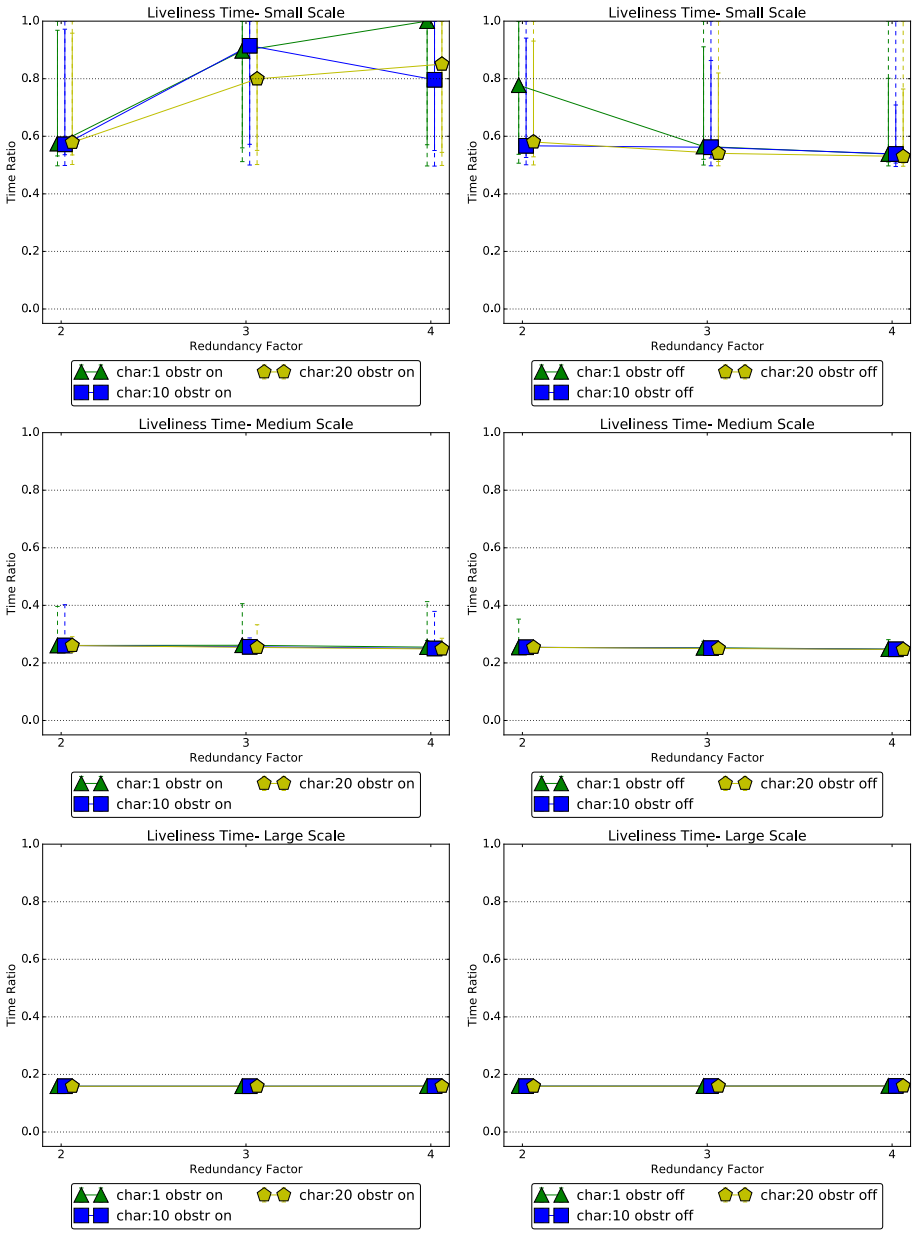

Figure 5.2: Liveliness Time Ratio

experiments, the workers were able to reach the target locations consistently and the swarm performances similar to the performance when only connectivity constraint is considered. It should be noted that the energy constraint forces the working robots to leave the tree whenever a substitution is available, this exchange of roles often makes the tree shrink for a brief period to ensure connectivity.

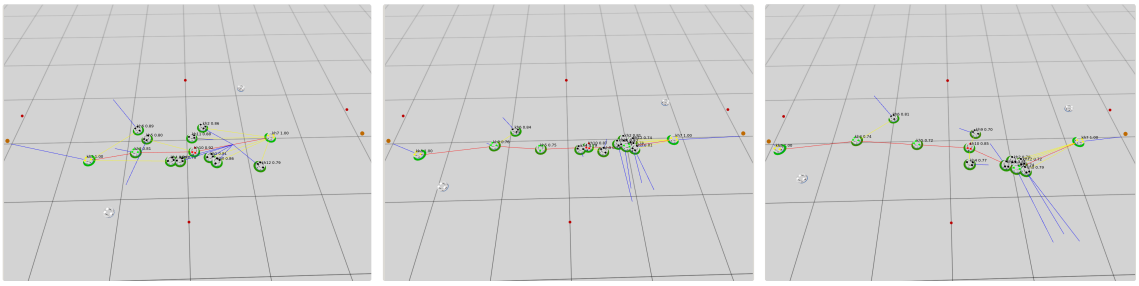

Figure 5.3: Example of crowding when line-of-sight communication is off

5.2.3 Disconnected Time

We studied the ability of our approach to maintaining connectivity. The result is shown in Fig. 5.5. We presented two sets of parameters, one considers the connectivity of the entire swarm and another is the connectivity of only the robots in the tree. The swarm-level connectivity is defined as the ratio of the number of time steps with fielder values less than 10^{-5} to the total experiment time. The tree-level connectivity is defined as the ratio of the number of time steps with at least one broken edge in the tree to the total experiment time. In small-scale, the swarm stays connected with occasional disconnection. In medium scale, with higher redundancy factor, the disconnections are noticed only at the tree level and the connectivity is usually maintained throughout the experiment. These disconnections are seen only when line-of-sight communication is true because of the possibility of the message drops. In large scale, the effect of a large number of robots is prominent and the tree edges are disconnected in most of the experiment. In this study, the trend between connectivity constraint and both connectivity and energy constraint is similar.

5.2.4 Energy Drop

From the results of liveliness and simulation time, it is evident that one charging station per task is not sufficient. In this study, we analyzed the effect adding an extra charging station. We examined only medium and large scale as it is noticed that in small-scale the swarm was able to reach the target locations consistently. In this study, the metric is the robots' energy drop when they are approaching the charging station. The results are presented in Fig. 5.6 for line-of-sight communication true and Fig. 5.7 for line-of-sight communication false. In medium-scale, when there is only one charging station per task, the energy drop is distributed between $[0, 0.2]$. The distribution is skewed towards zero when an additional charging station is introduced. In large-scale this effect is noticeable. This trend is also

