Dynamic Task Allocation in Robot Swarms with Limited Buffer and Energy Constraints

by

Janani Mohan

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

Janani

May 2018

APPROVED:

[illegible]

Dr. Carlo Pinciroli, Advisor

[illegible]

Michael A. Gennert, Professor

[illegible]

William R. Michalson, Professor

Abstract

Area exploration and information gathering are one of the fundamental problems in mobile robotics. Much of the current research in swarm robotics is aimed at developing practical solutions to this problem. Exploring large environments poses three main challenges. Firstly, there is the problem of limited connectivity among the robots. Secondly, each of the robots has a limited battery life which requires the robots to be recharged each time they are running out of charge. Lastly, the robots have limited memory to store data. In this work, we mainly focus on the memory and energy constraints of the robot swarm. The memory constraint forces the robots to travel to a centralized data collection center called sink, to deposit data each time their memory is full. The energy constraint forces the robots to travel to the charging station called dock to recharge when their battery level is low.

However, this navigation plan is inefficient in terms of energy and time. There is additional energy dissipation in depositing data at the centralized sink. Moreover, ample amount of time is spent in traveling from one end of the arena to the sink owing to the memory constraint. The goal is that the robots perform data gathering in the least time possible with the optimal use of energy. Both the energy and time spent while depositing data at the sink act as an additional overhead cost to this goal.

In this work, we propose to study an algorithm to tackle this scenario in a decentralized manner. We implement a dynamic task allocation algorithm which accomplishes the goal of exploration with data gathering by assigning roles to robots based on their memory buffer and energy levels. The algorithm assigns two sets of roles, to the entire group of robots, namely: Role A is the data gatherer, a robot which does the task of workspace exploration and data gathering, and Role B is data relayer, a robot which does the task of data transportation from data gatherers to the sink. By this division of labor, the robots dynamically decide which role to choose given the contradicting goals of maximizing data gathering and minimizing energy loss. The choice of a robot to perform the task of data gathering or data relaying is the key problem tackled in this work. We study the performance of the algorithm in terms of task distribution, time spent by the robots on each task and data throughput. We analyze the behavior of the robot swarm by varying the energy constraints, timeout parameter as well as strategies for relayer choice. We also test whether the algorithm is scalable.

Acknowledgements

I would like to express my sincere gratitude towards Professor Dr.Carlo Pinciroli for giving me the opportunity to work on the projects mentioned in this thesis and his invaluable support and direction throughout the duration of this work. I would also like to thank Professor William Michalson and Professor Michael Gennert for being a part of my thesis committee. NEST Lab has been a great place to collaborate and learn. Thanks to the team at the NEST Lab for their constant support during the course of my thesis. Finally, I would like to thank my family and friends for their motivation and encouragement to complete this work successfully.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A robot is a physical or virtual agent, which is programmed to perform specific tasks. The first mobile robotic system was developed in 1960's, SRI's "Shakey", a mobile robot equipped with a vision system and controlled by a computer the size of a room. Later in the late 1970's, we saw the development of the first mobile robot, Stanford Cart, to cross a chair-filled room without human assistance. Mobile robots were usually in the form of single-robot systems, where only one robot was responsible to participate in the task. The use of multi-robot systems offers significant advantages, among which are overcoming the single point of failure existing in single robot systems and having the ability to execute tasks which are beyond the capabilities of a single robot. In addition, multi-robot systems are able to deal with tasks which need to be executed in parallel.

Swarm robotics is a branch of robotics which consists of a large groups of robots deployed to achieve safety-critical as well as time-critical tasks. Robot swarms are implemented with inspiration from swarm behavior in nature namely, foraging techniques of ant colonies and collective motion of fish. Hence, nature suggests that efficient solutions to many real world problems exist, which are based on the

Figure 1.1: Multi-Robot systems

principles of swarm behaviors such as decentralization, no predefined roles, and cooperation. However, designing behaviors that follow these principles is HARD. Mobile robot swarms are usually suited for tasks, such as exploration, search and rescue, monitoring, mine exploration and material collection. In these contexts, using a multitude of robots is preferable to a single robot that performs all the work, since parallelism and fault tolerance increases. Additionally, the tasks mentioned are typically carried out in non-structured and potentially unknown environments which may change in time. This usually requires the robots to be (at least partially) autonomous, so that their behavior is flexible and it can dynamically adapt to a variety of environmental conditions.

In many robot swarm applications, the mission of the team is defined as a set of tasks that must be completed. Each task can usually be worked on by a variety of different robots; conversely, each robot can usually work on a variety of different tasks. In many applications, a task is decomposed into independent subtasks, hierarchical task trees, or roles either by a general autonomous planner or by the human designer. Independent subtasks or roles can be achieved concurrently, while subtasks in task trees are achieved according to their interdependence. Once the set of tasks or subtasks have been identified, the challenge is to determine the preferred mapping of robots to tasks (or subtasks). This is the *task allocation problem* [Bhadauria et al., 2011]. In this work, one such self-organized, decentralized task allocation scheme is proposed which considers the energy and memory limitations of robot swarm for decision making.

The rest of this chapter is organized as follows, Sec. 1.1 defines the motivation for this work, Sec. 1.2 introduces the problem of interest and Sec. 1.3 emphasizes the scientific contribution of this work. Finally, Sec. 1.4 gives the organization of rest of the work.

1.1 Context and Motivation

Most of the research conducted in the field of task allocation discusses techniques to minimize a cost function which is formulated based on task completion time, task distribution among robots and also factors such as net distance covered etc. In this research we consider a generic application of robot swarms; the task of data gathering by area exploration. The data to be gathered can be food as inspired by ant foraging or environmental data sensing applications such as measuring pollutant content etc. In order to accomplish tasks involving exploration and sensing, the swarm of robots can be divided into subgroups, with each subgroup performing a set of different tasks. The dynamic task allocation proposed in this research work studies the task switching tendency of robots given a set of contradicting goals. In real world scenarios, the entire task of area exploration and data gathering needs to be completed in a limited amount of time which each robot trying to gather as much data as possible with the constraints of limited energy and memory capacity of each robot.

When there are many tasks to be accomplished by a group of robots with limited resources, the strategy would be to perform multi-robot task allocation. The task allocation problem in swarm robotics can be divided into many phases: a. Decomposing the high-level tasks into simpler tasks attainable by the robots, b. Endowing the swarm with means to establish the number of robots assigned to each task, and c. Finally, switching the tasks between each robot in real-time. An efficient algorithm which attempts to solve the task allocation problem would have to address resource constraints of each robot and also yield a fault tolerant solution. In the current literature, such an algorithm does not exist, and this is the key problem tackled in this work.

Finding an appropriate mechanism for task assignment is also a challenging problem because task allocation is influenced by many factors. First of all, the complexity of tasks plays a major role. Tasks can be constrained by arbitrarily difficult dependencies. An example for such a dependency is a sequential order. In this case, robots have to wait for the completion of some tasks before others can be allocated. Another example is the definition of special factors that specify which number and which kinds of robots are needed to accomplish a task. Furthermore, a robotic swarm has to adapt its behavior to the environment. If the environment is dynamic, the robots will encounter situations that designers might have not considered, and non-trivial adaptation algorithms might be required. Lastly, in swarm robotics the physical robots will likely influence each other, mostly by their presence itself. This influence will constrain beneficial allocation patterns.

Previous researches in the domain of multi-robot task allocation have covered many aspects of this problem. [Gerkey and Mataric, 2003] has reduced the multi-robot task allocation problem to an instance of the Optimal Assignment Problem

(OAP), in which the authors have formulated architectures which address three characteristics in multi-robot system namely : computational requirements, communication requirements and task considerations. [McLurkin and Yamins, 2005] proposed a dynamic task assignment scheme in robot swarms based on a specified global task distribution . All robots are given the target distribution by a centralized source in the form of a vector of normalized relative subgroup sizes, e.g. the tuple (1/6; 1/3; 1/2) for a system with three subgroups. The algorithms are designed to address scenarios such as limited communication with neighbors and also time constraints. [Khaluf, 2014] in his work, concentrates on developing task allocation strategies which assign robots swarms to time-constrained tasks under the condition of the task deadlines. The author has designed task allocation strategies to address both hard and soft time constraints .

[Lauri et al., 2017] addresses a similar problem of distributed data gathering for a swarm of robots with intermittent communication. [Guo and Zavlanos, 2017] proposes an online coordination scheme for multiple robots under local data-gathering tasks under both communication and buffer size constraints. The key motivation for this thesis work is derived from these papers.In this work, we are interested in exploring the impact of energy constraints in addition to the buffer constraints and how it impacts the task coordination behavior of the robots while they perform area exploration and information gathering.

1.2 Problem Statement

In this thesis, we focus on problem of area exploration and data gathering using a homogeneous swarm of robot which have limited buffer and energy levels as depicted in Fig. 1.2. These constraints force the robots to travel to Sink location when their

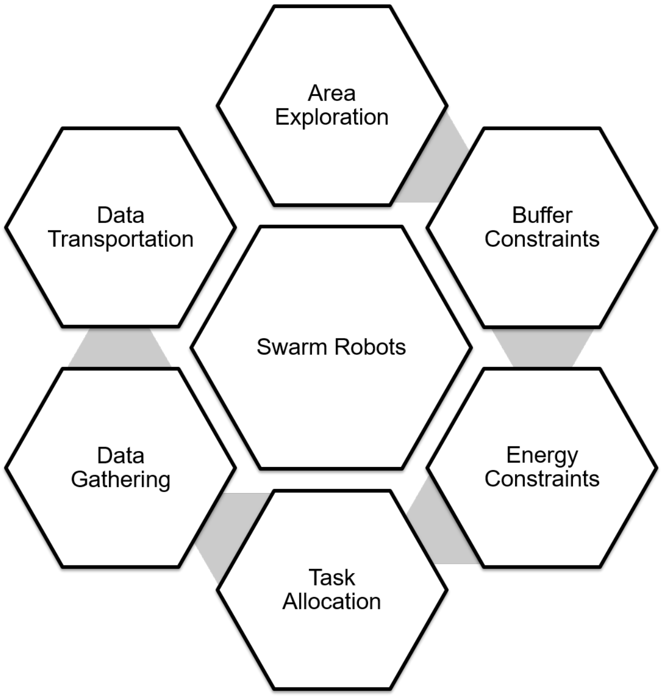


Figure 1.2: Main Idea of the Thesis

memory becomes full and to the Dock location when their energy is about to get depleted as depicted in the Fig. 1.3. This behavior impacts the efficiency of the robots as they explore deeper into the environment for data gathering. In order to improve the efficiency and task completion time of the group of robots, we propose an approach of dynamic task allocation. The main goals of the algorithm are as follows:

Global Goal

- Ensure high throughput without data loss.
- Ensure the entire area is covered by the robot swarm.
- All robot should collect as much data as possible.

Individual Robot Goal

- Each robot should minimize energy loss.

- Each robot should never reach zero battery level.

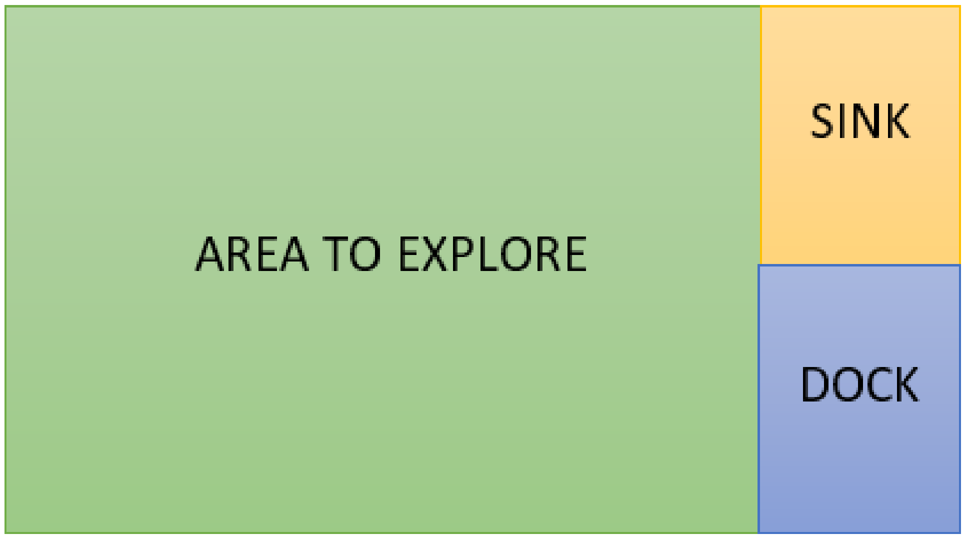


Figure 1.3: Arena considered for robot exploration

As we can see the global as well as the individual level goals are contradicting given that the main contribution for energy loss is due to navigation and sensing tasks. In order to satisfy these contradicting goals, the proposed algorithm defines a technique which dynamically allocates two roles to the robots considering the limited buffer and energy constraints of robots in the swarm. The entire group of robots can choose any of the following roles:

- Gatherers : These are Data Gatherer robots that explore the environment and gather data. As they move further from the centralized sink and explore deeper into the environment, they seek the help of the other robots called Relayers who work as relay robots transporting data from Gatherers to the sink.

- Relayers : These are Data Relayer robots that perform the task of transportation of data from Gatherers to the sink when they receive a relay request from Gatherers. This specialized functionality ensures higher data throughput at the sink and saves the energy and time of Gatherers.

In this work, we mainly study how the robots choose between the tasks of "data gathering" and "data transportation" in order to satisfy both the global and individual level goals. The role switching pattern and the selection criteria of Relayers are also studied in this work. The algorithm is designed to work with robots which can share information with its neighbors within a limited range of connectivity.

1.3 Scientific Contribution

The main contribution of this thesis can be summarized as follows: The development of autonomous task allocation strategy that assign robot swarms to parallel decentralized tasks based on their energy and memory constraints. Other contributions are listed below:

- Contradicting goals scenario : Developing an algorithm to understand the emergent behavior of robot swarms when presented with contradicting goals at global and individual levels.

- Dynamic Role Switching scenario : The algorithm assigns tasks to robot based on the energy and buffer constraints, thereby using neighbor robots to assist in task completion in a coordinated manner.

- Novelty in terms of constraint consideration: This work proposes a task allocation scheme which considers all the constraints including communication, memory and energy faced by a robot swarm in a real-world scenario.

- Extension of the algorithm to energy-sharing robot scenario: The same algorithm can be extended to share energy among robots in case one of the robots is critical in energy and another robot then can share its resource.

1.4 Organization of the Thesis

The contents of this thesis are organized as follows. In Chapter 2, we describe the concepts and principles of swarm intelligence and swarm robotics. The idea of task allocation is also illustrated by listing its benefits through examples taken from nature Finally, a review of related work on the topic, focusing on mobile robots is discussed. In Chapter 3, we illustrate task allocation problem as studied in this work and present our approach for dynamic task allocation accomplished by two teams of robots. We provide general definitions and concepts that are used throughout this thesis and describe the core algorithm which forms the basis of our approach. In the studied setup, the given task is pre-partitioned into a sequence of several sub-tasks, defined a priori, and the robots decide how to coordinate the tasks based on their data buffer and energy constraints. In Chapter 4, an overview of the software tools used to perform the research work is presented. The chapter introduces the features of these tools that are relevant to our experiments. Also we present a scenario in which the robots implement the proposed algorithm. Chapter 4 also reports the experiments performed and briefly summarizes the results of this thesis and gives a short outlook on some current research directions that intends to tackle the memory size, connectivity and energy limitations in multi-robot systems. Finally, in Chapter 5, we summarize the presented work, draw some conclusions and propose the scope for future work.

Chapter 2

Related Work

In this chapter, we discuss the research context related to the work presented in this work. In Sec. 2.1 we present some of the schemes in Swarm Robotics and the different algorithms for area exploration using robot swarms. In Sec. 2.2, we dive deeper into task allocation, the main subject of this work: we highlight its benefits and review the state of the art on the topic.

2.1 Swarm Robotics Background

In many situations,systems composed of a multitude of cooperating robots is preferable to one in which a single robot performs all the tasks. This is the case, for example, when the tasks are too complex for a single robot to accomplish, or when producing a number of simple robots is cheaper than producing a single robot capable of performing all the tasks [Cao et al., 1997]. However, as the number of robots increases, the complexity of the system also increases. Relying on centralized control and global communication becomes infeasible and one has to resort to other approaches.

An approach that has been applied is to implement and control robotic systems by taking inspiration from natural swarm intelligence systems. The application of swarm intelligence principles to collective robotics gave birth to swarm robotics. A swarm robotics system is one in which a large number of relatively simple robots cooperate to solve tasks that cannot be tackled efficiently by single individuals. Each robot acts autonomously on the basis of its local perception of the surrounding environment. Interactions and communication between robots are also local and are often mediated by the environment. One of the main challenges when designing swarm robotics systems is to understand the effect of the behavior of individual robots on the global behavior of the swarm.

Area exploration of an unknown environment is one of the major task of Multi-Robot systems. Many works have been proposed to coordinate multi-robots to accomplish exploration missions based on multi-agent systems techniques. Some of these works particularly focus on multi-robot exploration under communication constraint . In their paper, [Pal et al., 2012] proposes a solution to explore an open area while keeping robots close enough to ensure connectivity. In [Yamauchi, 1998], robots try to build and use a global map for the exploration. The postulates of the Yamauchi's work is the following: to speed up the exploration, robots have to gain new information about the environment. Therefore they have to move towards the boundary between open space and unexplored area. While there has been significant progress for multi-robot exploration of an unknown area, little attention has been given to the constraints such as energy and memory constraints of the robots. In the case of area exploration and data gathering, the global task is to explore as much area as possible and perform information gathering in each of the locations. The actual challenge is then to decide how to coordinate the robot swarms by assigning individual tasks which would give rise to an emergent behavior which accomplishes the global task. In this work, we consider the energy and memory constraints of each robot in the swarm to design a task allocation scheme to accomplish complete area exploration and information collection .

2.2 Task Allocation Background

Task allocation is one of the main focus areas in swarm robotics, which has gained a lot of interest in the last years. The efforts in task allocation algorithms dealt with finding efficient heuristics and allocation strategies such as in [Chevaleyre et al., 2005] and in [Endriss et al., 2006]. Task allocation can be observed in nature, as by ant and bee colonies ([Bonabeau et al., 1997] , [Anderson and Ratnieks, 2000]; [Hart and Ratnieks, 2001]) and wasps ([Akre et al., 1976]). In such colonies, we can notice how the members allocate different tasks among themselves. For example a part of the colony can perform foraging while another part looks after the larvae and they get specialized over time on their tasks, which improves the performance [Pini et al., 2013].

Task partitioning is a way of organizing work that consists in decomposing tasks into sequences of smaller sub-tasks ([Pini et al., 2011]; [Krieger et al., 2000]). Task partitioning is usually coupled with task allocation strategies: when a task is partitioned into multiple sub-tasks, workers must be assigned to each of these sub-tasks [Lauri et al., 2017]. The key difference between task partitioning and task allocation is that task allocation is a way of organizing the workforce to perform specific subtasks, while task partitioning organizes the way the work itself is performed ([Anderson and Ratnieks, 2000]).

Solutions proposed in the literature for task allocation in multi-robotic systems in general, can be classified into three broad categories: centralized, negotiation-based and self-organized. Centralized techniques assume the presence of a central coordinator responsible for the allocation of the tasks to agents (robots). The single point of control causes problems related to the robustness of the solution and to the availability of such central control over any application environment. Self-organized systems, on the contrary, are constituted by peers that take decisions autonomously, through limited negotiations with other peers and without any central point of control. This kind of systems are generally less prone to catastrophic failures and are considered as a better approach, when rapid adaptation to changes in the environment, is required [Nouyan et al., 2009]. Most of these studies tackle simple scenarios without task interdependencies [Dahl et al., 2009]. In swarm robotics, self-organized task allocation is often based on the response threshold model first introduced by [Bonabeau et al., 1999]. Threshold-based task allocation has been successfully employed in tasks such as foraging ([Labella et al., 2004], [Lauri et al., 2017]) and object clustering ([Agassounon and Martinoli, 2002]). In auction-based strategies, the robots bid on the announced tasks according to the task characteristics and to their relative capabilities [Hoeing et al., 2007]. Robots with the highest bid win the allocation to the announced task. The individual robots are not aware of the whole system and can only communicate with their neighborhood.

Most of the recent works in the field of multi-robot task allocation address task allocation among multi-robot systems under constraints. [Guo and Zavlanos, 2017] in their work, propose a task allocation algorithm based on LTL (Linear Temporal Logic) for assigning tasks for two different types of robots namely, Type A and Type B robots under buffer constraints and intermittent robot communication. The authors address the problem of coordinating the actions of a team of robots with periodic communication capability executing an information gathering task. The problem is formulated as a multi-agent optimal decision-making problem with an information theoretic objective function. The authors prove that appropriate techniques for solving decentralized partially observable Markov decision processes (DecPOMDPs) are applicable in such information gathering problems. [Bhadauria et al., 2011] in their work , tackle the problem of planning paths of multiple robots so as to collect the data from all sensors in the least amount of time. They present an optimal path planning algorithm which ensures task completion within time constraints.

The novelty in this work is the dynamic task allocation by dividing robot swarm into teams for data gathering and data transporting based on their buffer capacity and energy level. This work allows the robot swarm to effectively divide their work at the same time complete the tasks within the time bound and deliver consistent data without loss.

2.3 Summary

In this chapter, the research conducted in the field of swarm robotics and task allocation relevant to this work are discussed.

- **Swarm Robotics Background**: We present the concept of swarm robotics and some of the earlier works with swarm robotic systems.
- **Task Allocation Background**: We discuss the research specific to the field of task allocation and present their limitations.

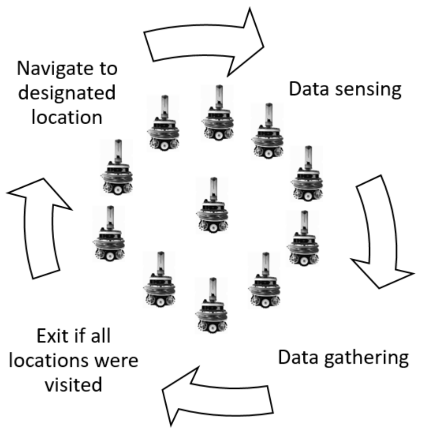
Chapter 3

Methodology

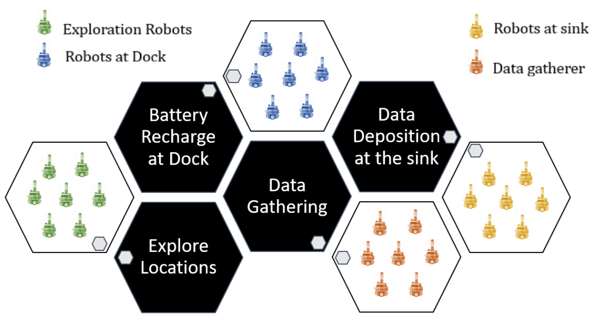
In this chapter, we present in detail the algorithm towards achieving dynamic task allocation under energy and memory constraints. This chapter is organized as follows. In Sec. 3.1, we present the dynamic task allocation problem studied in this work. In Sec. 3.2, we define the sets of tasks and roles assigned to the robots and how these roles help to achieve the global goal of area exploration and information gathering. In Sec. 3.3,the state machine of each robot and the algorithms used to implement the state machines are explained in detail. In Sec. 3.4, we present the dynamic task allocation scheme and how the role switching is executed, We also discuss the modeling of the memory and energy constraints and their characteristics with time.

3.1 Requirements and Problem Formulation

Most of the multi-robot applications, require the robots to perform data sensing namely, create a map of the environment, or locate targets or even measure a specific parameter in the environment. For each of these data sensing tasks, the robots need to explore the region, collect the data and store the data in its local buffer. These robots continue this work flow until the entire environment is explored and all the data locations have been identified. A typical work flow of such a task is shown in Fig. 3.1(a). In this work, we now consider a scenario when robots are more realistic with memory and energy limitations. A possible approach to tackle these constraints would be to have centralized sink and dock locations which enables the data deposition and battery recharge for the robots. Under these constraints, each robot explores the environment but has to travel back to centralized sink each time its buffer gets full. Additionally, the limited battery life forces the robot to get recharged at a centralized dock when the battery level is low. A centralized sink/dock acts as an additional overhead as far as the distance traveled and task completion time is concerned. This navigation scheme is inefficient in terms of energy and time. The Fig. 3.1(b) shows the pictorial representation of this approach. This work aims to study possible ways to improve this scenario by introducing dynamic task allocation to the swarm of robots.


Navigate to designated location
Data sensing
Data gathering
Exit if all locations were visited

(a) Work flow under no constraints



(b) Workflow under Energy and Memory constraints

Figure 3.1: Typical Work flow of Area Exploration using Robot Swarms

Dynamic task allocation is the process by which the robots determine which task to do based on a set of criteria available at that given time frame. In this work, we aim to reduce the additional navigation costs and energy loss due to data deposition cycles.Fig. 3.2 shows the environment under consideration where the sink and dock locations are situated at one end of the arena. All robots begin the exploration task from these points. They start exploring different locations within the arena and collect data as and when data is available. Once the memory buffer of the robot becomes full , they travel to the sink for data deposition. Each robot has limited energy and they travel to the dock when battery level is near zero.We apply dynamic task allocation principles in order to allow data deposition to be divided among multiple robots which is explained in Sec. 3.4. The dynamic task allocation scheme should meet the following requirements:

- Arena coverage : All locations in the arena should be visited atleast once by the robot swarm.
- Data gathering : Data available at different locations should be identified along with its location in the arena.
- No Dead Robots : None of the robots should reach zero energy level.
- No Data Loss : There should not be memory overflow for any robot.

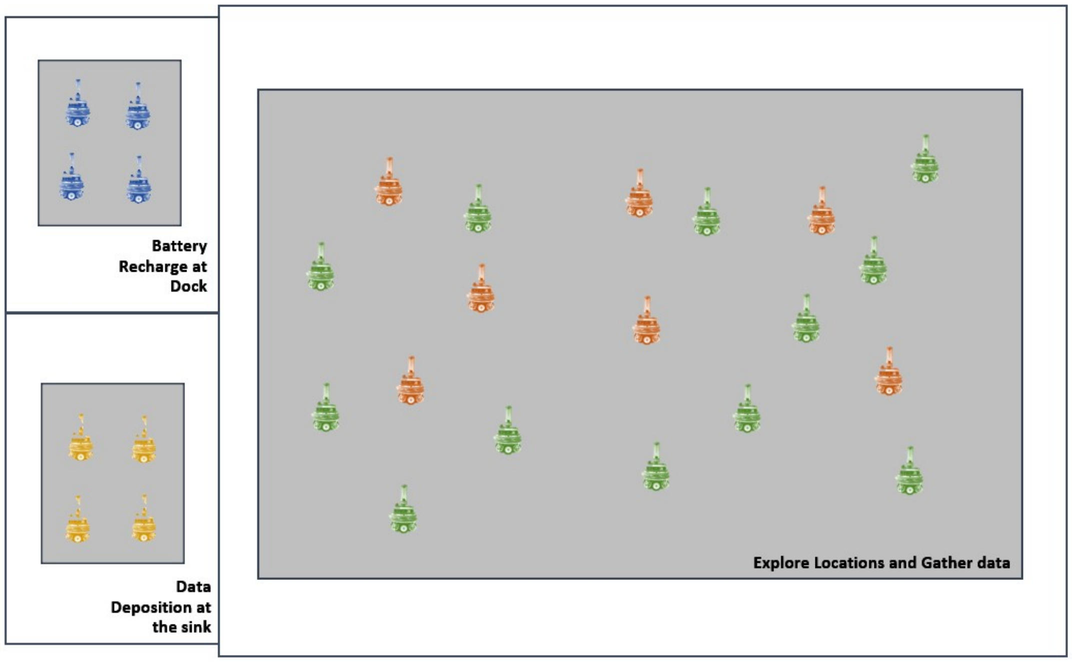

Battery Recharge at Dock
Data Deposition at the sink
Explore Locations and Gather data

Figure 3.2: Arena under study for exploration and data gathering

3.2 Tasks and Roles

In this work, we focus on assigning a set of tasks to a swarm of homogeneous robots, under limited buffer and energy constraints, to accomplish the goal of area exploration and information gathering. The robots can choose between two types of tasks : Task A (data gathering) and Task B (data transportation). Task A robots would perform data gathering by exploring the environment whereas Task B robots (data relayers) would perform data relaying from Task A robots (data gatherers) to the sink. The decision making lies in deciding which task each of the robots needs to perform at any given point of time. A number of factors contribute for this decision making including the net throughput of data gatherer robots, the distance of each robot from the sink, the buffer constraints as well as the energy constraints of each robot. The coordination of the data gatherers and data relayer needs to be accomplished by enabling inter-robot communication. In the following subsections, we will be discussing in detail about the tasks and the roles played by each robot while performing the tasks.

3.2.1 Data Gathering Task

The global goal for each robot is to navigate as much as locations as possible in the given arena and gather the data at different locations of the arena. The robots are not aware of where the data is located. Hence they need to explore the location and search for data in that location. The task of navigating to each location, searching for data and storing the sensed data to the local memory buffer is termed as *data gathering task*. In this work, each of the robots are assigned a buffer capacity of N units of data and the robot need to accomplish the global task of exploring the environment. The environment is divided into multiple grids and specific grids contain information of interest. The robots perform data gathering and store the data into their local memory buffer. Each grid of interest exactly has information of N units of data which implies that the data buffer for each robot is full after gathering data from a single sub-grid of interest. Once the buffer is full, the data gatherer robots has to decide whether they need to travel to the sink to deposit data or whether they need to request data relayers for support. Fig. 3.3 describes the task of data gathering and the roles played by a data gatherer robot. Data gatherers switch to the role of data exchangers when they request help from data relayers. The arena in this work, has specified data exchange locations for the data gatherers to meet the data relayers for data transform. The data gathering and role switching algorithm is explained in detail in Sec. 3.4.

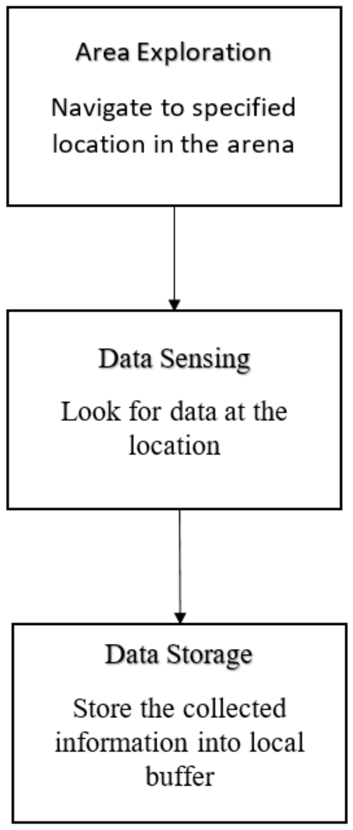

Area Exploration
Navigate to specified location in the arena
Data Sensing
Look for data at the location
Data Storage
Store the collected information into local buffer

Figure 3.3: Data Gathering Task Flow

3.2.2 Data Transportation Task

The data transportation task allows data gatherer robots to seek the help of other robots called data relayers to relay the data to the sink. The data relayer robots choose the data transportation task upon receiving a "relay" request from the data gatherers. Since data gatherers need to transfer the data to the data relayers, in this work we introduce specific data exchange grids. Data exchange grids are locations in the arena where the data gatherers meet with data relayers to transfer data. The data relayers upon receiving the information from data gatherers proceed to the sink for data deposition. The *data transportation task* is initiated upon receiving the relay request from data gatherers. Once the relay request is received, other robots in the swarm based on their availability sends availability responses. The data gatherer then choose one of the potential data relayers. The chosen relayer then meets the data gatherer at the data exchange, receive the data and finally proceeds to deposit data at the sink. Fig. 3.4 describes the sequential flow of the states of the robot while performing the data transportation task. The criteria for choosing data transportation task is presented in detail in Sec. 3.4.

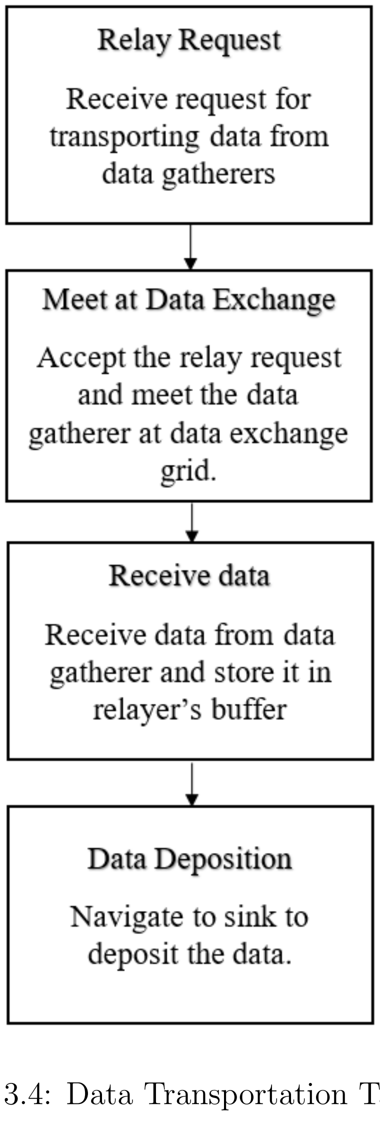

Relay Request
Receive request for transporting data from data gatherers
Meet at Data Exchange
Accept the relay request and meet the data gatherer at data exchange grid.
Receive data
Receive data from data gatherer and store it in relayer's buffer
Data Deposition
Navigate to sink to deposit the data.

Figure 3.4: Data Transportation Task Flow

3.3 State Machines

In this section, we understand the different states of the robots as they execute the global task of area exploration and information gathering. In the following subsections, we will define the states specific to each task and explain their algorithm in detail.

3.3.1 States specific to Data Gathering

In Sec. 3.2, we saw the two tasks performed by each robot in the swarm, namely data gathering and data transportation tasks. In the data gathering task, the robots execute three states.The three states are as follows: a. Navigation, b. Obstacle Avoidance and c. Data Sensing. The robots performing data gathering tasks executes each state in sequential order. Each robot chooses a specific location to navigate, while traveling to the location obstacle avoidance is implemented ,and once the robot has reached the specific location it performs data sensing. Each of these states are implemented using algorithms which are explained in the following sections.

Navigation Algorithm

In this work, we consider the arena to be divided into multiple grids and each grid specifies a location in the arena which is specified by x-y coordinates.Each grid has a specific area and all of grids are identical. Each robot is a differential drive robot equipped with range and bearing sensors to be aware of their current location in the arena. Additionally, the robots are equipped with sensors such as proximity sensor and an omni-directional camera which allows them to avoid obstacles and collect visual information respectively.All the robots start at specified sink/dock location in the beginning of exploration activity. These locations are fixed. In this work, we have defined information of interest to be colored blobs. Figure 2 illustrates the environment chosen for this work.

During each execution cycle, the robots navigate through the grids and look out for information in this case , would be the colored blobs. Each robot is assigned a grid using the nearest neighbor algorithm. The grid centers are chosen as destination points and each grid is marked as visited if the robot reaches the grid center.Alg. 1 illustrates the pseudo-code of the navigation algorithm.