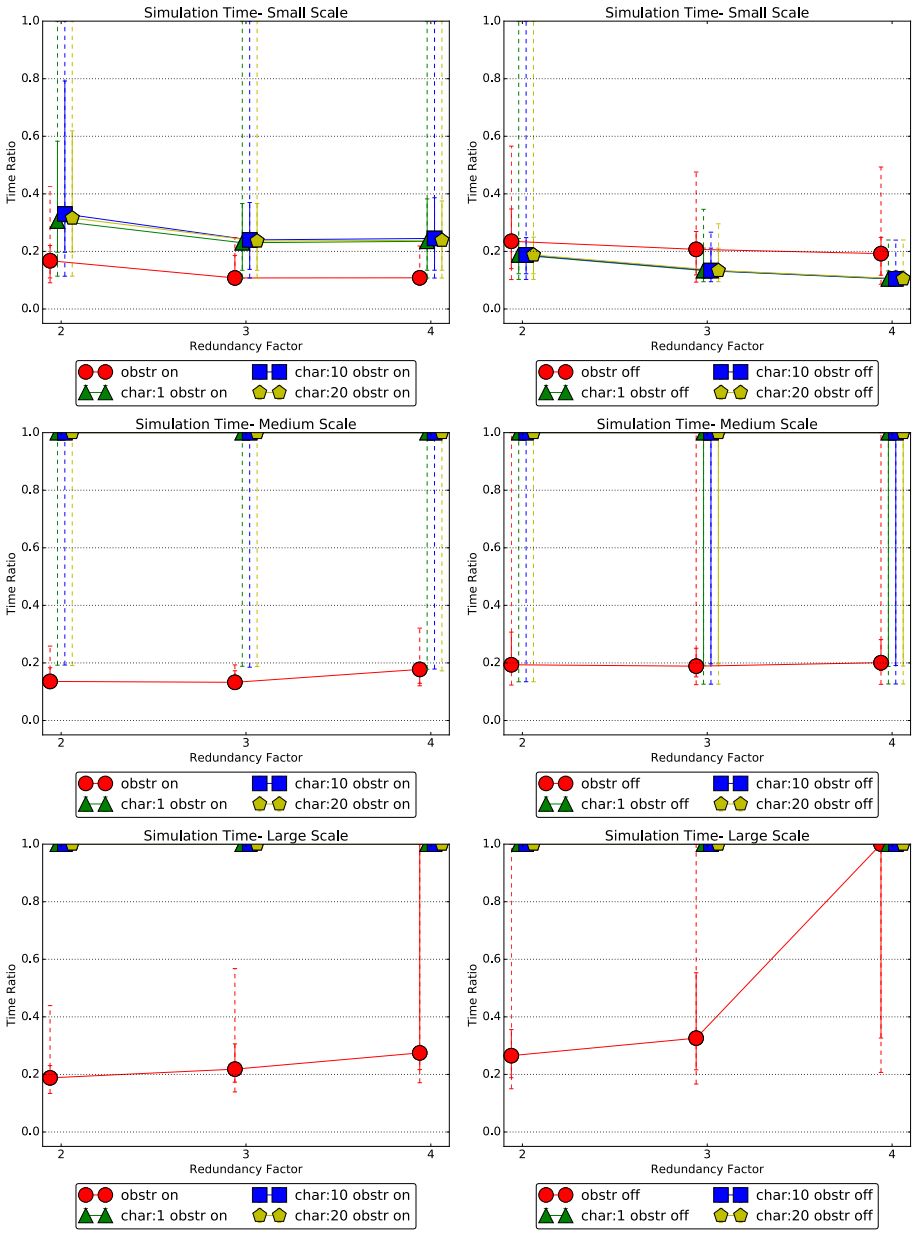

Figure 5.4: Simulation Time Ratio

followed when line-of-sight communication is false.

5.2.5 Scalability

We studied the ability of the algorithm to scale to a different number of robots. To understand scalability, we measured the time taken by the swarm to form a tree. The number of robots in the swarm is varied from 9 to 97 robots for different experimental

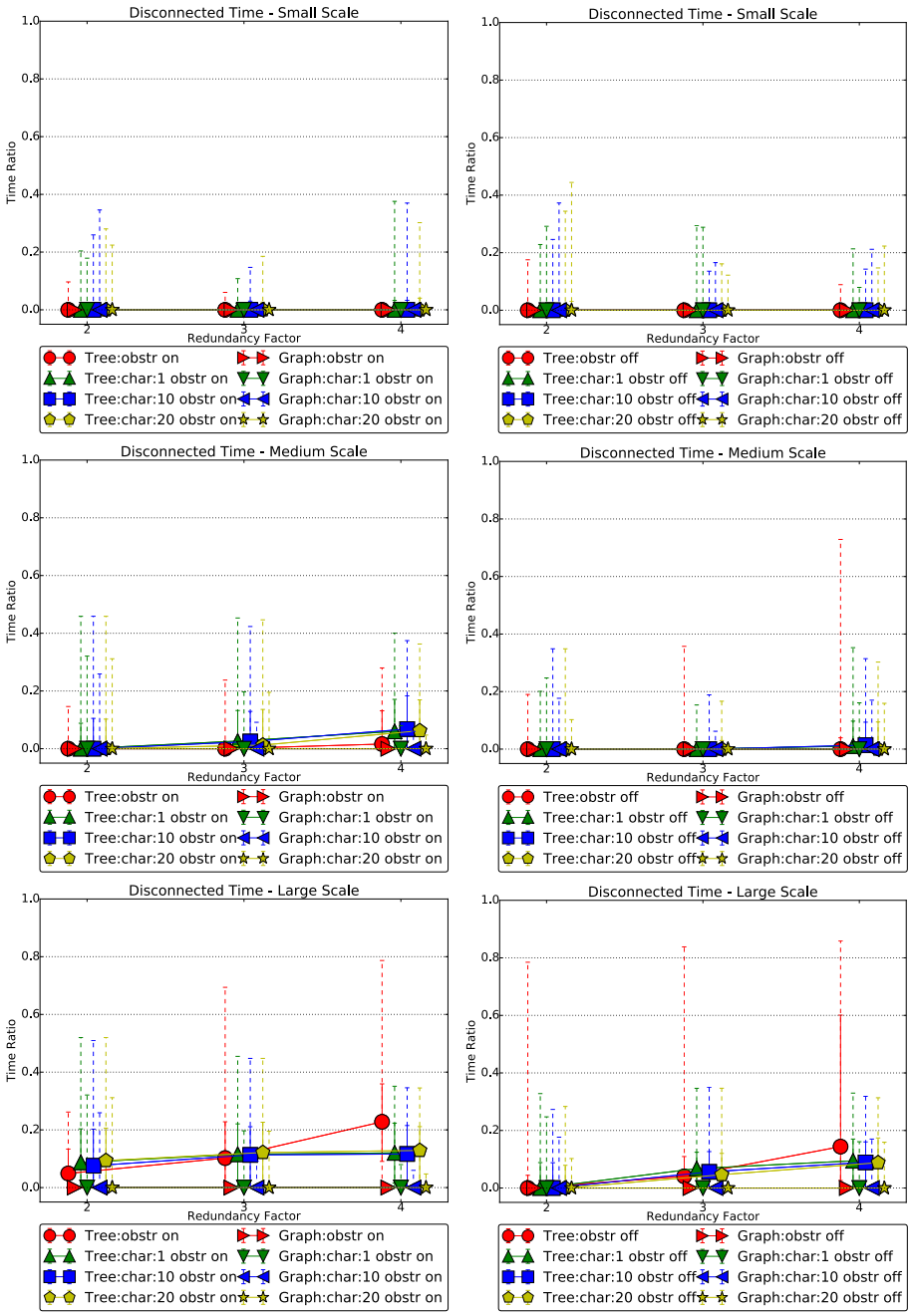

Figure 5.5: Disconnected Time Ratio

setups mentioned in Table 5.2. The obtained result is shown in Fig. 5.8. We also used a line fitting algorithm (linear regression) on the data points to calculate the slope of the line. The line equation we obtained is in Eq. 5.2. The variance score for the line fitting is 0.9, for the score in the range $[0, 1]$, 1 representing the best fit and 0 representing the worst fit.