Algorithm 1 Navigation

```
1: procedure NAVIGATE
2:     allLocations ← all grid locations
3:     robotLoc ← robot_xy_location
4:     nextLoc ← nearestNeighbor(robotLoc)
5:     gotoLocation(nextLoc)
6: nearestNeighbor(robotLoc):
7: minDist ← arenaDiagonal
8: for loc loop through allLocations do
9:     currDist = EuclideanDist(loc,robotLoc)
10:    if (currDist < minDist) then
11:        minDist = currDist
12:        newLoc = loc
13: return newLoc
14: EuclideanDist(a,b):
15: currDist ← $\sqrt{a^2 + b^2}$
16: return currDist
```

Obstacle Avoidance Algorithm

Collision avoidance is vital for any robot. The robot makes use of proximity sensors arranged uniformly in its peripherical circumference to avoid collisions. The proximity sensor informs about the range and bearing of obstacles up to 10 cm away from the robot. Readings from all sensors are added together to get the resultant vector of collision. The robots are then directed to move away from this vector of collision. The proximity sensor displays a value of 1 if robot is in collision at that sensor location and we use this to sense an obstacle in the proximity of the robot. We identify the location of the obstacles with respect to the robot with the proximity sensor's angle with respect to robot's heading. Alg. 2 explains the obstacle avoidance logic implemented in this work. We consider obstacles is the form of dynamic robots and no static obstacles are considered.

Algorithm 2 Obstacle Avoidance

```
 1: procedure AVOID_OBSTACLE
 2:     inCollision,collisionAngle ← collisionDetect(nextLoc)
 3:     gotoLocation(nextLoc,inCollision,collisionAngle)
 4: collisionDetect(nextLoc):
 5: minDist ← arenaDiagonal
 6: for prox loop through proximity_readings do
 7:     proximity_value_sum += prox.value
 8:     if (prox.value > 0) then
 9:         collisionAngle += prox.angle
10:         inCollision = 1
11:     else
12:         inCollision = 0
13: return inCollision,collisionAngle
14: gotoLocation(loc, collisionFlag, collisionAngle):
15: if (collisionFlag > 0) then
16:     if (collisionAngle > 0) then
17:         rotate clockwise
18:     else
19:         rotate anti-clockwise
20: else
21:     gotoLocation(loc)
```

Data Sensing Algorithm

In this work, the data points are colored blobs which are randomly distributed across the arena. The robot upon reaching a specific grid in the arena performs data sensing using the omni-directional camera of the robot. The omni-directional camera allows the robot to view colored objects within the distance specified. We consider three colors for the blobs, namely, red, blue and green. Alg. 3 explains the data sensing algorithm implemented in this work.

Algorithm 3 Data Sensing

```
1: procedure SENSE_DATA
2:     $blobCount \leftarrow$ cameraImageOutput
3:     if ($blobCount > 0$) then
4:         rotate in place to check for the blobs
5:         locate target blob within 1 grid size
6:         determine the color of the blob
7:         update memory with tuple <blob_color,grid_num>
8:     else
9:         no blob found
```

3.3.2 States specific to Data Transportation

Data Transportation mainly deals with two states: data exchange and data relay. During data exchange the robot accepts the relay request from the data gatherer robot, sends an available response based on its availability. The data gatherer chooses the relayer from the list of relayer who had offered to help. The chosen relayer then travels to a data exchange grid assigned to it by the data gatherer. The data exchange procedure involves robot-robot communication at close proximity to ensure faster transfer rate for large amount of data. Once the data is exchanged, the robot now executes data relay state. In this state, the data relayer travels for the data deposition to sink. This involves choosing the nearest sink location and traveling to that location. Once the robot reaches the sink location, it uploads the data to a common server (in our case a generic table) which is shared among all robots. Alg. 4 and Alg. 5 represents the pseudo-code for the data exchange and data deposit states.

Algorithm 4 Data Exchange

```
1: procedure DATA_EXCHANGE
2:     Accept relay request
3:     Send availability response
4:     Receive acceptance response
5:     exchangeLoc ← msgFromGatherer
6:     locReached = gotoLocation(exchangeLoc)
7:     if (locReached == 1) then
8:         request data from gatherer
9:         store tuple <blob_color,grid_num> into local memory
```

Algorithm 5 Data Relaying

```
1: procedure RELAY_DATA
2:     sinkLoc ← nearestSink(robotLoc)
3:     locReached = gotoLocation(sinkLoc)
4:     if (locReached == 1) then
5:         upload data to server
6:         clear local memory
```

3.3.3 States specific to Energy Management

In order to address the memory constraints of the robot, each robot executes an energy management state. The robots need to continuously monitor their energy levels to ensure that they never run out of charge and will reach the dock location for recharge before their battery level is zero. The energy management state monitors the current battery level and estimates the new battery level based on the distance to the next location. It then calculates whether the energy is sufficient to go to the dock after reaching the next location. If the energy is insufficient, the robot chooses to go to dock for recharge. This logic ensures that the robot never runs out of charge. The battery monitoring is done at every execution cycle. Alg. 6 briefly explains the energy management logic in this work.

Algorithm 6 Battery Monitoring

```
1: procedure BATTERY_RECHARGE
2:     nextLoc ← nearestNeighbor(robotLoc)
3:     netDisttoSink = (disttoNextLoc) + (distfromNextLoctoSink)
4:     estimatedBattLevel = dischargeRate ∗ netDisttoSink
5:     if ( estimatedBattLevel <= 0) then
6:         gotoLocation(nearestDockLoc)
7:     else
8:         gotoLocation(nextLoc)
```

3.4 Dynamic Task Allocation Scheme

In Sec. 3.3, we introduced the different states within the tasks of the robot swarm. In this section, we present the mathematical model of the buffer and energy constraints.We discuss the criteria to choose a particular task and how the division of labor is implemented with and without energy constraints. We also present the flow charts for the state machine execution and explain how the task switching takes place dynamically.

3.4.1 Constraint Modeling

In this work, we consider two sets of constraints: memory constraints and energy constraints. The total memory represents the maximum amount of data the robot can store at any given instance of time. The minimum energy is assumed to be the amount of energy needed to complete one round trip along the diagonal of the arena. In this section, we define the modeling of the constraints and explain the characteristics over time.

Memory constraints

The buffer for each robot is limited and it can store a maximum of N units of data. The arena considered in this work is divided into multiple grids of equal area. It is assumed that the data available at each grid is N units. Hence, we can see that the memory buffer of each robot becomes Full when it gathers data from a single grid. At this point, the task allocation scheme decides whether the robot needs to request for a data relayer or not. Fig. 3.5 shows the distribution of data in a 15x15 grid and the variation of the buffer status based on the data present in the grid.

$$Buffer\_status = \begin{cases} Full & \textit{if} \text{ data found in the grid} \\ Empty & \textit{otherwise} \end{cases}$$

Energy constraints

The energy of the robot is the battery level at an instance of time. In this work, the energy of all robots is specified by Q units at the beginning of each experiment run. It is assumed that the charging time for all the robots are the same and its

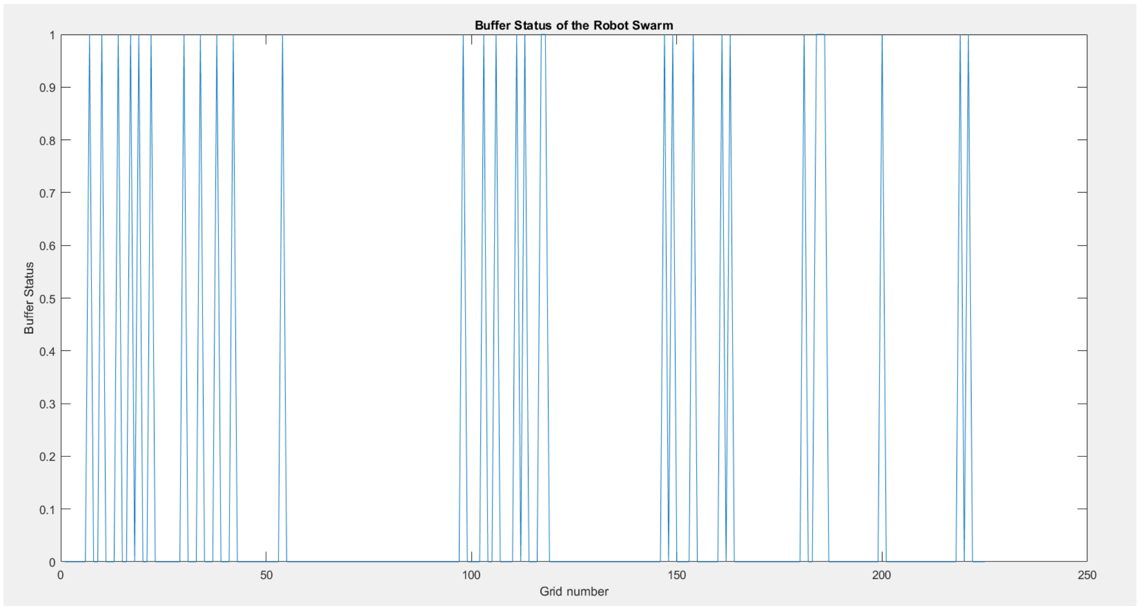


Figure 3.5: Buffer Status of the Robot Swarm

instantaneous upon reaching the dock. The rate of battery discharge is variable and is given by m percent which is dependent on the distance traveled by each robot. The robots needs to travel to the charging station each time its charge is slightly more than a single trip time, to be recharged to full. The energy discharge equation is given as follows:

$$e_{new} = e_{old} - \lambda_{pos}(\Delta_{pos})$$

The value of λ_{pos} is variable and it represents the battery discharge rate.The Fig. 3.6 shows the battery model for different values of λ_{pos}. The value of Δ_{pos} gives the distance traveled by the robot at any given instance of time. For simplicity, we have assumed a linear battery model for the robot with λ_{pos} being integer values.

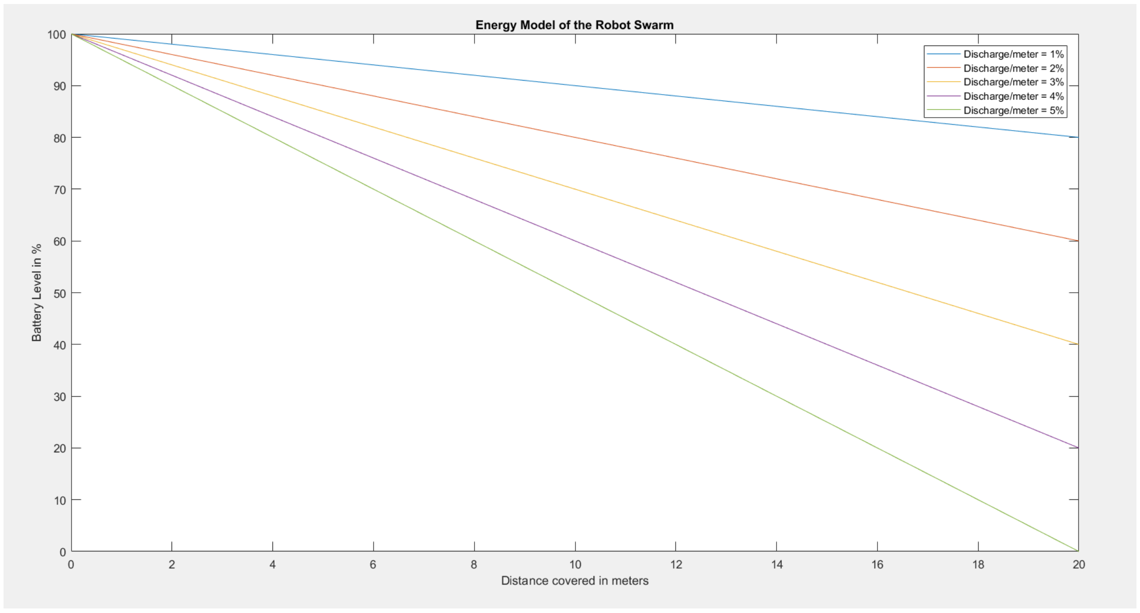


Figure 3.6: Battery model for different discharge rates

3.4.2 Strategy

The two main tasks to accomplish are : data gathering and data transportation. Given the tasks to perform, there are many ways to organize its execution. It can be based on the time spent on each task or cost for each task. In this work, in order to ensure that there is a smooth transition between the two tasks, we define a standard criteria to choose a particular task at any given instance of time.

Criteria for Data Gathering task

The main goal of each robot is data gathering. This is given the highest priority unless:

- There is a relay request from other data gatherers.
- Battery is low and there is a need to recharge.
- Memory is full and there is a need for either data deposition or data exchange.

Criteria for Data Transportation task

The role of a data relayer is triggered upon a "relay_request" from Data gatherers. A robot can be a data relayer provided:

- There is an open relay request from other data gatherers.
- Memory buffer of the robot is empty.
- The robot has completed the task it is currently executing.

Criteria for Data gatherers to request data relay

Data gatherers request data relay support from other robots only under the following conditions:

- The data gatherers are at distance of atleast half the arena away from the sink. This ensures that the robot need not dispense more energy just for data deposition, rather can seek the help of other robots by exchanging data at the data exchange grids.
- Memory buffer of the robot is full.
- Battery of the robot is not nearing zero.

Criteria for choosing a Data Relayer

Once the relay request is sent to other robots in the swarm, the data gatherers opens a recruitment slot (for a specific time)for getting availability requests from other robots. Once the time has lapsed, the data gatherer robots applies one of the two strategies to choose its data relayer robot from its recruitment applicants:

- Maximum Energy criteria : The applicant robot with the maximum energy is chosen as the data relayer and is requested to meet at a data exchange grid.

- Nearest Relayer criteria : The applicant robot which is nearest to the data exchange grid is chosen for data exchange.

3.4.3 Design

The dynamic task allocation algorithm is implemented as Finite State Machines. In this subsection, we present the state diagrams for each state and explain the overall task allocation scheme under various constraints.

State Diagrams

- Task Dispatch

  The task dispatch is executed in each robot of the swarm at every time step. The next state of the robot is determined in the task dispatcher based on changes in the energy and data memory of the robot while executing the previous state. The task dispatcher takes care of assigning roles to the robots based on the criteria explained in Sec. 3.4.2. Fig. 3.7 represents the decision making diagram of the task dispatcher.

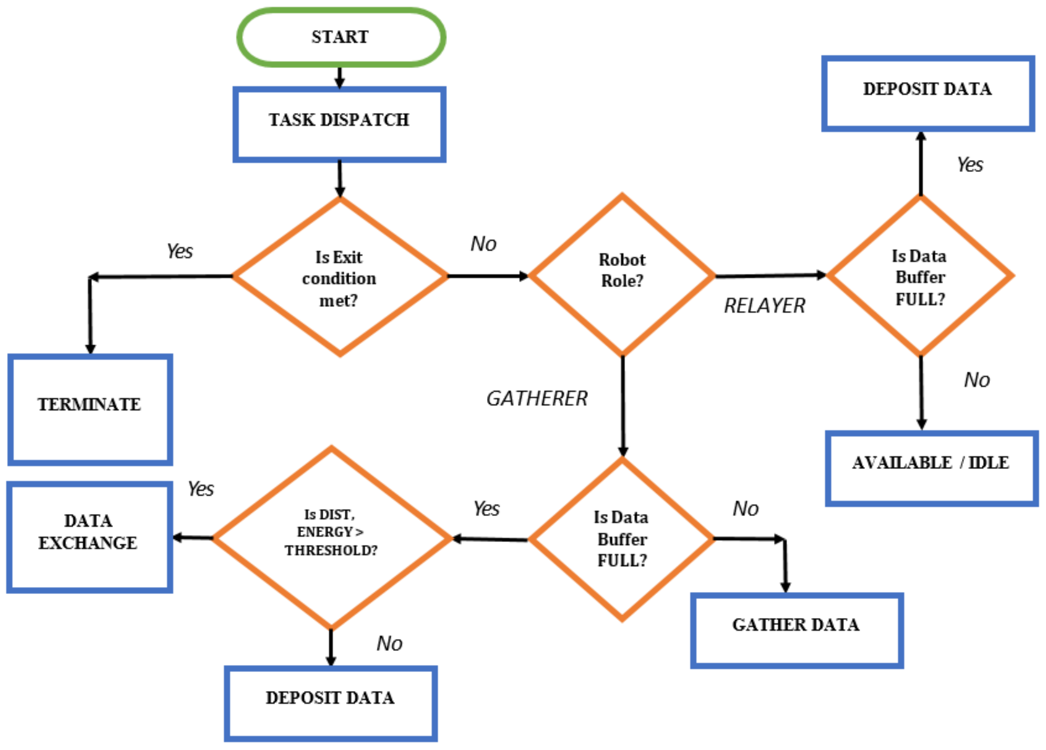


Figure 3.7: Flow Chart of Task Dispatch

- Data Gather

  Once in the data gather state, each robot is essentially a data gatherer. It travels to the designated location as per the location given by the nearest neighbor algorithm and ensures that any obstacle in the path is avoided using the obstacle avoidance algorithm. Once the location is reached, the robot uses the omni-directional robot to detect any colored target blobs in the location. The search algorithm detects the blob within a given radius. After data gather, the robot goes back to task dispatch for the next task as shown in Fig. 3.8.