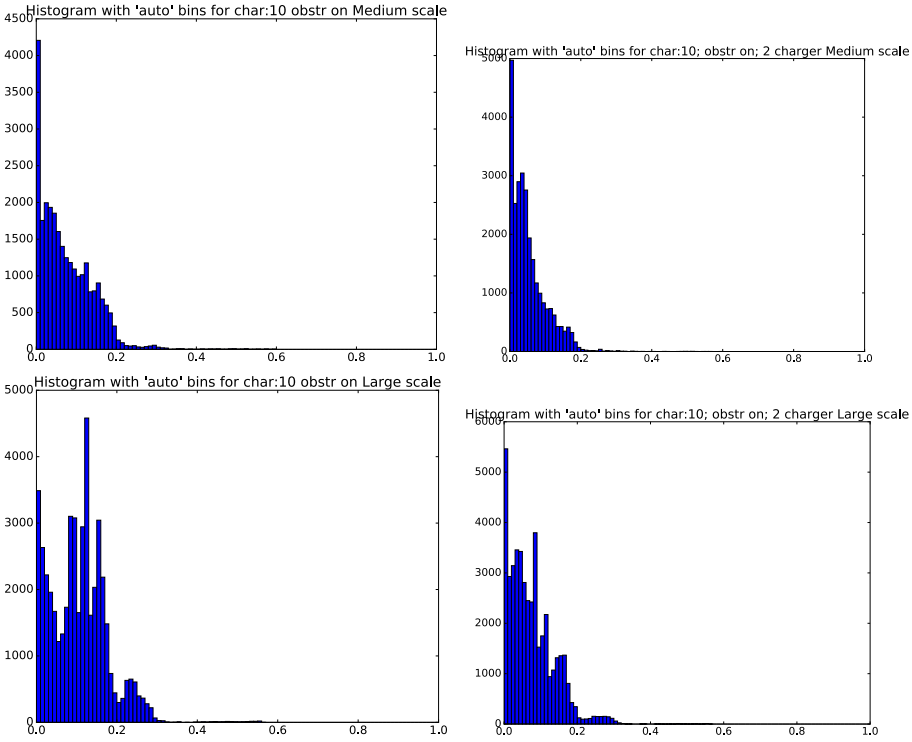

Figure 5.6: Energy Drop when approaching charging station with line-of-sight communication true

Using the line equation and the graph we can conclude that the time complexity is $\mathcal{O}(n)$.

$$y = 1.08x + 8.16 \tag{5.2}$$

where, x is the number of robots and y is the time taken for tree selection.

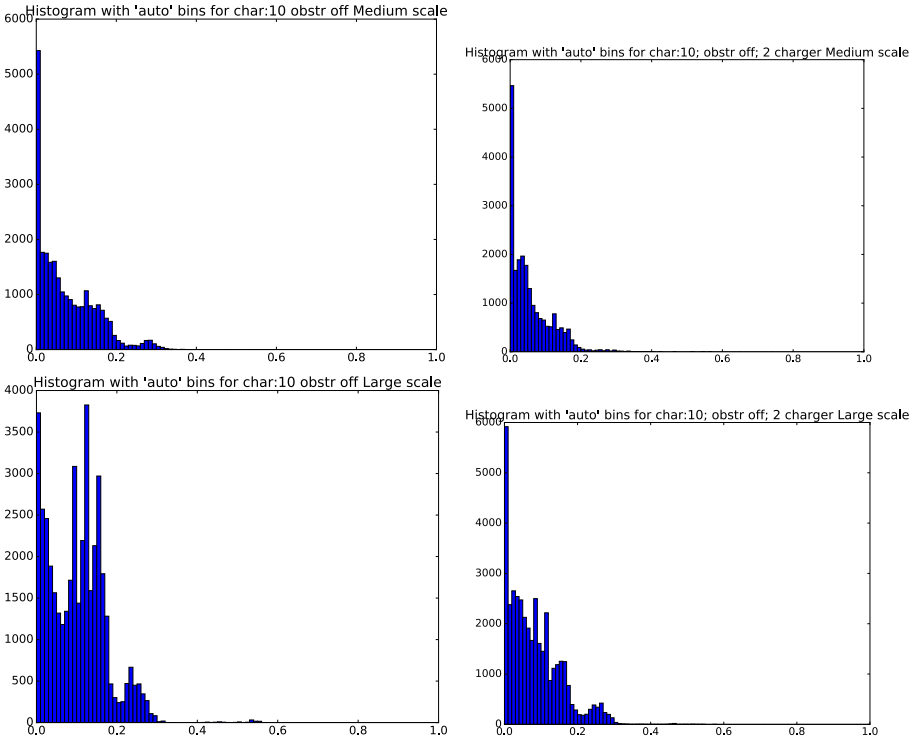

Figure 5.7: Energy Drop when approaching charging station with line-of-sight communication false

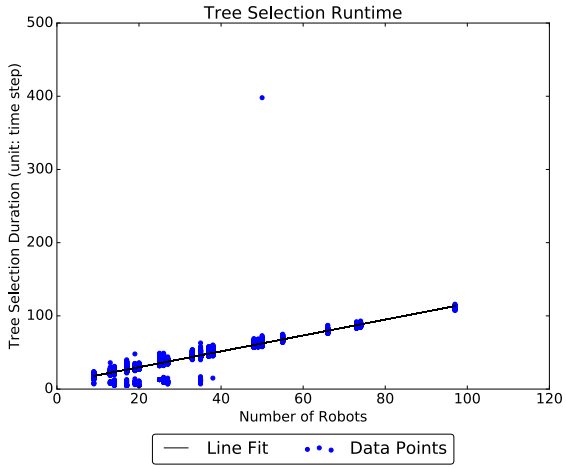

Figure 5.8: Time Complexity for Tree Selection: Data collected with different experimental setups fixing the charging factor to 0.1 and setting line-of-sight obstruction *true*. 10 timestep equals 1 second

Chapter 6

Conclusion

In this chapter, we present our conclusion in Sec. 6.1 derived from the results we presented in Chapter 5. In Sec. 6.2 we present some possible future directions of this work.

6.1 Conclusion

In this work, we present an approach to deploy a communication network that connects multiple target locations and maintain the network persistently. We considered connectivity constraint, the global property of the swarm and energy constraint, the local property of the robot. These two constraints are contradictory because connectivity requires as many robots as possible in the network while limited energy forces the robots to abandon the task for recharging. In this work, a logical tree is constructed to deploy a communication network from the worker to root. This approach allows our algorithm to scale to a large number of robots in the swarm. Considering limited energy of the robots, charging stations are placed in the area between adjacent target locations to recharge low energy robots. In this work, the communication network is deployed using only a group of robots in the swarm. Remaining robots take the role spare. These robots are later used to grow the tree and replace low energy robots from the tree.

We tested our approach in a variety of experimental setups to analyze swarm's performance in long-term connectivity maintenance, effectiveness in reaching target locations. The results suggest that our approach works in almost all the small-scale experiments. When the scale increases, our approach struggles to reach the target locations because robots' battery levels are critically low. For medium and large scale tasks, charging stations are

required to recharge robots over a large area, which is also affecting the performance of the swarm. We tested our algorithm with up to 97 robots in the swarm, indicating the scalability character of our solution.

6.2 Future Work

- **Charging station:** At times, the robots run out of energy because of the unavailability of the charging station in the neighborhood. Optimizing the charging station's motion might help the swarm to perform better.
- **Spare Robots:** The spare robots help in growing the tree and exchanging roles with low energy robots. Distribution of these robots across the environment might have the effect in the swarm's performance.
- **Environment:** Studying our approach in the presence of obstacles and moving target location would be close to realistic
- **Energy Constraint:** In this work, the energy levels are considered only in the GROWTH TREE state. Considering the energy constraint while building the logical tree would be an interesting improvement of our approach.

Bibliography

[1] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[2] Melvin Gauci, Jianing Chen, Wei Li, Tony J Dodd, and Roderich Groß. Self-organized aggregation without computation. *The International Journal of Robotics Research*, 33(8):1145–1161, 2014.

[3] Ryan K Williams and Gaurav S Sukhatme. Locally constrained connectivity control in mobile robot networks. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 901–906. IEEE, 2013.

[4] Alejandro Cornejo, Fabian Kuhn, Ruy Ley-Wild, and Nancy Lynch. Keeping mobile robot swarms connected. In *International Symposium on Distributed Computing*, pages 496–511. Springer, 2009.

[5] M Ani Hsieh, Anthony Cowley, Vijay Kumar, and Camillo J Taylor. Maintaining network connectivity and performance in robot teams. *Journal of Field Robotics*, 25(1-2):111–131, 2008.

[6] Lorenzo Sabattini, Cristian Secchi, and Nikhil Chopra. Decentralized estimation and control for preserving the strong connectivity of directed graphs. *IEEE transactions on cybernetics*, 45(10):2273–2286, 2015.