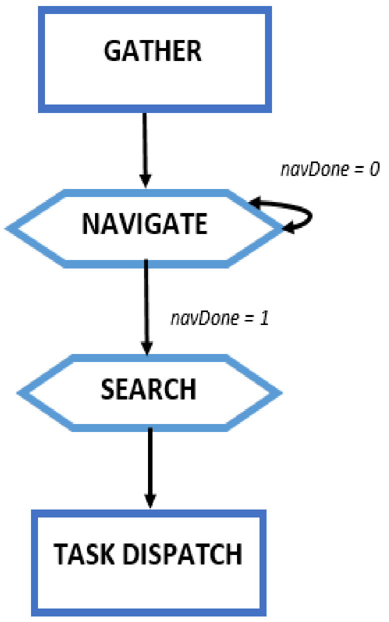


Figure 3.8: Flow Chart for Data Gather

- Deposit Data

  Both the data gatherers and data relayer robots enter the data deposition state in order to deposit data at the sink. Once the robot enters the data deposition state, it chooses the nearest sink location and navigates to it. Once it reaches the sink location, the robot uploads the data to a server shared by all the robots in the swarm. The robot enters task dispatch state once it completes data deposition and clears its own memory as shown in Fig. 3.9.

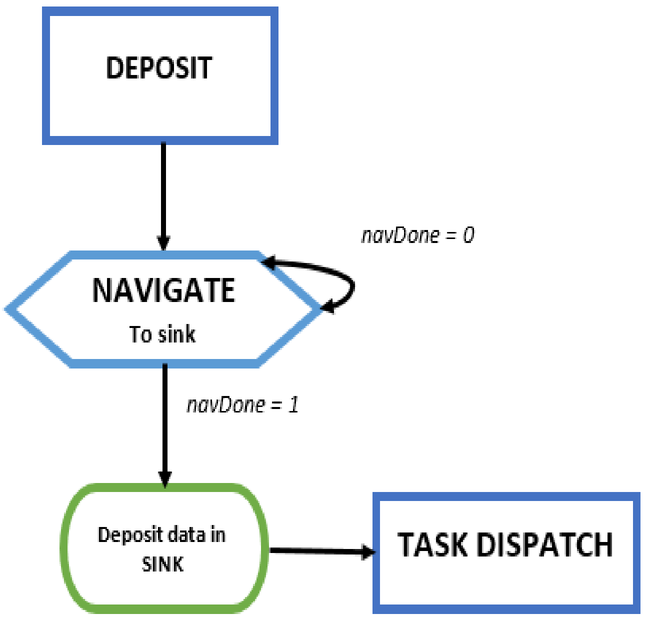


Figure 3.9: Flow Chart for Data Deposition

- Battery Recharge

  In the battery recharge state, the robots check their battery level and ensure that the next subsequent navigation does not result in zero energy. When the battery level is low, the robots travel to the nearest dock to get recharged. Since the dock and sink locations are close to each other, the robots check for data in the memory. If the memory is full, the robots proceed to sink for data deposition after recharge as shown in Fig. 3.10.

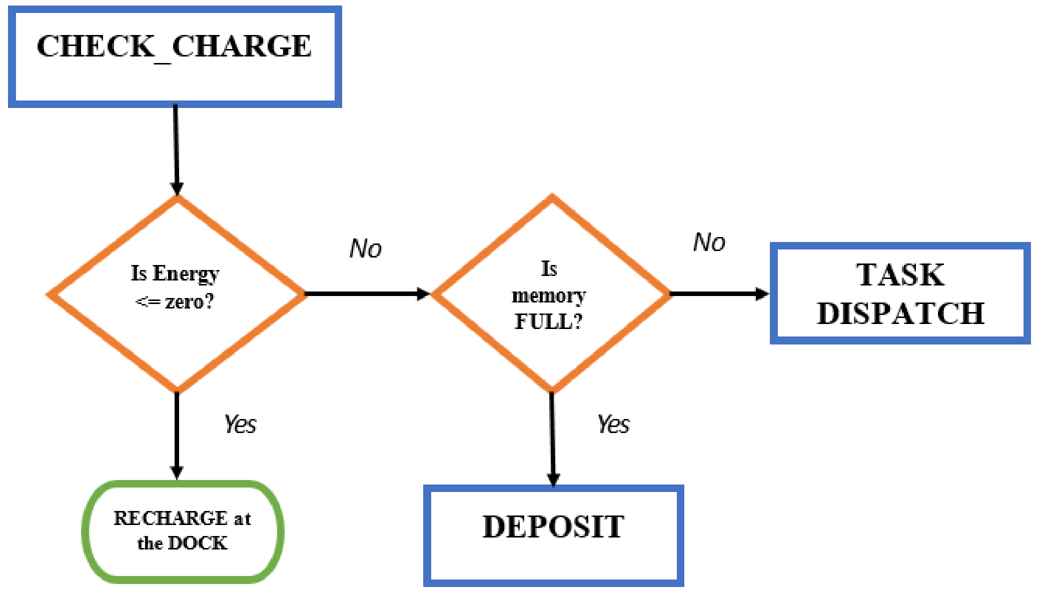


Figure 3.10: Flow Chart of Battery recharge

- Data exchange

  The data exchange is common to both data gatherers and data relayers. These robots perform different tasks in this state. The data gatherer enters data exchange state when it decides to relay the data with the help of a data relayer robot. The data gatherer transmits a relay request to notify the neighbor robots that it needs help for data transportation to the sink. The available robots responds with an available_response to the data gatherer. The data gatherer allots a recruitment slot for all available robots to apply for offering support. Once the slot is lapsed, the data gatherer chooses one of the available relayer robot based on either maximum energy criteria or nearest relayer criteria. The chosen relayer is then assigned the data exchange state. In this state, the relayer robot navigates to an assigned data exchange grid, to receive the data transferred by the data gatherer. Once the transfer is successful, the relayer robot proceeds with data deposition and the gatherer robot is assigned task dispatch in order to determine the next available task.Fig. 3.11 and Fig. 3.12 represents the state diagram for data exchange.

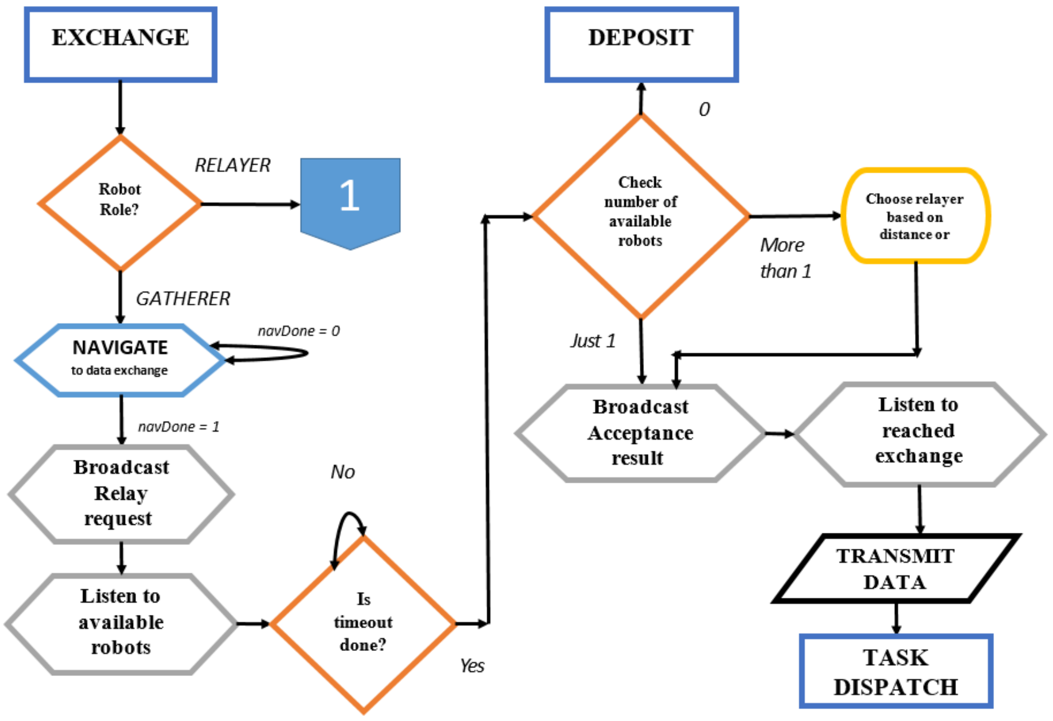

EXCHANGE
Robot Role?
RELAYER
1
GATHERER
NAVIGATE to data exchange
navDone = 0
navDone = 1
Broadcast Relay request
Listen to available robots
No
Is timeout done?
Yes
DEPOSIT
0
Check number of available robots
More than 1
Choose relayer based on distance or
Just 1
Broadcast Acceptance result
Listen to reached exchange
TRANSMIT DATA
TASK DISPATCH

Figure 3.11: Flow Chart for Data Exchange (Gatherer)

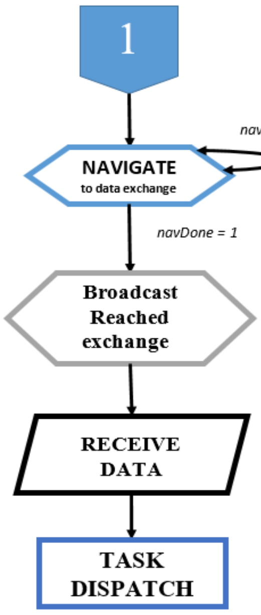


Figure 3.12: Flow Chart for Data Deposition (Relayer)

- Available/Idle

  Upon receiving the relay_request from the data gatherers, the robots after completing their current task, enters available/idle state. In this state, each robot broadcasts an available_response to the data gatherer to be enrolled for selection of data relayers. The robots remain in this state until the data gatherer selects one of the robots for data exchange. The selected robot enter data exchange to aid the requested data gatherer to relay data. The other robots then join data gather provided there are no other pending relay requests. This state signifies the number of robots actually ready to switch to data transportation task. Fig. 3.13 portrays the state diagram of each robot in available/idle state. Since the robots are not executing either data gathering or data transportation when in this state, this state is also termed as idle.

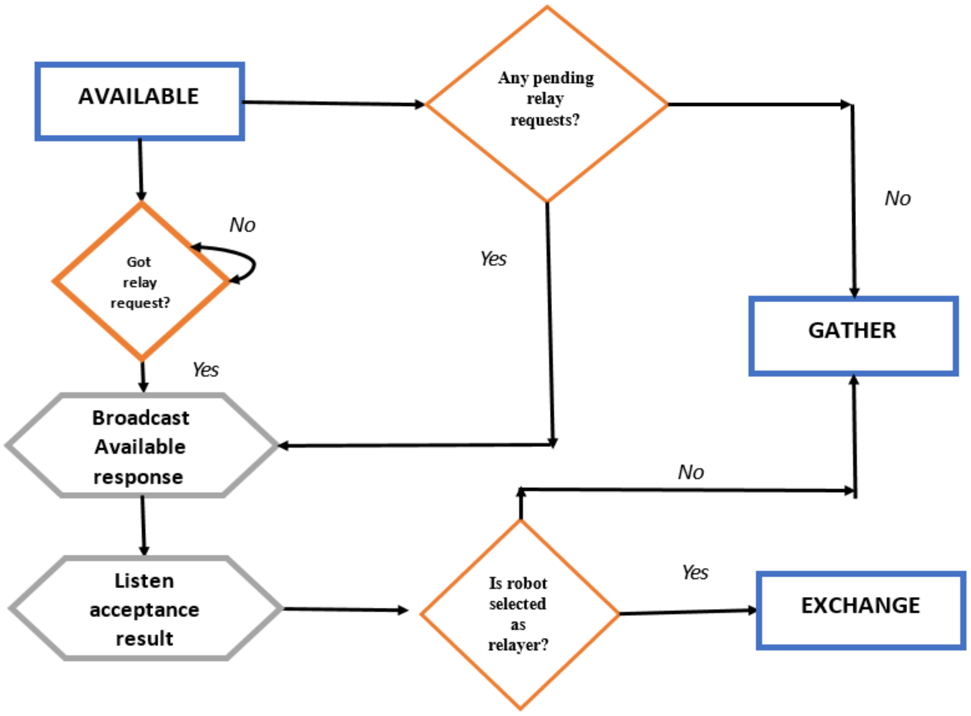

AVAILABLE
Any pending relay requests?
No
Got relay request?
No
Yes
GATHER
Yes
Broadcast Available response
No
Listen acceptance result
Is robot selected as relayer?
Yes
EXCHANGE

Figure 3.13: Flow Chart for Available/Idle

- Terminate

  This state marks the end of one complete experiment cycle. The robots enter the Terminate state once they have completed exploring all the locations of the arena collectively. The robots reach back to sink informing that the tasks are completed as shown in Fig. 3.14 .

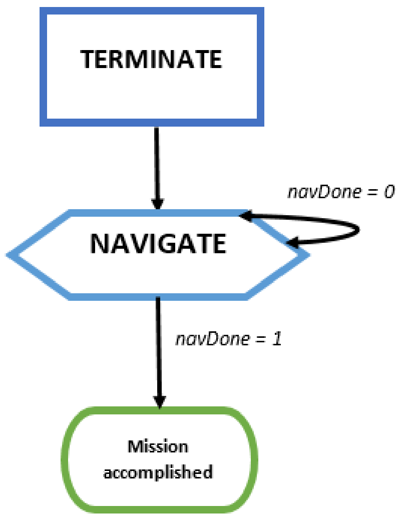


Figure 3.14: Flow Chart for Terminate

3.5 Summary

In this chapter, the design and implementation of the dynamic task allocation algorithm is discussed in detail.

- **Requirements and Problem Formulation**: We introduce the problem of interest and explain the limitations of the existing methods. We then list down the requirements of the dynamic task allocation algorithm which we propose.
- **Tasks and Roles**: We discuss the basis of our algorithm, the fundamental tasks the robots need to implement and the roles they play while implementing the tasks.
- **State Machines**: The tasks are now subdivided into different states and the robots now implement the tasks through different states. We explain the algorithms of each state in this section.

- **Dynamic Task Allocation Scheme** : We now explain in detail how the entire algorithm works. We present the mathematical model of battery and energy constraints and explain the criteria for task switching based on these constraints. We then elaborate the state diagrams of the algorithm and how the task are switched during runtime.

Chapter 4

Experiments and Results

Chapter 3 discussed the framework and design of dynamic task allocation algorithm. This chapter discusses the criteria for evaluation of the performance of the algorithm Sec. 4.1. Sec. 4.2 lists the experiment setup that is necessary to observe the behavior of the system. Sec. 4.3 shows the compiled results from selected sets of experiments. Since a vast number of simulations were carried out, it is not possible to show results from every set of simulations. Therefore, results that aid in understanding the nature of the system under various parameters are taken into consideration.

4.1 Criteria for Evaluation

In this section we specify the evaluation criteria and present the parameters that measure the performance of the algorithm.

4.1.1 Robot Specific Metrics

Distance Traveled

Reducing total distance traveled by the entire swarm is the primary criterion in robot metrics. In a physical world, reducing the total distance means using less energy to travel. This is critical as inexpensive robots usually have low battery life. Hence in this thesis, since we are imposing energy constraints on the robots, it is important to analyze the net distance traveled by each robot across all experiments.

4.1.2 Task Specific Metrics

Time Spent on each task

The time spent on each task is useful to understand how the swarm behaves under the contradicting goals of maximizing data gathering and minimizing energy loss. The time spent on data gathering and data transportation tasks will be of specific interest. The time spent by each robot across all experiments will be studied by varying the number of robots in the swarm, the timeout of the swarm as well as the energy constraint of the swarm.

Data Throughput of each task

Data throughput is the net amount of data arriving at the sink. In this algorithm, a robot can deposit data in the sink either as a data gatherer or as a data relayer. It will be interesting to study the data throughput of robot in each of these roles. It also measures directly the contribution of each task to prevent data loss.

4.1.3 Energy Specific Metrics

Energy Recharges per robot

The concept of energy recharge is defined as the number of visits to the dock. The energy recharge cycles indicate the cases in the experiment when the robot reached almost zero charge. By studying the energy recharge cycles across different parameters, we can better analyze the performance of the algorithm.

4.2 Experiment Design

Sec. 4.1 focuses on an evaluation metric to judge task allocation strategy. It is necessary to carry out extensive simulation to understand the behavior of this system with respect to different parameters. These parameters include strategy for relayer choice, the energy threshold for each robot, the timeout values for gatherer recruitment slots and number of robots in the arena. Additionally, varying these values for a large number of random seeds aids in understanding the system. Studying the behavior of swarm system requires simulations involving large number of robots. Sec. 4.2.4 describes the use of ARGoS simulator and the foot-bots used for simulations. The algorithm is implemented using the Buzz language [Pinciroli and Beltrame, 2016].

4.2.1 Fixed Parameters

Arena Size

To confirm the scalability of the system, it is important to test the system by increasing the number of robots. The arena size is kept constant in this case and the behavior of the system is studied under the situation that can possibly lead to crowding of robots. The arena size is fixed as 15 m$times$15 m.

Amount of data to be collected

Colored blobs are randomly distributed in the arena and these act as data of interest. The number of colored blobs in each experiment is fixed at 50.

Sink, Dock and Exchange locations

The location of the sink, dock and the exchange grids are fixed through all experiments. The sink and dock locations are located at one end of the arena and the exchange grid is located in the middle of the arena. The exchange grids divide the arena into two equal halves.Fig. 4.1 depicts the entire arena setup used for the experiments.

Start and End configuration of the robots

All the robots start at fixed locations in the sink and dock side of the arena. Once the task is completed the robots report back at the sink to check whether all the data has been uploaded to the server.

4.2.2 Variable Parameters

Distribution of data points

The data for each experiment is distributed randomly throughout the arena. The number of data points are fixed but their location is varied across each experiment.