[7] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[8] Cinara Ghedini, Cristian Secchi, Carlos HC Ribeiro, and Lorenzo Sabattini. Improving robustness in multi-robot networks**. *IFAC-PapersOnLine*, 48(19):63–68, 2015.

[9] Cinara Ghedini, Carlos HC Ribeiro, and Lorenzo Sabattini. Improving the fault tolerance of multi-robot networks through a combined control law strategy. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 209–215. IEEE, 2016.

[10] Vinicius A Battagello and Carlos HC Ribeiro. Analysis of the effects of failure and noise in the distributed connectivity maintenance of a multi-robot system. In *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on*, pages 427–432. IEEE, 2014.

[11] Vinícius A Battagello and Carlos HC Ribeiro. Analysis of the effects of communication delay in the distributed global connectivity maintenance of a multi-robot system. In *CompleNet*, pages 149–157, 2015.

[12] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. On decentralized connectivity maintenance for mobile robotic systems. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 988–993. IEEE, 2011.

[13] Tina Setter and Magnus Egerstedt. Minimum time power-aware rendezvous for multi-agent networks. In *Control Applications (CCA), 2014 IEEE Conference on*, pages 2159–2164. IEEE, 2014.

[14] Tina Setter and Magnus Egerstedt. Energy-constrained coordination of multi-robot teams. *IEEE Transactions on Control Systems Technology*, 25(4):1257–1263, 2017.

[15] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior*, 15(3):289–305, 2007.

[16] Kevin Leahy, Dingjiang Zhou, Cristian Ioan Vasile, Konstantinos Oikonomopoulos, Mac Schwager, and Calin Belta. Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints. *Autonomous Robots*, 40(8):1363–1378, 2016.

[17] Guannan Li, Ivan Švogor, and Giovanni Beltrame. Self-Adaptive pattern formation with battery-powered robot swarms. *2017 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2017*, pages 253–260, 2017.

[18] Jan Bjerknes and Alan Winfield. On fault tolerance and scalability of swarm robotic systems. *Distributed autonomous robotic systems*, pages 431–444, 2013.

[19] Nathalie Majcherczyk, Adhavan Jayablan, Giovanni Beltrame, and Carlo Pinciroli. Decentralized connectivity-preserving deployment of large-scale robot swarms. In *Intelligent Robots and Systems, 2018 IEEE/RSJ International Conference on*. IEEE, 2018. Submitted for review.

[20] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

Chapter 7

Appendices

Appendix A

This section includes the paper we submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and System.

Decentralized Connectivity-Preserving Deployment of Large-Scale Robot Swarms

Nathalie Majcherczyk[1], Adhavan Jayabalan[1], Giovanni Beltrame[2] and Carlo Pinciroli[1]

Abstract— **We present a decentralized and scalable approach for deployment of a robot swarm. Our approach tackles scenarios in which the swarm must reach multiple spatially distributed targets, and enforce the constraint that the robot network cannot be split. The basic idea behind our work is to construct a logical tree topology over the physical network formed by the robots. The logical tree acts as a *backbone* used by robots to enforce connectivity constraints. We study and compare two algorithms to form the logical tree: *outwards* and *inwards*. These algorithms differ in the order in which the robots join the tree: the outwards algorithm starts at the tree root and grows towards the targets, while the inwards algorithm proceeds in the opposite manner. Both algorithms perform periodic reconfiguration, to prevent suboptimal topologies from halting the growth of the tree. Our contributions are (i) The formulation of the two algorithms; (ii) A comparison of the algorithms in extensive physics-based simulations; (iii) A validation of our findings through real-robot experiments.**

I. Introduction

Swarm robotics [1] is a branch of collective robotics that studies decentralized solutions for the problem of coordinating large teams of robots. Robot swarms are a promising technology for large-scale scenarios, in which performing spatially distributed tasks would entail prohibitive costs for single-robot solutions [1]. Typical examples include planetary exploration [2], deep underground mining [3], ocean restoration, and agriculture.

A common aspect in these scenarios is the necessity to maintain a coherent state across the swarm. Many basic coordination problems can be solved assuming low-bandwidth, occasional communication or even no communication. However, global connectivity is an asset when information must be exchanged in a timely manner, either to optimize a global performance function, or to aggregate data in a sink. Task allocation scenarios with stringent space and time constraints, such as warehouse organization and search-and-rescue operations [4] are prime examples of this category of problems. In these scenarios, it is desirable for the robot network to allow both short-range and long-range information exchange.

In this paper, we tackle the problem of deploying a robot network in a decentralized fashion, under the constraint that long-range information exchange must be possible at any time during a mission. We assume that the robots must reach a number of distant locations. While navigating to these locations, the robots must spread without splitting the network topology in disconnected components. The robots must achieve a final configuration in which data can flow between any two target locations, using the robots as relays.

It is important to notice that it is not required for all of the robots to take part in the final topology. Rather, it is desirable that as few robots as possible are engaged in connectivity maintenance, as this would free any extra robot for others tasks or to act as occasional replacement for damaged robot in the topology. In constrast, the robots that are part of the final topology must form a persistent communication backbone that can be used by any robot when necessary.

This aspect sets apart our work from existing research on connectivity maintenance, which generally requires *all* robots to be part of the connected topology. The literature on this topic can be broadly divided in two classes: algorithms in which the robots must attain a final, static structure to maximize coverage [5], and algorithms in which global connectivity is enforced while navigating to a specific location as a single unit (flocking) [6]. Our work, in contrast, aims to create a dynamic, decentralized communication infrastructure that connects specific locations and uses as few robots as possible.

Our approach assumes that the robots are initially deployed in a compact, connected cluster. The robots then form a logical *tree* over the physical network topology. By growing the tree over time, the distribution of the robots progressively and dynamically extends to reach the target locations. The final configuration is a star-like topology, in which data can flow between any two target locations.

The main contributions of this work are:

1) The formalization of two algorithms to form and grow logical tree topologies that connect multiple target locations;
2) A comparative study of the algorithms, based on extensive physics-based simulations;
3) The validation of our findings through a large set of real-robot experiments.

The rest of this paper is organized as follows. In Sec. II we formalize the problem statement. In Sec. III we present our methodology. In Sec. IV we report an evaluation of the algorithms. In Sec. V we discuss related work. The paper is concluded in Sec. VI.

[1] N. Majcherczyk, A. Jayabalan, and C. Pinciroli are with Robotics Engineering, Worcester Polytechnic Institute, 85 Prescott St., Worcester MA 10609, USA. E-mail: {nmajcherczyk, ajayabalan,cpinciroli}@wpi.edu

[2] G. Beltrame is with Department of Computer and Software Engineering, École Polytechnique Montréal, 2900 Edouard Montpetit Blvd, Montréal, QC H3T 1J4, Canada. E-mail: giovanni.beltrame@polymtl.ca

II. Problem Statement

A. Robot Dynamics

We consider N robots with linear discrete dynamics

$$x_i(t+1) = Ax_i(t) + Bu_i(t)$$

where $x_i(t) \in \mathbb{R}^{2M}$ is the state of robot i at time t, $u_i(t) \in \mathbb{R}^{2M}$ is the control signal, and $A, B \in \mathbb{R}^{2M \times 2M}$. The state $x_i(t)$ is defined as $[p_i(t), v_i(t)]$, where $p_i(t) \in \mathbb{R}^M$ designates the position of robot i and $v_i(t) \in \mathbb{R}^M$ its velocity. State and controls are subject to the convex constraints

$$\forall t \geq 0 \quad x_i(t) \in \mathcal{X}_i \quad u_i(t) \in \mathcal{U}_i.$$

In this work we focus on 2-dimensional navigation ($M = 2$).

B. Robot Communication

We assume that the robots are capable of *situated communication*. This is a communication modality in which robots broadcast data within a limited range C, and upon receiving data, a robot is able to estimate the relative position of the data sender with respect to its own local reference frame.

We define the *communication graph* $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$, where $\mathcal{V}$ is the set of robots $\{1, \ldots, N\}$, and $\mathcal{E}_C \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting the robots. An edge (i, j) between two robots exists at time t if their distance is within their communication range C, i.e., $\| p_i(t) - p_j(t) \| \leq C$.

Definition 1 (Graph connectivity): A graph is *connected* is there exists a path between any two nodes.

Graph connectivity can be verified through well-known concepts in spectral graph theory. From the definition of the graph adjacency matrix

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E}_C \\ 0 & \text{otherwise} \end{cases}$$

and of the graph degree matrix

$$D_{ij} = \begin{cases} \sum_k A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

we can derive the Laplacian matrix $L = D - A$. The graph is connected if and only if the second smallest eigenvalue of L is greater than 0. For this reason, this eigenvalue is called *algebraic connectivity* or *Fiedler value* [7]. We will employ algebraic connectivity as a performance measure in the experiments of Sec. IV.

C. Objectives

The objective of this work can be stated as follows: we aim to create a progressive deployment strategy that can reach an arbitrary number of geographically distant tasks while satisfying connectivity constraints. In particular, the final configuration of the network topology must allow communication between any two target locations.

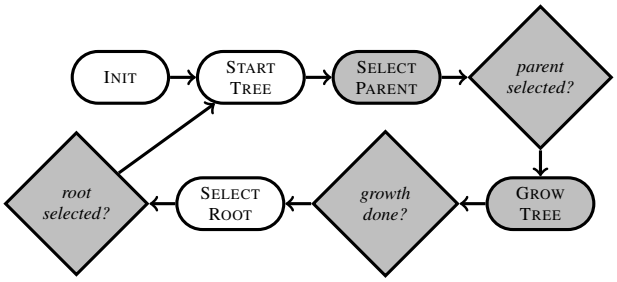

Fig. 1: The high-level Finite State Machine that formalizes the individual robot behaviors in the two tree-formation algorithms. Rounded rectangles denote states, and diamonds denote *barriers*, i.e., conditions that all robots must meet before proceeding to the next state. States filled in white are common among both algorithms; states and barriers filled in light gray differ across algorithms.

III. Approach

A. Roles

In both algorithms, we assume that the robots are initially deployed in a fully connected cluster. Subsequently, the robots must form a tree by dynamically assuming a specific role in the process.

In both tree-forming algorithms, the robots can have four possible roles: *root*, *worker*, *connector*, or *spare*. The *root* robot corresponds to the tree root, and at any time during the execution only one robot can assume this role. The *worker* robots are the tree leaves, and they correspond to robots that must reach the target locations, forcing the tree to grow progressively. The *connector* robots dynamically join the tree to support its growth, leaving the pool of available *spare* robots.

B. High-Level Behavior Specification

The algorithms can be formalized through a high-level state machine that encodes the behavior of every robot, as depicted in Fig. 1.

Every robot starts in state INIT. We assume that a process that assigns the role of *worker* to the robots closest to the targets has been already executed, through, e.g., a task allocation algorithm or a gradient-based algorithm. In addition, a random robot is assumed assigned the role of *root*. The other robots are initially *spare*.

The START TREE state is triggered by the root, which propagates a signal throughout the robot network. This state signifies that a new tree must be created. As the message propagates throughout the network, the robots estimate their distance from the root. This is possible because of situated communication—every robot can estimate a relative vector to each of its immediate neighbors.

Robots receiving a "start tree" signal switch to SELECT PARENT. In this state, each robot must identify a new parent to attach to. The selection of a new parent aims to create the shortest possible paths between the root robot and the *worker* robots, i.e., the leaf nodes in the tree. The specifics of this

state are different in the *outwards* and *inwards* algorithms, and are explained in Sec. III-D and Sec. III-E. At the end of this state, a robot is part of two trees—the one from the previous iteration of the algorithm (excluding the very first iteration), and a new one that reflects the new parent.

Once every robot has selected a new parent, the robots switch to the GROW TREE state, in which the robots forget the tree from the previous iteration and *spare* robots are accepted to join an edge. The algorithms differ in the implementation of this state, and details are reported in Sec. III-D and Sec. III-E.

Once the growth state is complete, the robots switch to the SELECT ROOT state. As the tree grows, the initial choice of the root robot (which is random) or an uneven distribution of target locations might render the tree topology unoptimal. By selecting a new root, the swarm can balance the tree branches, thus fostering even growth over time. The design of this state is illustrated in Sec. III-C.

Finally, the new assigned root switches to state START TREE and broadcasts a new "start tree" signal.

In Fig. 1, certain state transitions are marked with diamonds. These transitions, which we call *barriers*, are special in that they correspond to "wait states" in which the robots must stay until a certain condition is verified for every robot. The specific implementation of these conditions depends on the algorithms. However, the general principle is that the root aggregates the information necessary to evaluate a certain condition, and then broadcasts a "go" signal throughout the tree. The "go" signal triggers a state transition in the robots that receive it.

C. Selection of a New Root

The purpose selecting a new root is to balance the tree, which fosters better growth and compensates for an uneven distribution of target locations. In addition, balancing the tree has positive effects on the scalability of our algorithms. Every state in our algorithms involves some form of diffusion/aggregation process across the tree, with a time complexity that is linear with the depth of the tree. By balancing the tree, we also shorten its depth, thus lowering the time for diffusion/aggregation processes to complete.

These considerations suggest that the best location for the root is as close as possible to the centroid of the distribution of robots. The selection of a new root occurs at the end of a tree configuration loop, but the data upon which the process depends is collected in state SELECT PARENT, when the robots select a new parent.

The algorithm provides an estimate of the centroid in the root reference frame by adding up each robot contribution from the leaves to the root. The algorithm is formalized in Alg. 1. An intuitive explanation of this algorithm proceeds as follows. Since each robot only knows its relative position to other robots, it must send to its parent an accumulation vector $\mathbf{q}_i$ which aggregates its contributions and that of all its descendants in the tree, according to its own reference frame. Fig. 2 reports an example with three robots, where robot 0 is the root, robot 2 is a *worker*, and robot 1 is a *connector*.

Algorithm 1 Distributed centroid estimation algorithm executed by robot i: $\mathbf{a}_i$ denotes an accumulator value; $\mathbf{q}_i$ denotes the contribution of robot i to the estimation algorithm; c_i and d_i denote the number of robots in the swarm estimated by robot i and the tree depth of robot i, respectively; and $\mathbf{p}_i^{\text{parent}}$ is the vector from robot i to its parent.

$\mathbf{a}_i = 0$
for all child j **do**
  $\mathbf{q}_j^i$ = express $\mathbf{q}_j$ in i's reference frame
  $\mathbf{a}_i = \mathbf{a}_i + \mathbf{q}_j^i$
end for
if robot i has a parent **then**
  $\mathbf{q}_i = \mathbf{a}_i - (\underbrace{c_i - d_i}_{\text{nb descendants}} + 1) \cdot \mathbf{p}_i^{\text{parent}}$
end if
if robot i is the root **then**
  $\mathbf{q}_i = \mathbf{a}_i / \underbrace{c_i}_{\text{robot count}}$
end if

Algorithm 2 Tree-based count algorithm for robot i. The depth of robot i in the tree is denoted as d_i. The depth of the tree root is set to 1. The count calculated by robot j is denoted as c_j.

switch number of children **do**
  case 0
    return d_i
  case 1
    return c_{child}
  default
    return $\sum_{\text{neighbors } j}(c_j - d_i) + d_i$
end switch

To perform the final calculation of the centroid, Alg. 1 needs the number of robots in the swarm. A tree-based distributed algorithm to count the number of robots currently committed in the tree is reported in Alg. 2. This algorithm requires the robots to aggregate a partial count, denoted with c_i, from the tree leaves to the root.