Maximum Energy per robot

The maximum energy at the start of the experiment is varied for each set of experiments. The energy is varied across three values (Emin, 2Emin, Emax). The value

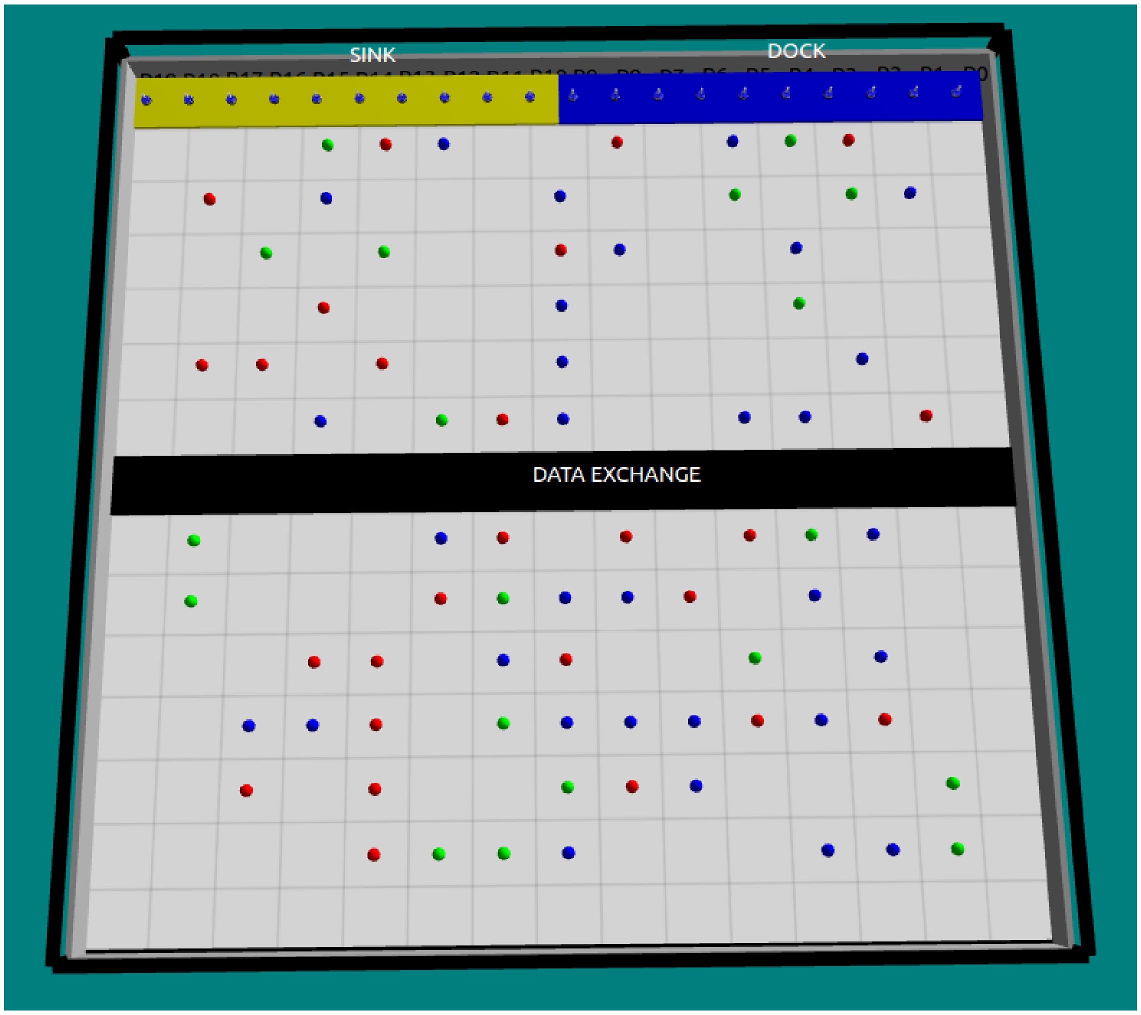


Figure 4.1: Arena of the experiment

of Emin is calculated as per the equation:

$$\text{Emin} = 2 \cdot \text{arena diagonal length} \cdot \text{energy per grid}$$

Timeout for relayer recruitment

The time for which data gatherers accept available requests from potential data relayers is the timeout parameter. During the timeout, the data gatherers recruit as many potential data relayers but once the timeout has lapsed only one relayer is selected based on the strategy for relayer choice. The potential relayer robots on the other hand, remain in the idle state till there are no pending relay requests. Table 4.17 shows the range of values of the timeout parameter.

Number of robots

To test the scalability and robustness of the algorithm for large number of robots, the number of robots, N is varied across the experiment. The number of robots are chosen such that they do not completely overcrowd the arena but could potentially overcrowd. Table 4.17 shows the values of N used in the experiment.

Strategy for choosing relayers

As in Sec. 3.4.2, the data gatherers choose data relayers either based two strategies.

- Strategy 0: Based on the relayer with maximum energy.
- Strategy 1: Based on the closest relayer in terms of distance.

Table 4.1: Parameter Table for Experiment

Parameters	Values
$Energy$	(Emin , 2Emin , Emax)
$Timeout$	(Tmin , 2Tmin , Tmax)
N	(10 , 14 , 20)
Strategy for relayer choice	0 , 1

4.2.3 Numerical Parameters

Random Seed

Since the algorithm depends on random number selection for data distribution in the arena, each experiment setting is carried out for 10 different values of random seed. This is important because data points (blobs) are distributed uniformly and the colors of the data points are also randomized.

4.2.4 ARGoS

ARGoS [Pinciroli et al., 2011] is a multi-robot simulator and was built for the Swarmanoid Project [Dorigo et al., 2013] at IRIDIA Lab at the Université Libre de Bruxelles, Belgium. ARGoS is highly flexible and efficient as it offers modularity in terms of design, parallelism in execution, and composability of objects. ARGoS is the most efficient tool to simulate the physics of thousands or even tens of thousands of robots.

Simulator

ARGoS facilitates use of multiple physics engines. ARGoS can divide the arena in regions and use a different dedicated physics engine for each region. This improves the flexibility and efficiency of the simulator and also allows robots to switch physics engines based on the position in the simulation space. ARGoS supports 2D kinematics and dynamics engines, a 3D particle engine and a 3D-dynamics engine.
The ARGoS simulator is highly modular. It facilitates the addition of different robots, physics engines,entities, visualizations, and communication media. This modularity makes ARGoS highly flexible. ARGoS simulator supports visualizations using a combination of Qt5 and OpenGL, ray tracing using POV-Ray, and text-

based visualization.

ARGoS is a multi-threaded simulator that maximises performance and improves speed of execution on modern CPUs. ARGoS executes an experiment in multiple simulation steps.

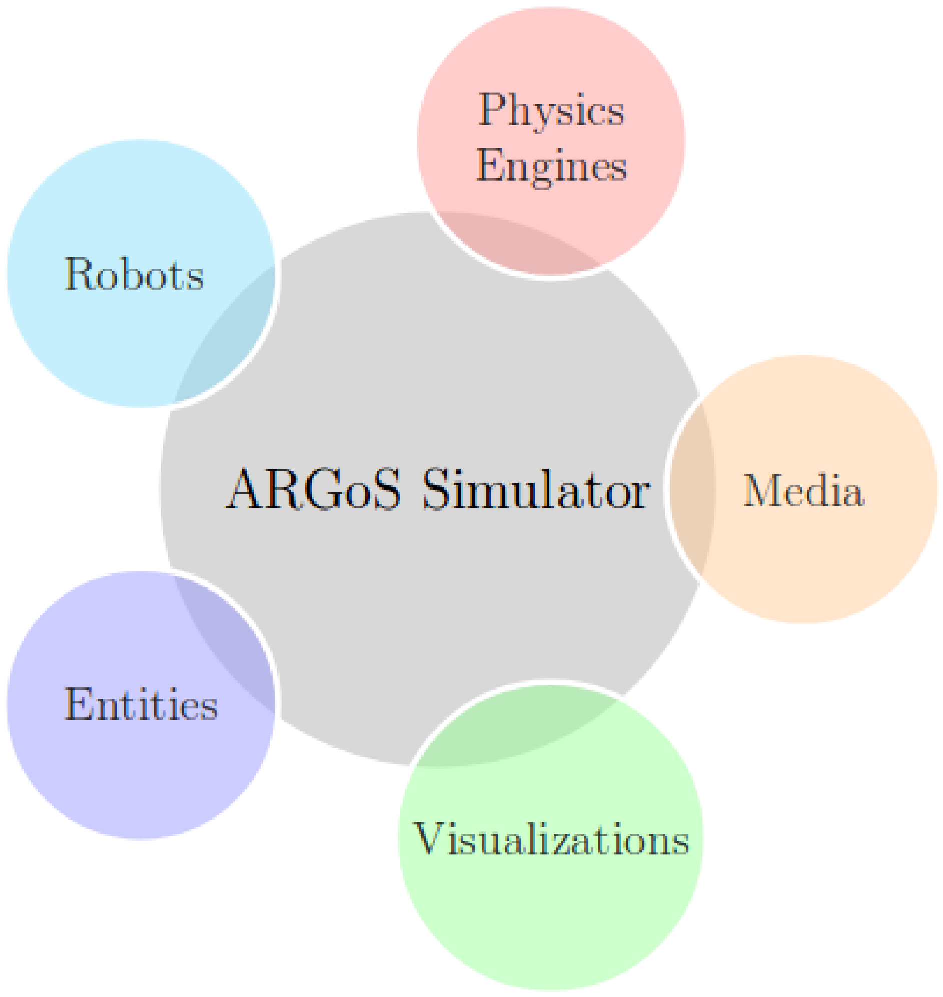


Figure 4.2: ARGoS Simulator

Footbot

The foot-bot robot was built at École Polytechnique Fédérale de Lausanne, Switzerland, for the Swarmanoid Project [Dorigo et al., 2013]. The foot-bot is a differential drive robot (maximum speed 30 cm/s) with a circular chassis of diameter 13cm and 28cm high. The foot-bot has mechanical modularity and can attach itself to other foot-bots or hand-bots with the aid of a docking ring and gripper. The foot-bot contains 24 IR sensors along the circular chassis which act as proximity sensors and additional 8 IR sensors to the base of the foot-bot which perceive reflected shades of grey. The foot-bot is also equipped with 13 LEDs, 12 on the circular ring and one on top. The foot-bot is also equipped with two cameras; one top-front camera and an omni-directional camera. It contains two distance sensors; one for near range (40-300 mm) and a long range sensor (200 - 1500 mm). Communication between foot-bots is achieved using the Range and Bearing system.

The foot-bot is an ideal robot for swarm applications due to the amount of sensing capability, mechanical modularity, interaction with other robots and environment, small size, and ability to hot-swap the battery. Thus the foot-bot can be used to mimic recharging robots, self-assembling robots, autonomous and decentralized swarm experiments. In this work proximity, range and bearing, positioning, LED, and omni-directional camera are used.

Controller

A controller is a plug-in that controls the behaviour of robots. ARGoS aids in developing user code, which is directly usable on real robots by making the controller access a Control Interface. Control Interface enables users to access sensors and actuators in a manner similar to that of real robots. Controllers for robots are written in C++, Lua, and Buzz [Pinciroli and Beltrame, 2016]. Inside the ARGoS config-

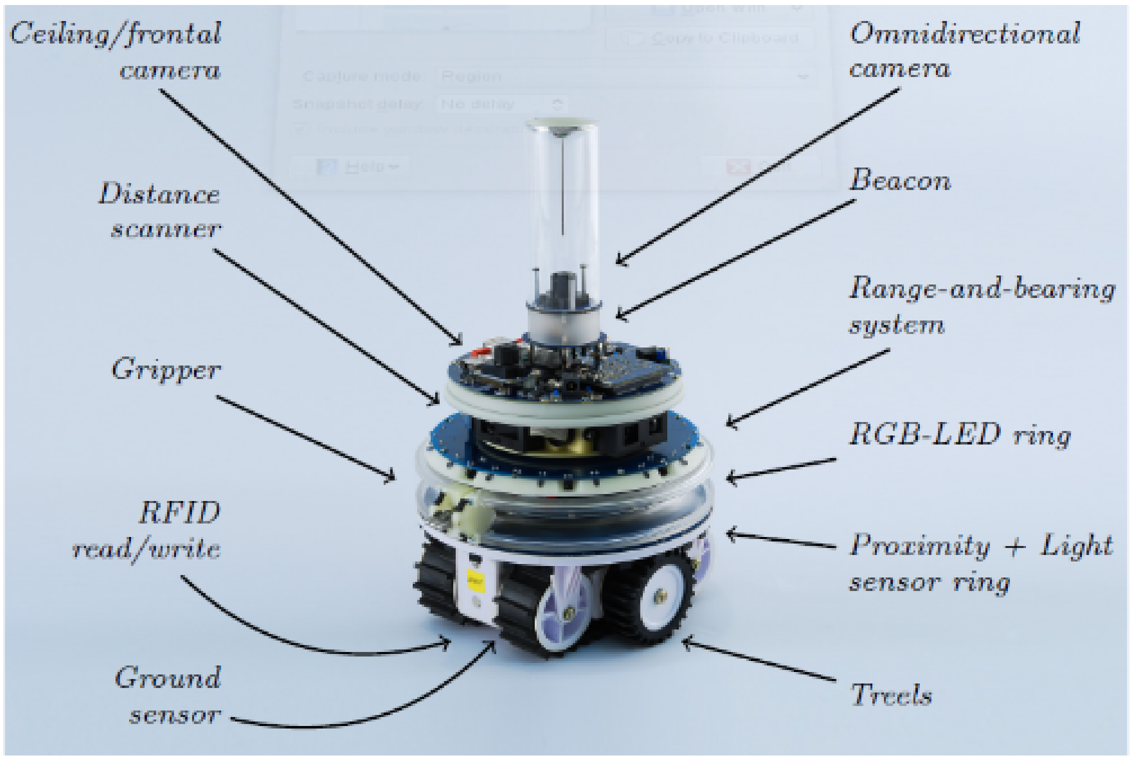


Figure 4.3: Foot-bot(Source: IRIDIA)

uration file (.argos), the controller section allows addition of multiple controllers to an experiment.

Loop Functions

Loop functions allow users to modify the simulation. Loop functions act as hooks in an experiment and aid in initialization as well as determining the end of an experiment. Loop functions enable users to capture robot and environment data after every time-step and offers hooks for analysis after the experiment. Loop functions also provide functions for adding, removing, altering parameters or modifying states of entities. Entities in this case are robots, and objects in the arena. In this work, loop functions are used to initialize the experiment by placing target blobs in different locations in the arena. Once the experiment begins, the Shared Table of grid locations of the arena is communicated to all robots at the start of each control loop. The shared table is updated by counting the robots that visited a particular location. This provides the robots with updates of the locations which are already visited and hence the robots makes sure that a location is not visited twice. Loop functions are also used to monitor and record task allocation data after every 100 time-steps (10 secs) and distance traveled by the robots. Data analyzed from various experiments in Sec. 4.3.

4.3 Results

This section shows results for various set of experiments described in Sec. 4.2. We present the experiment results for each criteria. Under each criteria, we compare the effects of different parameters of robot task distribution. Each experiment is run for 10 different random seed which varies the distribution of the data blobs in the arena.The data plots are represented mainly as box plots which highlight the mean behavior of the system in addition to outliers. In each section, we highlight the inference drawn from the graphs and how they help to understand the emergent behavior of the system.

4.3.1 Robot Specific Metrics

Distance Traveled by Each Robot

The total distance covered by each robot while performing area exploration and data gathering is studied in this section. The data plots are represented in the form of box plots to show the mean, median and variance of the distance covered by each robot across 10 experiments.

- *Varying Timeout Parameter*

The timeout parameter specifies the time each data gatherer waits for receiving available requests from data relayers. This time window allows more robots to volunteer as data relayers and hence gives a better choice for data gatherers while choosing a suitable data relayer for data transportation to the sink. Fig. 4.4 shows the box plots of the distance covered by each robot for different values of timeout.

Parameter Table

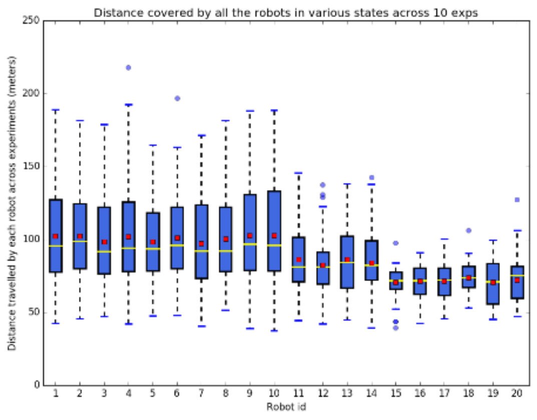
Table 4.2: Parameter Table for Distance Traveled, Varying Timeout

Parameters	Values
$Energy$	Emin
N	20
Strategy for relayer choice	Nearest Relayer (1)

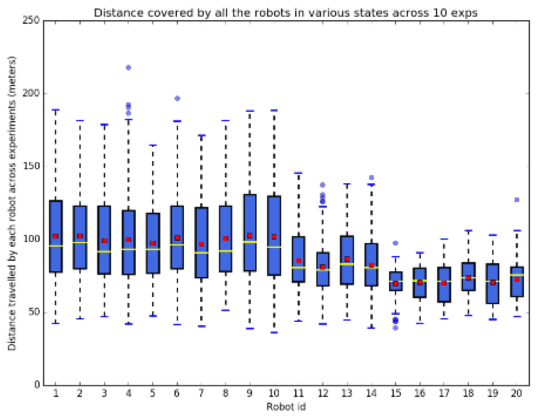
Inference

The impact of the timeout parameter on the distance covered by the robots is low. We can conclude from the plots that the distance traveled by the robots across varying parameters is more or less same. This is because the timeout does not directly impact the distance covered by the robots.