In our implementation, both Alg. 1 and Alg. 2 are executed in parallel in state SELECT PARENT. In SELECT ROOT, the current root compares its position and the position of its neighbors to the centroid estimate (all are expressed in its reference frame). If the current root is the closest to the centroid, it remains the root and restarts a new tree loop. Otherwise, it designates a new root and sends the centroid vector and the angle to the new root. When the new root receives this message, it sends an acknowledgement message to the old root, and then it expresses the centroid in its own reference frame. The process is repeated until the root is the closest robot to the centroid estimate.

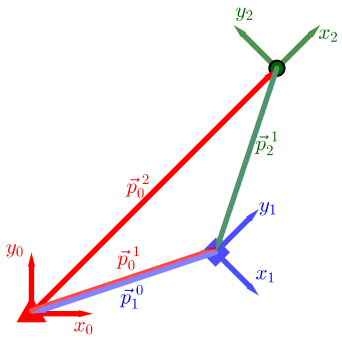

Fig. 2: The red triangle represents robot 0 with the root reference frame. The blue square represents robot 1, which is a child of robot 0 and a parent of robot 2, in turn represented by the green circle.

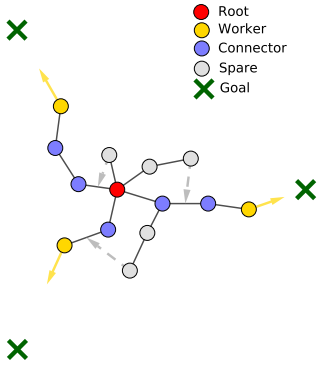

Fig. 3: Spare management in the *outwards* algorithm. The useful tree edges (blue nodes) are extended by pruning useless tree branches (grey nodes).

D. The Outwards Algorithm

The intuition behind the outwards algorithm is to build a logical spanning tree over the entire robot network. The process starts at the root, and robots join the tree progressively.

In state SELECT PARENT, robot i considers its neighbors as potential candidates. Viable candidates are non-*workers* already in the tree and at a distance smaller than the communication range. Among these, the robot selects the closest robot. The robot commits to the tree and starts broadcasting its parent id, which indicates to the parent robot that robot i is a child and that i is a *connector*. Each *connector* maintains its list of children and checks for obstructions of line-of-sight with respect to its parent. If a robot can not receive data from its selected parent, it selects another parent and updates its data.

In state GROW TREE, the robots undergo two main phases: first, they discard the information about the old tree; second, they prune tree branches that contain no *workers*. To establish whether a branch contains a *worker*, when a *worker* selects a parent (state SELECT PARENT), the latter propagates this information upstream towards the root.

The branches not containing a *worker* are considered

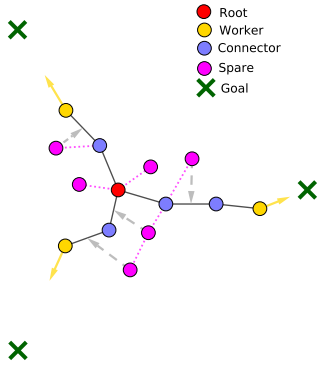

Fig. 4: Spare management in the *inwards* algorithm. Useful tree edges (blue nodes) are extended by adding *spare* robots (purple nodes)

"useless" and the robots that are part of them take the *spare* role. To disband a useless branch, *spare* robots leave it starting from the leaves. The leaves curl the branch back towards the root, and upon entering in contact with another branch might decide to join it. The logic for spares to join a branch is explained in Sec. III-F

E. The Inwards Algorithm

The intuition behind the *inwards* algorithm is that the robots join the tree starting from the workers towards the root. Growth is therefore directed, and the final topology is a *sparse* tree, in that only a subset of the robots takes part in it. The *spare* robots, in contrast to the *outwards* algorithm, do not form branches; rather, they disperse along the tree and select a robot to use as reference.

In state SELECT PARENT, viable candidates for parent selection are non-workers in the tree or robots not in the tree which are at a distance smaller than the communication range C. Among these, a robot selects a neighbor with the smallest distance to the root. When the robot i commits to the tree, it broadcasts its parent id, which indicates to the parent robot that robot i is a child and that i is a *connector*. In the *inwards* algorithm, by definition, all branches are useful because they all terminate with a *worker* as leaf node.

In state GROW TREE, *spare* robots attempt to join a branch. The logic for branch joining is the same as in the *outwards* algorithm, and it is explained in Sec. III-F.

F. Spare Management

The state machine diagram in Fig. 5 describes the part of the GROW TREE state that concerns the interaction between *spare* robots and non-*spare* robots (i.e., *connectors*, *workers*, and *root*).

Non-spare robots enter the NO NEED state when they have no need for a spare robot. They exit this state either if their distance to their parent becomes smaller than the *safe communication range* S, or if at least one of their children's state is the NEED state. In the NEED state, each robot continuously checks if it is in an edge selected by a *spare* robot, or if their parent is in the AWAIT state. If

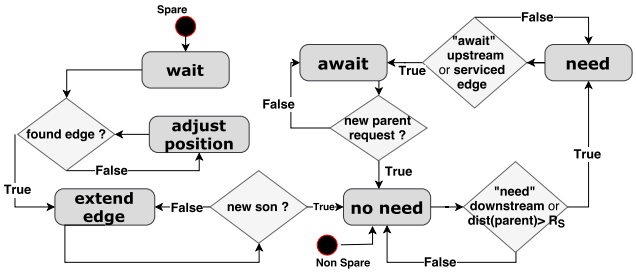

Fig. 5: Interaction between *spare* and *non-spare* robots.

one of these conditions is fulfilled, the robot transitions to the AWAIT state. In the AWAIT state, the robot is waiting the insertion of a spare robot either in one of its edges or upstream in the tree.

Spare robots enter the WAIT state and look for an edge to extend. They transition to the EXTEND EDGE state or the ADJUST POSITION state after performing a search for edges in need among their neighbors. In the ADJUST POSITION state, spare robots rotate around their parent if they are within the safe radius or move towards their parent in a straight line otherwise. In the EXTEND EDGE state, spare robots head for the middle of the edge to be extended.

G. Robot Motion

The integrity of the tree over time is ensured by constraining the robots' motion. We enforce the constraints by expressing the robot motion as a sum of virtual potential forces (we omit time dependency for brevity of notation):

$$u_i = \begin{cases} u_i^{\text{tree,old}} + u_i^{\text{tree,new}} + f_i(d_i^{\text{parent}})(u_i^{\text{target}} + u_i^{\text{avoid}}) & \text{if } d_{i,j} \leq E \\ \mathbf{p}_i^{\text{parent}} & \text{otherwise} \end{cases}$$

where $d_{i,j} = \| p_i - p_j \|$, $E < C$ is the *emergency* range beyond which a robot is dangerously distant from its parent, and

- $u_i^{\text{tree,old}}$ and $u_i^{\text{tree,new}}$ indicate the interaction law between robots (i, j) in a parent-child relationship, in either the old or the new tree. We use the control law

$$u_i^{\text{tree}} = \frac{\epsilon}{d_{i,j}} \left(\left(\frac{\delta}{d_{i,j}} \right)^2 - \left(\frac{\delta}{d_{i,j}} \right)^4 \right)$$

  where $\delta = E$ and ϵ are parameters to set at design time.
- u_i^{target} is a control law that attracts a robot to a target, promoting tree growth. For workers, this is a force that points the assigned target location l_i and calculated with

$$u_i^{\text{target}} = \tau \frac{l_i - p_i}{\| l_i - p_i \|}$$

  where τ is a design parameter. Workers propagate to their parents the calculated u_i^{target}, and connectors apply it in turn.
- u_i^{avoid} is a repulsive force for obstacle avoidance between neighbors not in a parent-child relationship.
- $f_i(d_i^{\text{parent}})$ is a function defined as follows:

$$f_i(d_i^{\text{parent}}) = \begin{cases} 1 & \text{if } d_i^{\text{parent}} \leq S \\ 0 & \text{otherwise} \end{cases}$$

  where d_i^{parent} is the distance between a robot and its parent and $S < E$ is the *safe communication range*. Through this function, a robot can ignore navigation to target and obstacle avoidance to perform emergency maneuvers when the distance to its parent becomes unsafe.