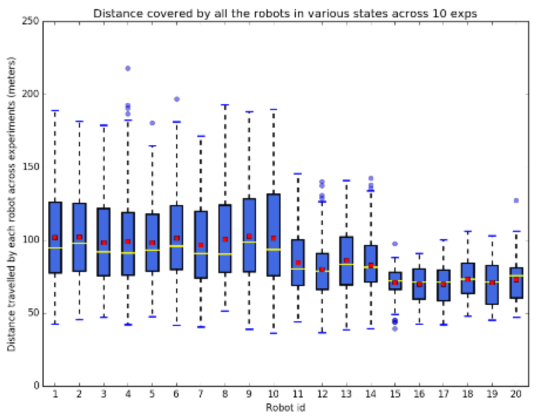
Verdict: Timeout to choose a relayer does not impact the distance covered by each robot.



(a) Timeout = Tmin



(b) Timeout = 2Tmin



(c) Timeout >> 2Tmin

Figure 4.4: Impact of variation in timeout parameter on the distance traveled

- *Varying Maximum Energy per robot*

  The maximum energy is the amount of energy available to each robot at the start of the experiment. By varying the maximum energy, we vary the energy constraint of each robot. Fig. 4.5 shows the variation in the distance covered by each robot for different values of maximum energy.

  Parameter Table

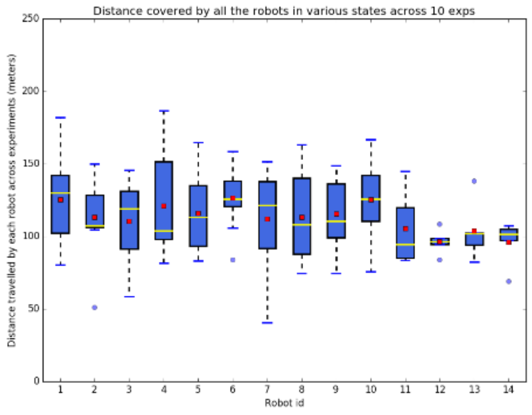
Table 4.3: Parameter Table for Distance Traveled, Varying Energy

Parameters	Values
$Timeout$	Tmin
N	14
Strategy for relayer choice	Max Energy Relayer (0)

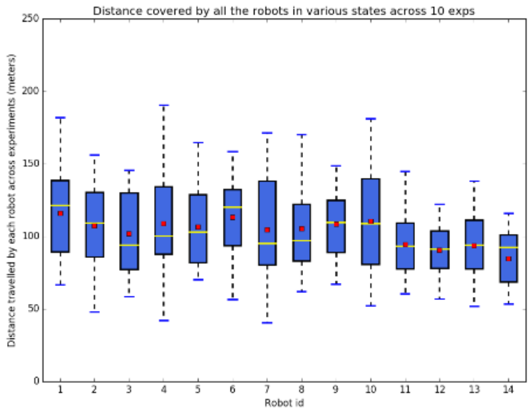
Inference

There is a marginal decrease in the distance covered by each robot for maximum energy case as the robot do not additionally travel for recharge. We can observe this decreasing trend in the graphs by monitoring median values of the graph.

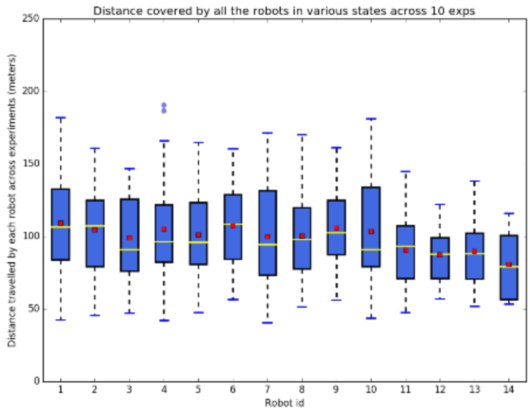
Verdict: The increase in energy of the robots marginally decreases the distance covered as the robots need not travel additionally to the dock to recharge.



(a) Energy = Emin



(b) Energy = 2Emin



(c) Energy >> 2Emin

Figure 4.5: Impact of variation in energy on distance traveled

- *Varying Number of Robots*

  We analyze how well the algorithm scales with the robot density in terms of distance covered by each robot in the swarm. Fig. 4.6 portrays the plots corresponding to 10, 14 and 20 robots respectively.

Parameter Table

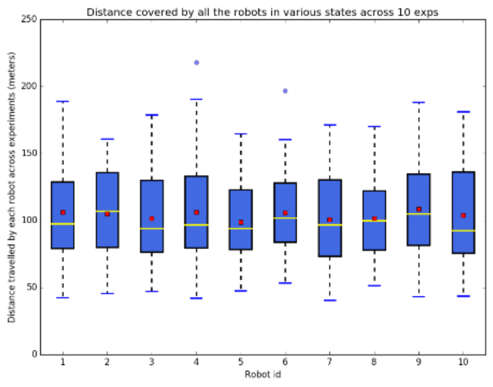
Table 4.4: Parameter Table for Distance Traveled, Varying Number of Robots

Parameters	Values
Energy	2Emin
Timeout	2Tmin
Strategy for relayer choice	Max Energy Relayer (0)

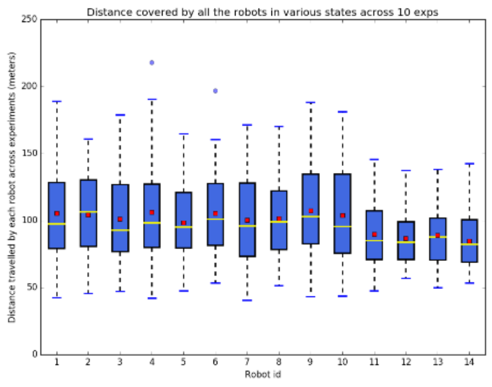
Inference

The plots convey decrease in the distances covered by each newly added robot to the swarm. When the number of robots are 10, the distance covered by the robots are almost the same. On adding more robots, the newly added robots tend to cover relatively less distance. This indicates an observation that the newly added robots are prominently data gatherers hence they do not travel to sink frequently for data deposition and hence dispense less energy. Hence they do not travel to the dock for recharging.

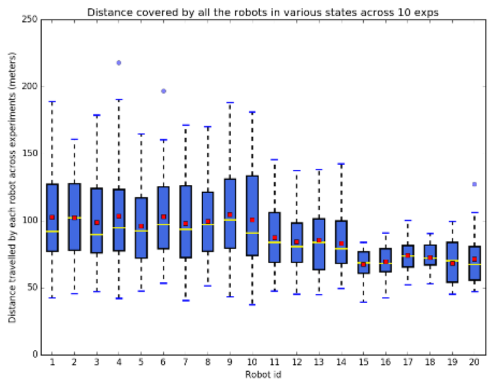
Verdict: Newly added robots potentially behave as data gatherers and hence they cover less distance. The decreasing trend of the distance covered by newly added robots is a significant observation about the task allocated to newly added robots to the swarm.



(a) N = 10



(b) N = 14



(c) N = 20

Figure 4.6: Impact on the distance traveled when number of robots is varied

- *Varying the strategy of choosing relayers*

  The data gatherers can choose relayers using two strategies. The relayer could be the robot with the maximum energy or the nearest robot to data exchange. Fig. 4.7 shows the variation in the distance covered by the robots for both the strategies.

Parameter Table

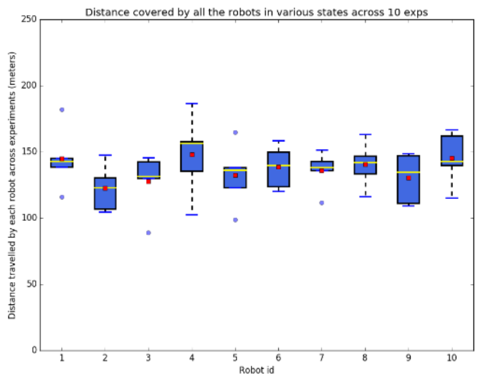
Table 4.5: Parameter Table for Distance Traveled, Varying Strategy for relayer choice

Parameters	Values
$Energy$	Emin
$Timeout$	Tmin
N	10

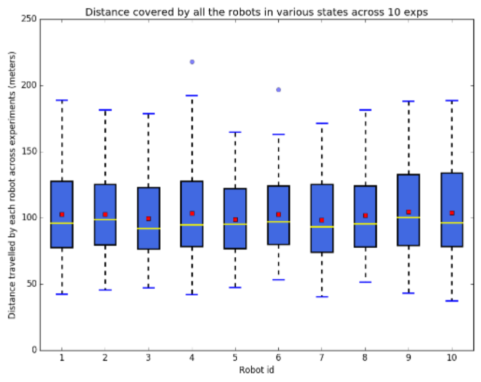
Inference

It is seen that the distance covered by the robots is relatively higher for Max Energy Relayer strategy. Since the closest relayer is chosen in the Nearest relayer strategy we expect that the robots travel less distance for data transportation. The plots justify this reasoning. The distance covered by each robot in Strategy 1 is more uniform than Strategy 0.

Verdict: Strategy 1 (Nearest Relayer) performs better in terms of distance covered by each robot.



(a) Strategy = Max Energy Relayer



(b) Strategy = Nearest Relayer

Figure 4.7: Impact of variation in the strategy on the distance traveled

4.3.2 Task Specific Metrics

Time Spent on each task

The time spent by the robot in various states of the task are studied in this experiment. The main states considered as parameters are as follows:

- ⋆ Gatherer Time: Fraction of time spent in gathering data from the arena.
- ⋆ Relayer Time: Fraction of time spent as data relayer.
- ⋆ Exchange Time: Fraction of time data gatherer spent at the data exchange for data transfer to relayer.

⋆ Idle Time: Time the robots volunteer as relayers and wait for acceptance from gatherers for data transportation task.

⋆ Recharge Time: Time spent to navigate to dock and get recharged.

⋆ Deposition Time: Time spent to navigate to sink and deposit data.

⋆ Zone Time: Time spent on either side of data exchange zone. This helps to study the robot distribution in the environment.

- *Varying Timeout Parameter*

  In this experiment, we analyze the impact of varying the timeout parameter on the time spent on each state of the robot swarm. Fig. 4.8 shows the variation of timeout parameter and the variation in the time of each state in the form of box plots.

  Parameter Table

  Table 4.6: Parameter Table for Time spent on each task, Varying Timeout

Parameters	Values
Energy	Emin
N	10
Strategy for relayer choice	Max Energy Relayer (0)

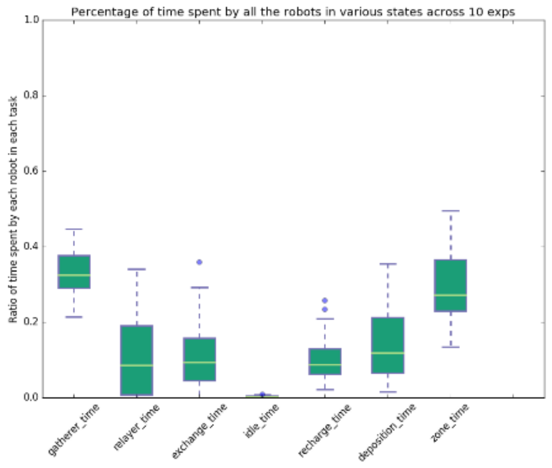
  Inference

  The key observations from the data plots are as follows:

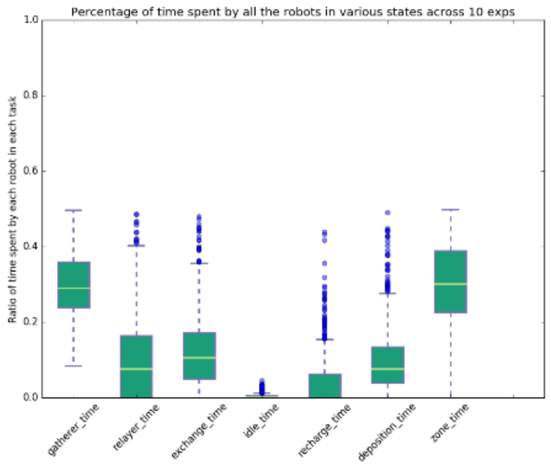
  – Idle Time (↑): The idle time of the swarm increases with increase in time. This is an expected outcome since timeout essentially means window for robots to volunteer themselves as relayers. The longer the timeout, the longer the data gatherer takes to choose the best relayer.

- Data Deposition Time (↓): The time taken to deposit data reduced. This is an interesting observation. As the timeout window increase, the gatherer are actually getting better relayer choices, hence the time to deposit data is decreasing.
- Recharge Time (↓): A better relayer choice also contributes in saving more of battery. The increase timeout value hence yields a better relayer for data transportation.

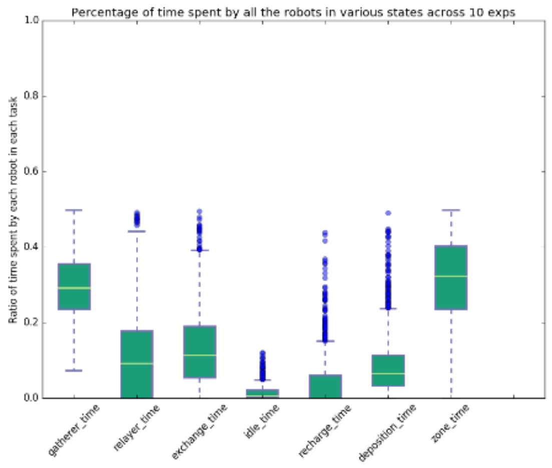
Verdict: Increase in timeout parameter increases the idle time but decreases the data deposition and recharge time. This is because more the timeout the gatherer chooses the best relayer amongst the choice of relayers. This allows for less data deposition and recharge time.



(a) Timeout = Tmin



(b) Timeout = 2Tmin



(c) Timeout >> 2Tmin

Figure 4.8: Impact of timeout variation on the time spent in each task

- *Varying Maximum Energy per robot*

  We study the impact of varying the energy constraint to the time spent by each robot of different states. Fig. 4.9 portrays the variation in time based on decreasing the energy constraint.

Parameter Table

Table 4.7: Parameter Table for Time Spent on each task, Varying Energy

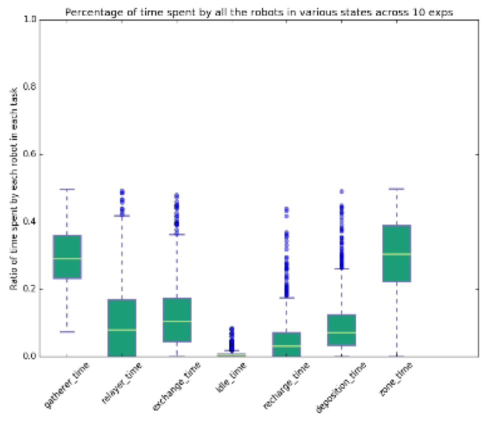
Parameters	Values
$Timeout$	2Tmin
N	20
Strategy for relayer choice	Max Energy Relayer (0)

Inference

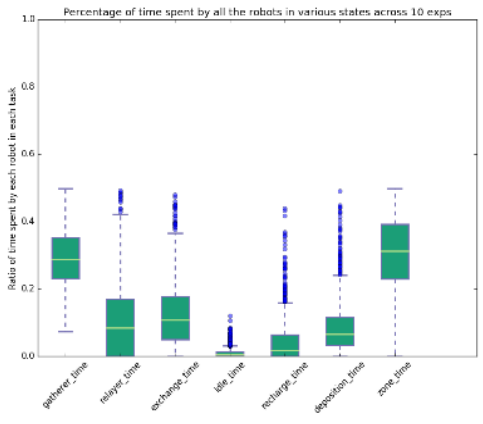
The key observation is as follows:

- Recharge time ($\downarrow$): The decrease in energy constraint gives more energy to each robot and this reduces the need for more recharges.

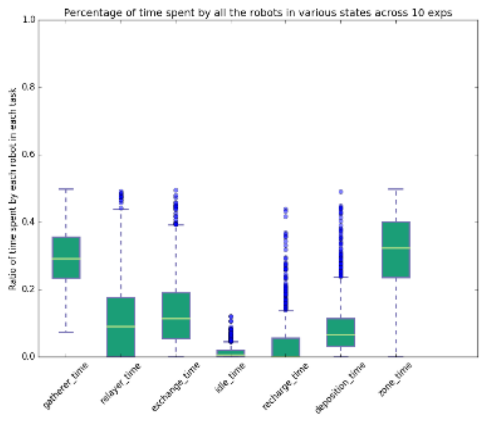
Verdict: The decrease in recharge time is an expected outcome since we are providing more energy to the robots.



(a) Energy = Emin



(b) Energy = 2Emin



(c) Energy >> 2Emin

Figure 4.9: Impact of energy variation on time spent in each task

- *Varying Number of Robots*

  We test the scalability of the algorithm and how it impacts the time spent in each state of the task as shown in Fig. 4.10.

Parameter Table

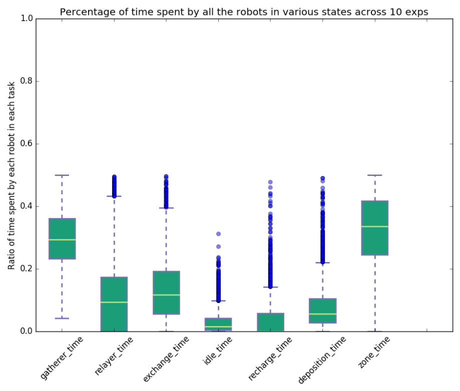
Table 4.8: Parameter Table for Time spent on each task, Varying number of robots

Parameters	Values
Energy	Emax
Timeout	Tmax
Strategy for relayer choice	Nearest Relayer (1)

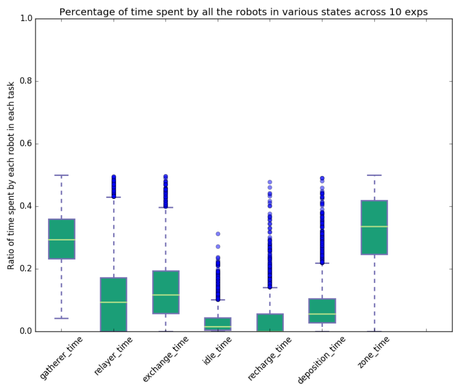
Inference

The effect of newly added robots to the swarm marginally decreases the deposition time. We can observe that the robots are allocated tasks such that they essentially complete each state in the same time.

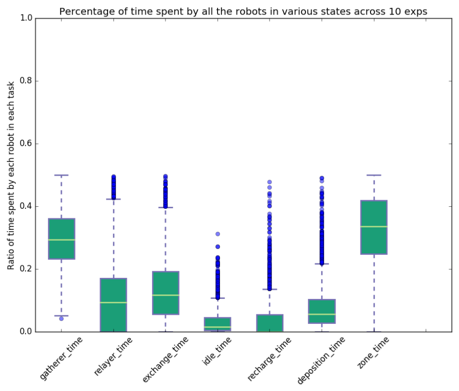
Verdict: We affirm the scalability of the algorithm with this result.



(a) N = 10



(b) N = 14



(c) N = 20

Figure 4.10: Impact on time spent on each task when the number of robots is varied

- *Varying the strategy of choosing relayers*

  The relayer choice is determined by either the robot with the maximum energy or the robot nearest to the data exchange grid. The impact of relayer choice on the time distribution for each state is studied in Fig. 4.11.

Parameter Table

Table 4.9: Parameter Table for Time Spent on each task, Varying strategy

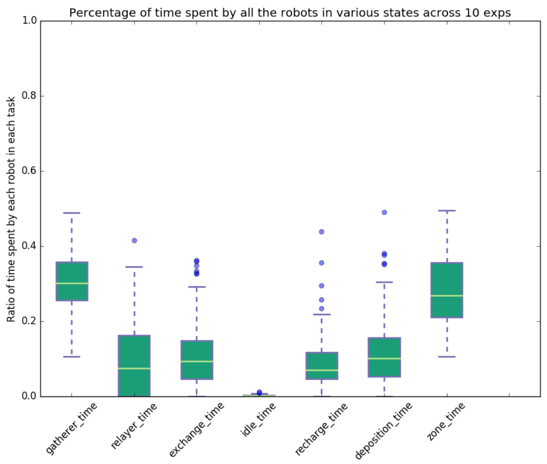
Parameters	Values
Energy	Emin
Timeout	Tmin
N	14

Inference

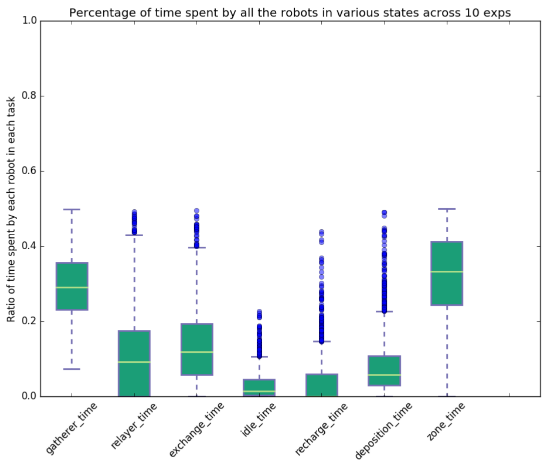
The key observation from the data plots are as follows:

- Idle Time(↑) for nearest relayer strategy
- Recharge Time(↓) for nearest relayer strategy
- Data Deposition Time(↓) for nearest relayer strategy

Verdict: Strategy 1 (nearest relayer) performs better in terms of reducing the recharge time and data deposition time of the swarm. This is due to choosing robots closer to the data exchange which yield more efficient distribution of robots.



(a) Strategy = Max Energy Relayer



(b) Strategy = Nearest Relayer

Figure 4.11: Impact of variation in strategy on time spent on each task

Data Throughput of each task

We study contribution of data gatherers and data relayers in the task of data deposition to the sink. The data throughput is defined as the net data deposited in the sink at given point of time. We compare the data deposition by gatherers and relayers to understand the distribution of work load in the system.

- *Varying Timeout Parameter*

  In Fig. 4.12, we observe the effect of varying timeout parameter on the data throughput of the system.

Parameter Table

Table 4.10: Parameter Table for Data Throughput of each task, Varying Timeout

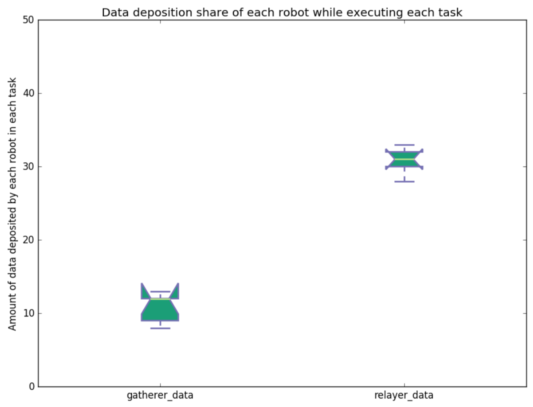
Parameters	Values
$Energy$	Emin
N	14
Strategy for relayer choice	Nearest Relayer (1)

Inference

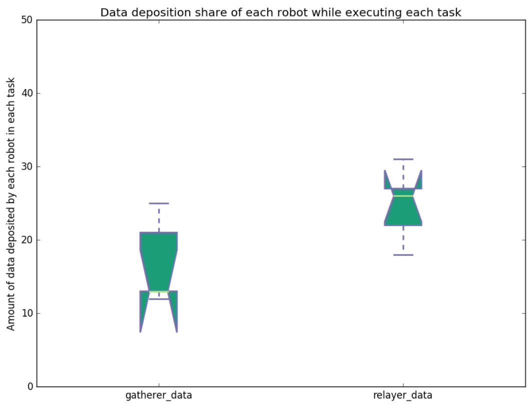
The key observation in this experiment are as follows:

- Lower Timeout: Relayers dominate the net data deposition.
- Higher Timeout: The data deposition tends to be equally divided among the robots.

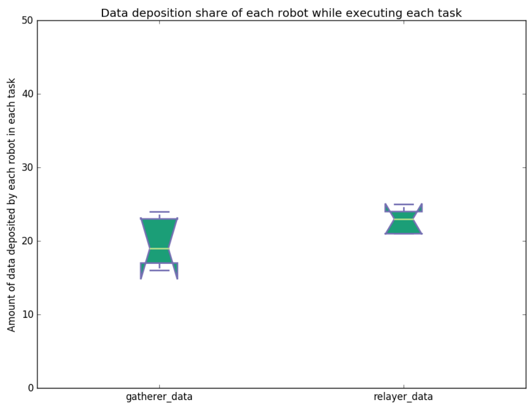
Verdict: More timeout window as opposed to yielding a better relayer choice, forced more gatherers to deposit data. It is a contradicting condition which can occur when more data gatherers are at the exchange and the best relayers get allocated to some set of these data gatherers whereas the other set of data gatherer are not having data relayers to help.



(a) Timeout = Tmin



(b) Timeout = 2Tmin



(c) Timeout >> 2Tmin

Figure 4.12: Impact of variation in timeout on data throughput

- *Varying Maximum Energy per robot*

  The maximum energy available to each robot is varied in this experiment. We study the impact on the data throughput by varying the energy of the robot.

Parameter Table

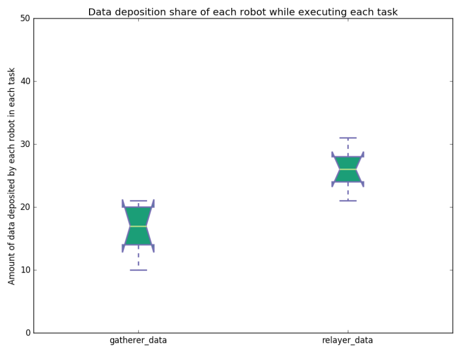
Table 4.11: Parameter Table for Data Throughput of each task, Varying Energy

Parameters	Values
Timeout	2Tmin
N	14
Strategy for relayer choice	Max Energy Relayer (0)

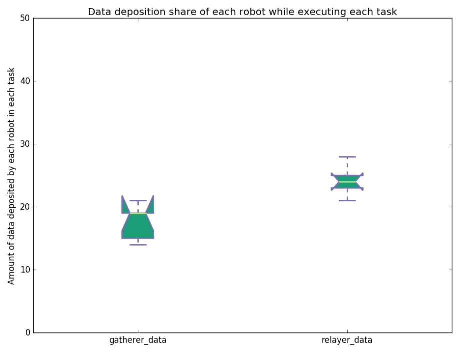
Inference

From the data plots, we can observe that as the energy increases the robots tend to split the data deposition task equally. So when there is a hard constraint on energy, the relayers are more active in depositing data than gatherers.

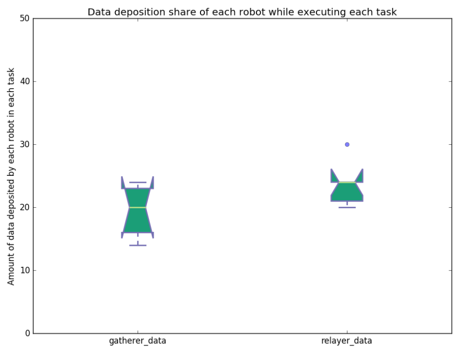
Verdict: The algorithm performs better when there is a hard constraint on the energy available to the robot. This facilitates better use of data relayers. The robots favor depositing data as data relayers than gatherers.



(a) Energy = Emin



(b) Energy = 2Emin



(c) Energy >> 2Emin

Figure 4.13: Impact of variation in energy on data throughput

- *Varying Number of Robots* The data deposition share between data relayers and data gatherer is estimated across different number of robots in the swarm.

Parameter Table

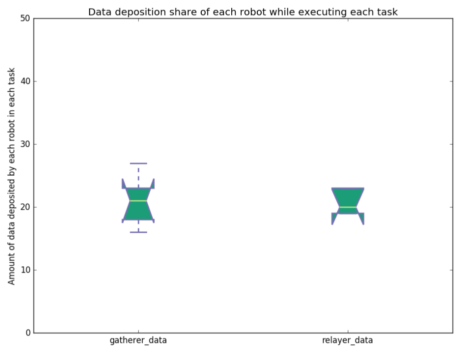
Table 4.12: Parameter Table for data throughput of each task, Varying Number of Robots

Parameters	Values
$Energy$	Emax
$Timeout$	Tmax
Strategy for relayer choice	Nearest Relayer (1)

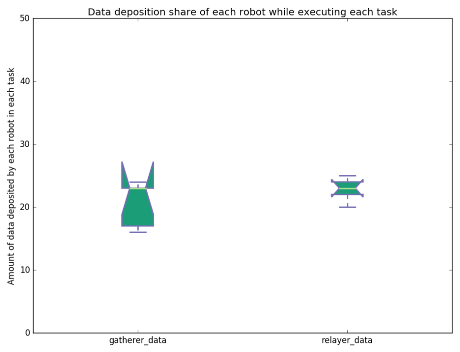
Inference

The variation across the number of robots is minimal and we can observe that the robots allocate the tasks between themselves to deposit data in the same manner when new robots are added to the swarm.

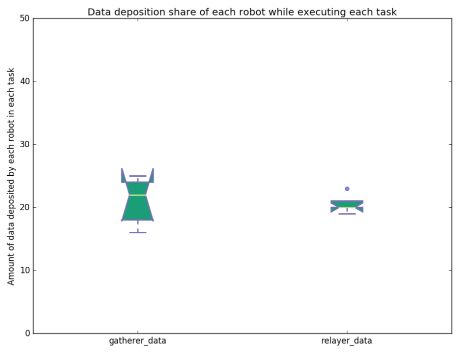
Verdict: Newly added robots does not impact the data deposition distribution by gatherers or relayers.



(a) N = 10



(b) N = 14



(c) N = 20

Figure 4.14: Impact of variation in number of robots on data throughput

- *Varying the strategy of choosing relayers*

  The strategy for choosing the relayer robot is varied and we study the impact of this variation on the data throughput of data gathering and data transportation tasks of the robots.

Parameter Table

Table 4.13: Parameter Table for data throughput of each task, Varying Strategy of relayer choice

Parameters	Values
Timeout	2Tmin
Energy	2Emin
N	10

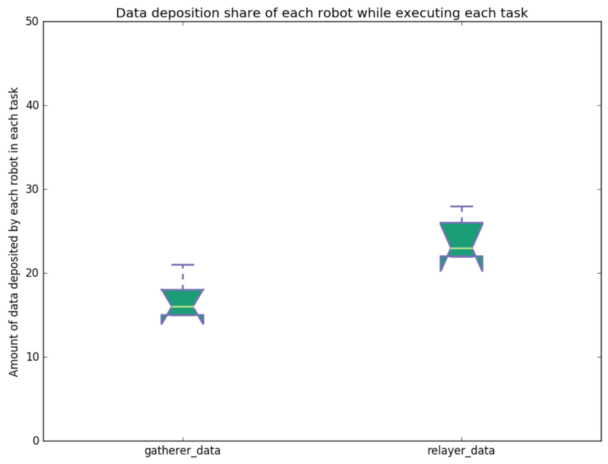
Inference

The key observations from the data plots are as follows:

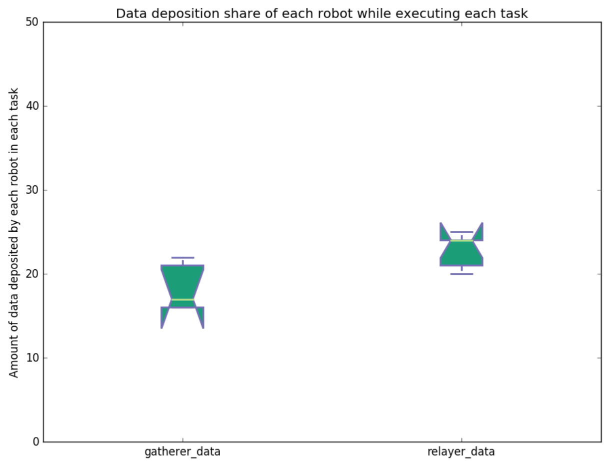
– Max Energy Relayer Strategy: The relayer robots deposit less data compared to data gatherers.

– Nearest Relayer Strategy: The relayers have deposited more data compared to the max energy strategy.

Verdict: The Strategy 1 is better in utilising the data relayers for data deposition than Strategy 0.



(a) Strategy = Max Energy Relayer



(b) Strategy = Nearest Relayer

Figure 4.15: Impact of variation of strategy on data throughput

4.3.3 Energy Specific Metrics

We study the energy dissipation in each robot by varying different parameters. The number of battery recharge cycles per robot is studied across all experiments.

Energy Recharges per robot

The energy recharges per robot is defined as the number of times the robot visits the dock to recharge its battery. We study this parameter to understand the behavior of the energy constraint of the swarm.

- *Varying Timeout Parameter*

  We study the impact of varying the timeout parameter on the energy dissipation rate in each robot.

  Parameter Table

  Table 4.14: Parameter Table for Energy Recharges per robot, Varying Timeout

Parameters	Values
Energy	2Emin
N	14
Strategy for relayer choice	Max Energy Relayer (0)

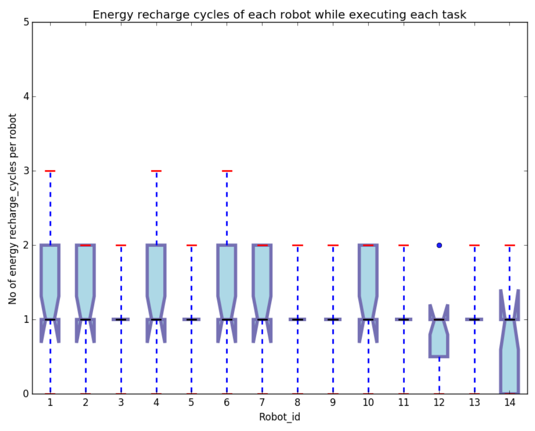
  Inference

  The key observations derived from the data plots are as follows:

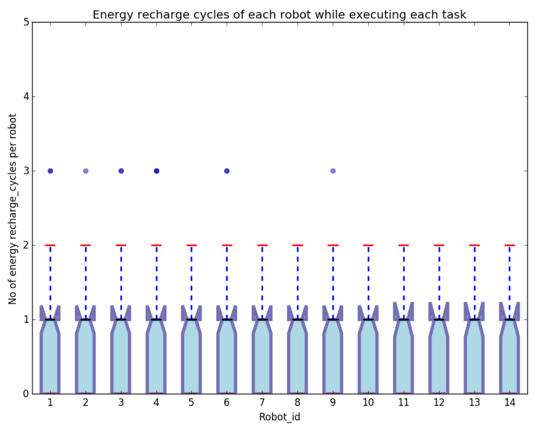
  – Lower Timeout: The number of recharges are relatively more. The choice of relayer may not be the best for lower timeout hence more robots end up recharging their battery.

  – Higher Timeout: The average recharge cycles is just 1 per robot which essentially means the task allocation is more effective in these cases.