IV. Evaluation

A. Parameter Setting

The dynamics and the performance of our algorithms depends on the design parameters reported in Table I. To set their value, we used a genetic algorithm. We ran multiple instances of the optimization process for both *inwards* and *outwards*, and Table I reports the best values we found.

Every instance of the optimization was executed for 100 generations. We set this number as a reasonable margin after observing that, across instances, after about 50 generations the optimization process would find a plateau beyond which no improvement was found.

Every generation consisted of trials in which 9 Khepera IV robots[1] were placed in the arena in a tight cluster. We configured two types of trials:

- 2 target locations on a circle with a radius of 2.3 m at 180 ° from each other;
- 3 targets on a circle with a radius of 1.6 m at 120 ° from each other.

We ran the trials in the ARGoS multi-robot simulator [8], and maximized a two-step performance function. The first step (performance 0 to 1) promoted connectivity maintenance by penalizing the time spent with disconnected robots; the second step (performance 1 to 2) was activated when no disconnections occurred, and higher values corresponded to lower times to reach the targets.

B. Simulated Experiments

We tested the performance of the algorithms by varying three parameters: the target radius, the redundancy factor, and number of targets. We placed multiple targets on a circle with equal angles between each other. The *target radius* is the radius of the circle. We chose radii of 3, 6 and 9 meters corresponding to small, medium and large scales. The *redundancy factor* is the factor by which we multiply the minimum required number of robots needed to reach all the targets given our communication range. We tested the values of 2, 3 and 4 for this parameter. The *number of targets* was 2, 3, and 4. The largest configuration we considered involved 94 robots. Each scenario was executed with 50 different random seeds. We ran all the experiments for both algorithms with and without activating line-of-sight obstructions in the communication models of ARGoS, to test the effect of this aspect.

[1] https://www.k-team.com/mobile-robotics-products/khepera-iv

TABLE I: Optimized Design Parameters

Type	Symbol	Meaning	Outwards	Inwards	Unit
Motion	S	Safe range between parent and child	138.93	135.25581	cm
	A	Non-parent-child avoidance range	43.16	40.99	cm
	δ	Ideal distance between parent and child	190	154.0841	cm
	ϵ	Factor gain in parent-child interaction	10	10	
	τ	Magnitude of attraction to target	0.49	0.2539	
Tree Growth	R	Reconfiguration period	38.8	44.0	sec
	I	Information liveness period	1.2	0.5	sec
Uncommitted Management	E	Distance threshold for *spare* recruitment	132.09	132.1353	cm
	J	Distance threshold to switch to *connector*	9.79	6.6395	cm

1) Simulation Time: We studied the time performance of both algorithms, and declared an experiment finished when all workers reach their targets. To compare results across different scales, we normalized the mission duration by the maximum allowed time. The maximum allowed time was computed by considering the time for a robot to reach a target from the center of the arena; this time was then multiplied by 10. The results are reported in Fig. 6. For small scales, the *outwards* algorithm outperforms the *inwards* algorithm. However, as the scale of the experiment is increased, the directed growth of the *inwards* algorithm is increasingly advantageous. In addition, with the *outwards* algorithm, some missions do not reach their targets in the allotted time limits when higher redundancy factor is employed. This is due to the increased interference that too many useless branches create in robot navigation. This effect is not prominent in the *inwards* algorithm because the robots are added to the tree only when it is necessary.

2) Disconnected Time: We studied the ability to maintain connectivity by considering the following metrics: (i) The *disconnected time ratio*, defined as the number of time steps (over the total experiment time) with at least a broken edge in the tree; (ii) The *Fiedler value time ratio*, defined as the number of time steps (over the total experiment time) with swarm-wide Fiedler value lower than 10^{-3}. The results are reported in Fig. 7. In small-scale scenarios, in only two experiments out of 50 have positive disconnected time, and the global communication graph always stays connected. In medium-scale scenarios, larger numbers of redundant robots cause occasional line-of-sight obstructions that delay messages exchanges, but connectivity is generally maintained throughout the duration of the experiment. In large-scale scenarios, the disruptive effect of a large number of redundant robots is prominent for both algorithms. With fewer robots, the inwards algorithm is capable of maintaining global connectivity in all of the experiments, despite occasional breaking of tree edges (in less than 5% of the experiments).

C. Real-Robot Validation

To validate the simulated results simulations, we tested our algorithms with 9 Khepera IV robots. A Vicon motion capture system was used to track the position and orientation of the robots throughout the duration of the experiments, and to simulate situated communication. We employed 2 experimental scenarios: (i) 2 targets on a circle with a radius of 2.3 meters at approximately 180 degrees from each other; (ii) 3 targets on a circle with a radius of 1.6 meters at approximately 120 degrees from each other. We rescaled the distance-related paremeters in Table I to fit the arena and accommodate for the small number of robots involved. We repeated these experiments 15 times for setup (i) and 10 times for setup (ii) with robots starting from the same positions and orientations, to allow for better comparison. We also performed the same experiments in simulation, with the same initial positions.

Fig. 8 shows that real-robot and simulated experiments follow analogous trends. In particular, we verified that for small-scale experiments with low redundancy factor (in these experiments it was set 1) the *outwards* algorithm has better performance than the *inwards* algorithm.

V. Related Work

Extensive literature exists on methods for connectivity preservation. Several recent works consist of motion control laws that include an estimate of the Fielder value. Yang *et al.* [9] introduced a decentralized algorithm to estimate the Fiedler value and use it to maintain connectivity while moving towards a target location. This algorithm was later refined by Sabattini *et al.* [10] and Williams *et al.* [11]. Further extensions include inter-robot collision avoidance [12] and multi-target exploration [6]. The main advantage of this family of approaches is that they allow navigation with arbitrary topologies. However, accurate decentralized computation of the Fiedler value is not easy in realistic settings in which messages might be lost due to communication interference [13]. In addition, computing the Fielder value in a decentralized manner involves network-wide power iteration methods [14], the slow convergence of which makes them suitable only for small teams of robots [15], [11]. It should also be noted that all of the above algorithms, with the exception of [12], have only been demonstrated in simulated environments.

A second family of methods select a communication subgraph and aim to preserve its edges through some form of global consensus. Hsieh *et al.* [16] devised a reactive control law based on radio signal and bandwidth estimation, in which links between robots can be activated and deactivated as the topology changes over time. Michael *et al.* [17] employed distributed consensus and auctions algorithms to establish which links to activate and deactivate over time. Cornejo *et al.* [18], [19] proposed a distributed algorithm

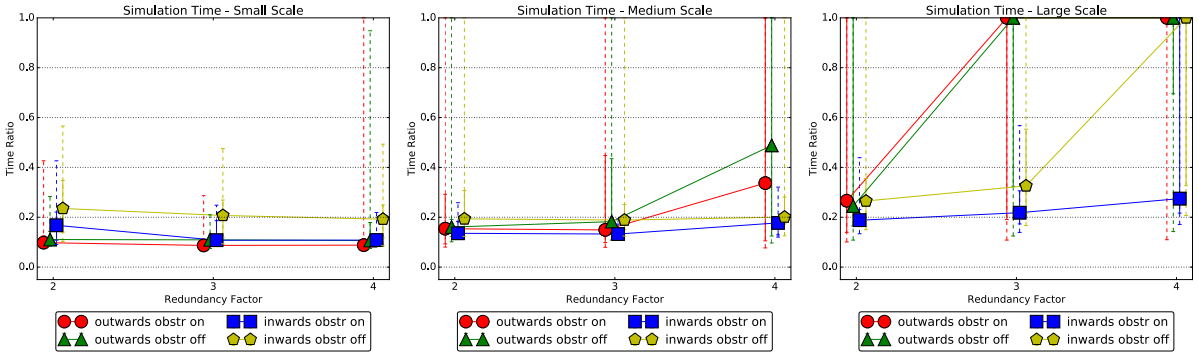

Fig. 6: Assessment of mission completion time in simulation.

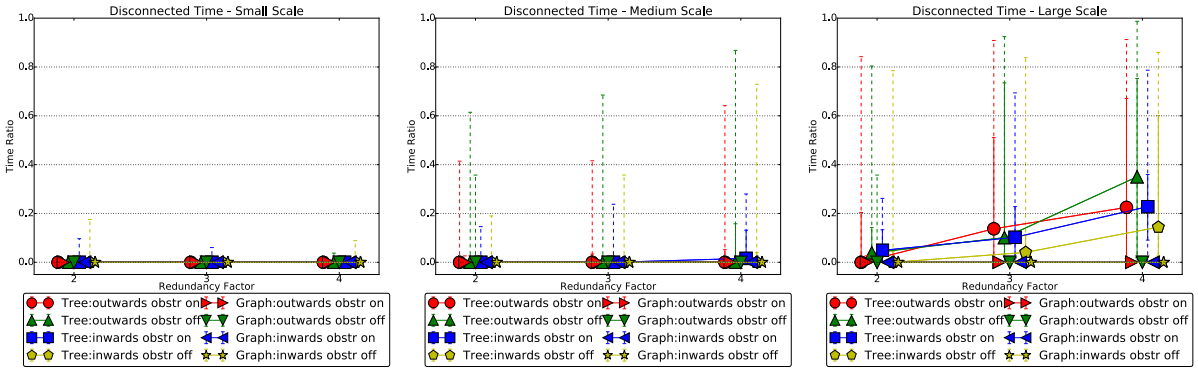

Fig. 7: Assessment of connectivity loss.

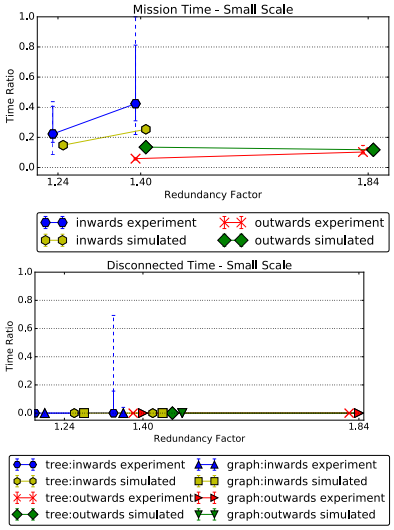

Fig. 8: Results of real-robot evaluation.

for link selection in which the robots undergo a number of motion rounds, during which the selected links must be preserved. Being based on achieving global consensus before any topology modification can be finalized, these algorithms are not scalable and work best when teams involve a small number of robots.

A third class of connectivity-preserving algorithms assumes that a certain structure is pre-existing. The dynamic structure is some form of logical tree, dynamically built and updated over the physical links of the robot network. Our work falls into this category. Krupke *et al.* [20] employed a Steiner tree as a pre-existing structure, and use spring-like virtual forces to balance connectivity and cohesiveness while reaching distant targets. A number of works, which constitute our main source of inspiration, utilized minimum spanning trees as structures to preserve. Aragues *et al.* [5] focused on a distributed coverage strategy with connectivity contraints, and proposed a method based on maintaining a network-wide minimum spanning tree. Analogously, Soleymani et al. [21] proposed a distributed approach that constructs and preserves a network-wide minimum spanning tree, allowing for tree switching. Schuresko *et al.* [22] studied a theoretical approach for distributed and robust switching between minimum spanning trees. All these works were only demonstrated in numerical simulations. The main advantage of these methods is the ease and speed with which spanning trees can be built and updated in a distributed manner. However, as discussed in this paper, spanning trees do not scale well with the number of robots involved.

VI. Conclusions

In this paper, we presented two algorithms to construct a long-range communication backbone that connects multiple distant target locations. The algorithms are decentralized and

based on the idea of constructing a logical tree over the set of physical network links.

We performed an extensive large set of experiments, both in simulation and with real robots, to assess the performance of the algorithms according to various experimental conditions. Our results show that, in small-scale scenarios, *outwards* tree growth, corresponding to spanning tree formation, is a viable approach. However, as the scale of the environment and the number of robots involved increase, a more directed, *inwards* growth from target locations towards the tree root, is a preferable approach.

Our results also show that, as the number of unnecessary robots increases, the benefit of redundancy is voided by the increased physical interpherence in navigation. While a better spare robot strategy could diminish this phenomenon, our results suggest that a more progressive approach to deployment might be a better idea.

Nonetheless, the presence of a reasonable number of spare robots offers the opportunity to tackle the problem of maintaining *persistent* long-range global connectivity despite individual limitations in the energy supply of individual robots. We plan to consider this scenario in future research.

In addition, possible extensions of our work include the presence of moving targets, rather than static ones, and the presence of obstacles in the environment.

REFERENCES

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] D. Goldsmith, *Voyage to the Milky Way: The Future of Space Exploration*. TV Books, NY, 1999.

[3] R. F. Rubio, "Mining: The challenge knocks on our door," *Mine Water and the Environment*, vol. 31, no. 1, pp. 69–73, 2012.

[4] D. P. Stormont, "Autonomous rescue robot swarms for first responders," in *Computational Intelligence for Homeland Security and Personal Safety, 2005. CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 151–157.

[5] R. Aragues, C. Sagues, and Y. Mezouar, "Triggered minimum spanning tree for distributed coverage with connectivity maintenance," in *2014 European Control Conference (ECC)*, 2014, pp. 1881–1887.

[6] T. Nestmeyer, P. R. Giordano, H. H. Bülthoff, and A. Franchi, "Decentralized simultaneous multi-target exploration using a connected network of multiple robots," *Autonomous Robots*, vol. 41, no. 4, pp. 989–1011, 2017.

[7] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298—-305, 1973.

[8] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[9] 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.

[10] L. Sabattini, N. Chopra, and C. Secchi, "On decentralized connectivity maintenance for mobile robotic systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 988–993, 2011.

[11] "Locally constrained connectivity control in mobile robot networks," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 901–906. [Online]. Available: http://ieeexplore.ieee.org/document/6630680/

[12] P. Robuffo Giordano, A. Franchi, C. Secchi, and B. HH, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[13] P. Di Lorenzo and S. Barbarossa, "Distributed Estimation and Control of Algebraic Connectivity over Random Graphs," pp. 1–13, sep 2013. [Online]. Available: http://arxiv.org/abs/1309.3200http://dx.doi.org/10.1109/TSP.2014.2355778

[14] A. Bertrand and M. Moonen, "Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks," in *Signal Processing*, vol. 93, no. 5, 2013, pp. 1106–1117.

[15] T. Sahai, A. Speranzon, and A. Banaszuk, "Hearing the clusters of a graph: A distributed algorithm," *Automatica*, vol. 48, no. 1, pp. 15–24, 2012.

[16] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 111–131, 2008.

[17] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Maintaining Connectivity in Mobile Robot Networks," *Springer Tracts in Advanced Robotics*, vol. 54, pp. 117–126, 2009.

[18] A. Cornejo, F. Kuhn, R. Ley-Wild, and N. Lynch, "Keeping mobile robot swarms connected," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5805 LNCS, pp. 496–511, 2009.

[19] A. Cornejo, "Local Distributed Algorithms for Multi-Robot Systems," *PhD Thesis*, 2012.

[20] D. Krupke, M. Ernestus, M. Hemmer, and S. P. Fekete, "Distributed cohesive control for robot swarms: Maintaining good connectivity in the presence of exterior forces," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, 2015, pp. 413–420.

[21] T. Soleymani, E. Garone, and M. Dorigo, "Distributed Predictive Connectivity Control for Double Integrator Agents based on a Receding Horizon Scheme," in *American Control Conference, ACC 2015*, 2015, pp. 1369–1374.

[22] M. Schuresko and J. Cortés, "Distributed tree rearrangements for reachability and robust connectivity," *SIAM Journal of Control Optimization*, vol. 50, no. 5, pp. 2588—-2620, 2012.