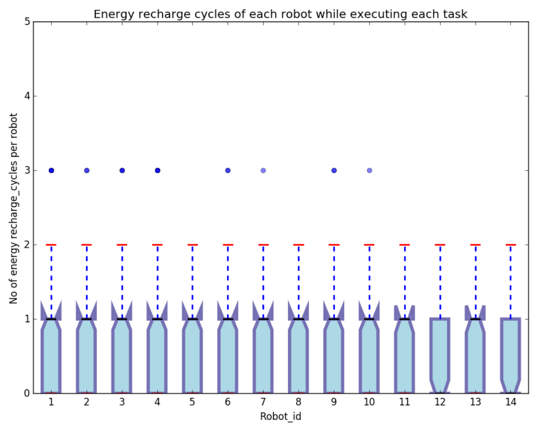
  Verdict: More timeout window, lesser the number of recharge cycles per robot.



(a) Timeout = Tmin



(b) Timeout = 2Tmin



(c) Timeout >> 2Tmin

Figure 4.16: Impact of variation of timeout on the energy recharge cycles

- *Varying Maximum Energy per robot*

  More the energy, the lesser the number of recharge cycles. We study the impact of hard and soft energy constraint on the battery recharge cycle of each robot.

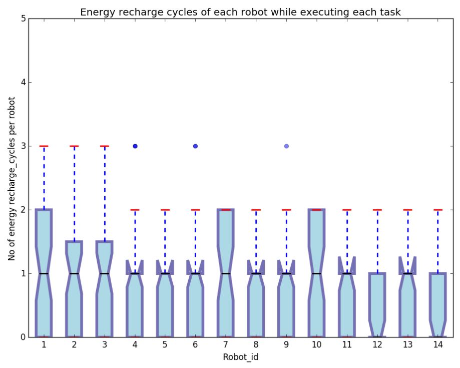
Parameter Table

Table 4.15: Parameter Table for energy recharges per robot, Varying Energy

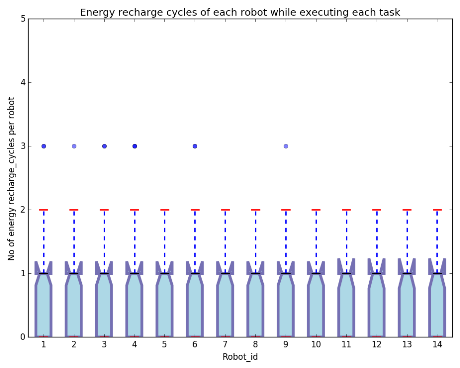
Parameters	Values
Timeout	2Tmin
N	14
Strategy for relayer choice	Max Energy Relayer (0)

Inference

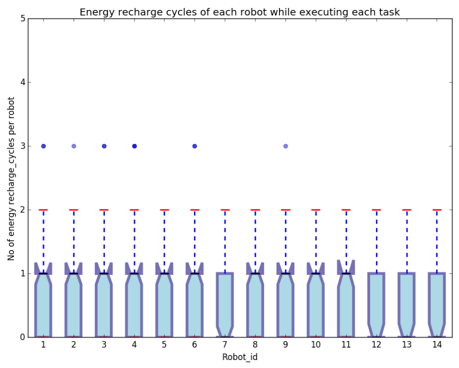
We observe a transition in the number of recharges from maximum of 2 to maximum of 1 per robot as energy is increased. For the maximum energy case, 4 out of 14 robots have never recharged once during the entire experiment. Verdict: The number of recharge cycles decreases as the maximum energy is increased.



(a) Energy = Emin



(b) Energy = 2Emin



(c) Energy >> 2Emin

Figure 4.17: Impact of variation of maximum energy on the recharge cycles per robot

- Varying Number of Robots

  We study the behavior of the system when more robots are added to the swarm. Fig. 4.18 shows the variation in the recharge cycles of each robot when number of robots is increased.

Parameter Table

Table 4.16: Parameter Table for energy recharges per robot, Varying Number of robots

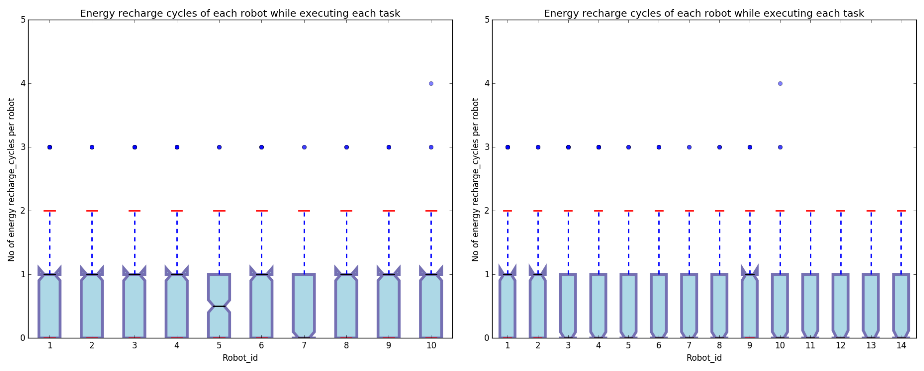
Parameters	Values
Energy	Emax
Timeout	Tmax
Strategy for relayer choice	Nearest Relayer (1)

Inference

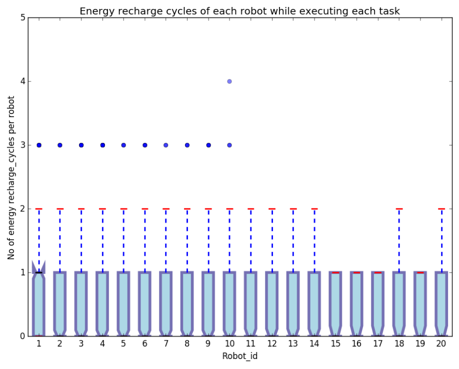
We observe that with increase in the number of robots, the number of recharge cycles per robot decreases.

- N = 10 : For the 10 robot case, we can observe that the median of atleast 1 recharge cycle occurs for each robot.
- N = 14: For the 14 robot case, we can observe that more robots have zero recharge cycles.
- N = 20: For the 20 robot case, the recharge cycles is zero for almost all robots.

Verdict: More the robots, energy is conserved better in this algorithm.



(a) N = 10 (b) N = 14



(c) N = 20

Figure 4.18: Impact of variation of number of robots on the recharge cycles per robot

- *Varying the strategy of choosing relayers*

  The strategy to choose a relayer determines whether a relayer with maximum energy or close proximity is chosen for data transportation. In Fig. 4.19, we study the effect of both these strategies on the number of energy recharges per robot.

Parameter Table

Table 4.17: Parameter Table for Energy recharges per robot, Varying Strategy

Parameters	Values
Timeout	Tmax
Energy	2Emin
N	20

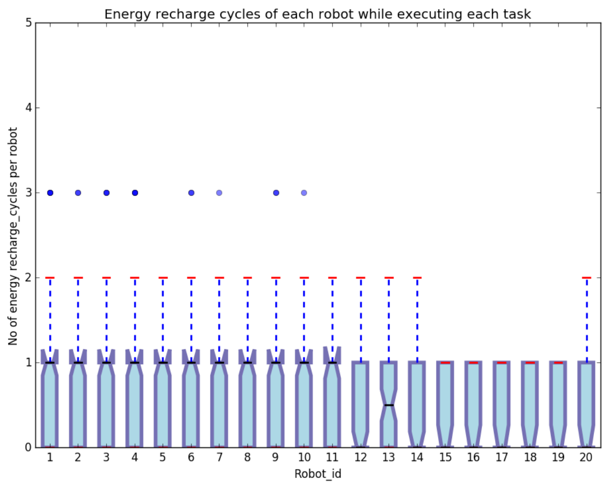
Inference

The number of recharge cycles is slightly lower in nearest relayer strategy than Max energy relayer strategy.

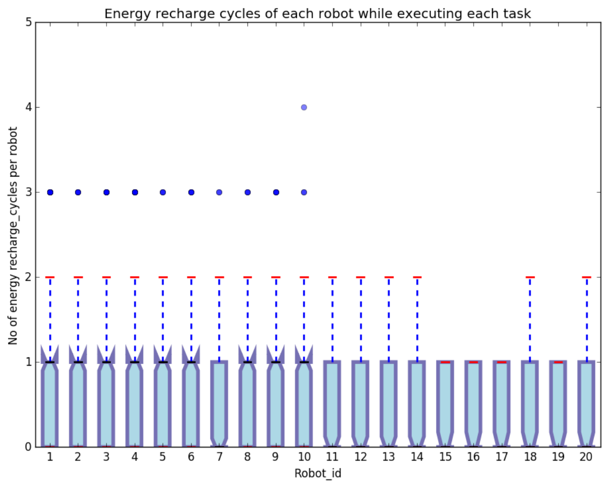
– Max Energy Relayer Strategy: We can observe that around 8 out of 20 robots have zero recharge cycles.

– Nearest Relayer Strategy: We can observe that 10 out of 20 robots have zero recharge cycles.

Verdict: Strategy 1 performs slightly better than Strategy 0 in terms of the number or recharge cycles per robot.



(a) Strategy = Max Energy Relayer



(b) Strategy = Nearest Relayer

Figure 4.19: Impact of varying the strategy for relayer choice on the energy recharges per robot

4.3.4 Final Verdict

The best performance criteria is defined as:

- Equal distances covered by each robot.
- Less idle time and recharge time.

- More data deposition by data relayers than data gatherers.
- Less recharge cycles per robot.

The set of parameters which yield the best performance is given by Table 4.18.

Table 4.18: Parameter Table for Best Performance

Parameters	Values
Energy	2Emin
Timeout	2Tmin
N	20
Strategy for relayer choice	1

4.4 Summary

In this chapter, the experiment design and the results of the dynamic task allocation algorithm is discussed in detail.

- **Criteria for Evaluation**: We introduce the different criteria for evaluation of the algorithm. We propose three metrics: Robot specific, Task specific and Energy specific metrics. These metrics serve to understand the emergent behavior of the system.
- **Experiment Design**: The outline of each experiment and the parameters for the experiment is explained in this section. We define the fixed, variable and the numerical parameters which aid to evaluate the performance of the algorithm.

- **Results**: We present the plots for different criteria and discuss the inference of each parameter on the behavior of the system.

Chapter 5

Conclusion

Chapter 4 concentrated on the evaluation of the Dynamic Task Allocation Strategy. Large set of simulated experiments were analyzed to observe trends in behavior of the system with respect to change in one or more parameters of experiment setup. This chapter derives conclusions based on results in Sec. 5.1. In Sec. 5.2, we present the relevance of this algorithm in real-world application scenarios. We conclude with listing the future prospects of this work in Sec. 5.3.

5.1 Summary of Contributions

The dynamic task allocation algorithm studied in this work propose a solution to explore an arena and gather under memory and energy constraints of robots in a swarm. The collective goal of the system is to gather as much data as possible. A the same time, it is also essential that the robots do not die. We tackle the contradicting goal scenario in this work. The dynamic task allocation algorithm assigns two roles to the robots: data gatherer and data relayer. This division of labor ensures that the energy dissipated in traveling to the sink is reduced. This algorithm ensures that the robot never reach zero battery level, by allowing them to recharge at a common docking station. By defining a strategy for choosing relayer robot, this algorithm reduces the energy dissipation of the swarm by choosing better relayer robots. Data deposition time is also saved. We analyze the performance of the algorithm by defining a set of evaluation criteria which tests parameters specific to each task, each robot and the energy of each robot. From the results, we understand that the algorithm is scalable. The algorithm organizes the task distribution among robots such that there is a smooth transition between the roles of each robot and ensures no robot reaches zero battery level.

5.2 Application Scenarios

Robot swarms are useful in the field of area exploration and data gathering. To design a practical solution using robot swarms, we need to consider the memory, energy and communication constraints of the swarm. The proposed dynamic task allocation algorithm focuses on designing an area exploration scheme under the energy and buffer constraints of the robot. This work, on the high level implements the simplest form of supply chain management cycle. Resource limitation is present in all systems and as far as robots are concerned the resources are memory capacity and battery life. As the robots perform different tasks in a supply chain, their resources gets depleted. The idea of dynamic task allocation is applicable under this scenario. The use of certain members of the swarm to refill the depleted resources of the remaining swarm is an useful solution. In this way, we provide a backup system for resource depletion in multi-robot systems.

This dynamic task allocation strategy would be useful in military applications. The first set of robots (data gatherers) could act as the attacker robots and when they are low in ammunition or energy, the next set of robots (data relayers) can refill the depleted resources of the attacker robots. Similar application would be in agriculture when we can deploy two sets of robots, one set for weed removal and the other for weed disposal. Once the tank of weed removal robots is full with weeds, these robots can make use of weed disposers to transfer the weeds to a disposal site.

5.3 Future Work

The future work can be expanded upon improving the task allocation strategy and evaluating the performance of allocation strategy under various aspects of task allocation. Additionally, future work also includes performing experiments with real robots and performing robustness testing.

- **Variable Buffer Capacity**: The amount of data which can be stored in the buffer of the robot can be varied and the performance of the algorithm can be evaluated. This also gives an additional criteria for data transportation task. We need to check the buffer capacity of the relayer robot and the data gatherer should only choose a relayer which can transport entire data of the data gatherer to the sink.

- **Navigation Algorithm**: A more efficient navigation strategy with a better obstacle avoidance scheme which would align with the area exploration can be a future prospect to this work. Algorithms such as MRTA (Multi-Robot Task Allocation), TSA (Traveling Salesman Algorithm) and Potential Field for obstacle avoidance can be potential improvement options.

- **Task C Energy robots**: Apart from the tasks of data gathering and data transportation, a third task can be incorporated in this algorithm. The Role of Energy Robots which would patrol in different sections of the arena and serve other robots when they run out of charge. In this way, the time spent in battery recharge can be avoided and this would improve the performance of the algorithm.

- **Dynamic Exchange locations**: In this work, we consider the location of the data exchange grid to be fixed. As an alternative, the robots can dynamically choose an obstacle free grid to perform data exchange.

- **Data Loss Handling**: As a future work, we can potentially detect data loss and propose methods to handle it. One method would be to revisit the location to gather the data again or convey the loss of information to the robot closest to that location. Data loss handling is an important aspect to improve robustness of the algorithm.

Bibliography

[Agassounon and Martinoli, 2002] Agassounon, W. and Martinoli, A. (2002). A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. In *Proc. of the IEEE Conf. on System, man and Cybernetics SMC-02*, number SWIS-CONF-2002-003.

[Akre et al., 1976] Akre, R., Garnett, W., Donald, J. M., Greene, A., and Landolt, P. (1976). Behavior and colony development of vespula pensylvanica and v. atropilosa (hymenoptera: Vespidae). *Journal of the Kansas Entomological Society*, pages 63–84.

[Anderson and Ratnieks, 2000] Anderson, C. and Ratnieks, F. (2000). Task partitioning in insect societies: novel situations. *Insectes sociaux*, 47(2):198–199.

[Bhadauria et al., 2011] Bhadauria, D., Tekdas, O., and Isler, V. (2011). Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*, 28(3):388–404.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems.* Number 1. Oxford university press.

[Bonabeau et al., 1997] Bonabeau, E., Sobkowski, A., Theraulaz, G., Deneubourg, J.-L., et al. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In *BCEC*, pages 36–45. Citeseer.

[Cao et al., 1997] Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27.

[Chevaleyre et al., 2005] Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Maudet, N., and RodrÍGuez-Aguilar, J. A. (2005). Multiagent resource allocation. *The Knowledge Engineering Review*, 20(2):143–149.

[Dahl et al., 2009] Dahl, T. S., Matarić, M., and Sukhatme, G. S. (2009). Multirobot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6-7):674–687.

[Dorigo et al., 2013] Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., et al. (2013). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71.

[Endriss et al., 2006] Endriss, U., Maudet, N., Sadri, F., and Toni, F. (2006). Negotiating socially optimal allocations of resources. *Journal of artificial intelligence research*, 25:315–348.

[Gerkey and Mataric, 2003] Gerkey, B. P. and Mataric, M. J. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3862–3868 vol.3.

[Guo and Zavlanos, 2017] Guo, M. and Zavlanos, M. M. (2017). Multi-robot data gathering under buffer constraints and intermittent communication. *arXiv preprint arXiv:1706.02092*.

[Hart and Ratnieks, 2001] Hart, A. G. and Ratnieks, F. L. (2001). Task partitioning, division of labour and nest compartmentalisation collectively isolate hazardous waste in the leafcutting ant atta cephalotes. *Behavioral Ecology and Sociobiology*, 49(5):387–392.

[Hoeing et al., 2007] Hoeing, M., Dasgupta, P., Petrov, P., and O'hara, S. (2007). Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 280. ACM.

[Khaluf, 2014] Khaluf, Y. (2014). *Task Allocation in Robot Swarms for Time-Constrained Tasks*. PhD thesis, University of Paderborn.

[Krieger et al., 2000] Krieger, M. J., Billeter, J.-B., and Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992.

[Labella et al., 2004] Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2004). Efficiency and task allocation in prey retrieval. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 274–289. Springer.

[Lauri et al., 2017] Lauri, M., Heinänen, E., and Frintrop, S. (2017). Multi-robot active information gathering with periodic communication. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 851–856. IEEE.

[McLurkin and Yamins, 2005] McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Robotics: Science and Systems*, volume 8. Citeseer.

[Nouyan et al., 2009] Nouyan, S., Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711.

[Pal et al., 2012] Pal, A., Tiwari, R., and Shukla, A. (2012). Multi-robot exploration in wireless environments. *Cognitive Computation*, 4(4):526–542.

[Pinciroli and Beltrame, 2016] Pinciroli, C. and Beltrame, G. (2016). Buzz: An extensible programming language for heterogeneous swarm robotics. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3794–3800. IEEE.

[Pinciroli et al., 2011] Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al. (2011). Argos: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 5027–5034. IEEE.

[Pini et al., 2011] Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., and Birattari, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3-4):283–304.

[Pini et al., 2013] Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., and Birattari, M. (2013). Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive behavior*, 21(2):118–136.

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM.