



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Doctoral Thesis

Improving Robot Team's performance by Passing  
Objects between Robots

Dohee Lee

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

2022

# Improving Robot Team's performance by Passing Objects between Robots

Dohee Lee

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

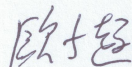
# Improving Robot Team's performance by Passing Object between Robots

A thesis/dissertation submitted to  
Ulsan National Institute of Science and Technology  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

Dohee Lee

06.14.2022 of submission

Approved by



---

Advisor

Tsz-Chiu Au



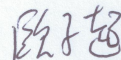
# Improving Robot Team's performance by Passing Object between Robots

Dohee Lee

This certifies that the thesis/dissertation of Tsz-Chiu Au is approved.

06.14.2022 of submission

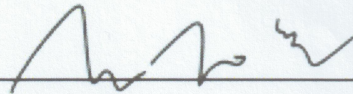
Signature



---

Advisor: Tsz-Chiu Au

Signature



---

Jeong hwan Jeon

Signature



---

Namhoon Lee

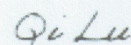
Signature



---

Yuseok Jeon

Signature



---

Qi Lu

## Abstract

A transport robot system is a robotic system in which robots move objects from one place to another place. Most existing transport robot systems perform three tasks: loading an item, moving to another location, and unloading the item. Traditional mobile robots, which carry objects one at a time, is not suitable for repeatedly transporting objects over a long distance. Therefore, in the factory or warehouse environment, they still use conveyor belts to transport a large number of objects. However, the existing conveyor belts are physically fixed in their environments, and it is difficult to reconfigure the layout of a conveyor network. In this thesis, I present three new robotic systems that have the ability to pass objects at a distance between mobile robots. These three robotic systems are mobile conveyor belts, dynamic robot chains, and mobile workstations.

First, conveyor belts are commonly used to transport many objects rapidly and effectively. I present a novel conveyor system called a mobile conveyor line that can autonomously organize itself to transport objects to a given location. In this thesis, I analyze the reachability of multiple mobile conveyor belts and present an algorithm to verify the reachability of a specified destination, as well as a way to generate a configuration for connecting conveyor belts to reach the destination. The key results include a complete set of equations describing the reachable set of a mobile conveyor belt on a flat surface, which leads to an effective probabilistic strategy for autonomous configuration. The results of the experiment demonstrated the overlap effect, which states that reachable sets frequently overlap. This system can be suitable for locations where it is difficult to install a conveyor line, such as disaster zones.

Second, I present to use mobile conveyor belts in foraging tasks in environments with obstacles. Foraging robots can form a dynamic robot chain network that can quickly send resources received from other foraging robots to a collecting zone called a depot area. A robot chain is essentially a sequence of mobile robots with the ability to quickly pass resources at a long distance. A dynamic robot chain network is a network of robot chains that allow the branches of the robot chains to connect multiple resource clusters. By allowing branching, the traffic near the end of the robot chain network can be distributed to several branches, and congestion can be avoided. The dynamic robot chain network leverages mobility to relocate, reduce collection time for other robots, and quickly send resources received from other foraging robots to the depot area. The key result is the formation of robot chains capable of overcoming the two major limitations of existing dynamic depot foraging systems: the long travel distance for delivery and congestion near the central collection zone. In the experiments, given the same number of robots, a dynamic robot chain network outperformed existing dynamic depots in multiple-place foraging problems.



Third, I consider the idea of mobile workstations, which integrate mobile platforms with production machinery to improve efficiency by overlapping production time and delivery time. I describe a task planning algorithm for multiple mobile workstations and offer a model of mobile workstations and their jobs. This planning problem for mobile workstations includes the features of both traveling salesman problems (TSP) and job shop scheduling problems (JSP). For planning, I present two algorithms: a) a complete search algorithm that offers a minimum makespan plan and b) a local search in the space of task graphs to offer suboptimal plans quickly. According to the experiments, the second algorithm can generate near-optimal temporal plans when the number of jobs is small. In addition, the second algorithm can generate noticeably shorter plans than a version of the job shop scheduling algorithm and SGPlan 5 when the number of jobs is large.

This research shows that transport robot systems could work together with other robots or machines in various environments to overcome the limitations of existing systems for the environments. A mobile conveyor line can pass quickly objects at a long distance and can apply to many different environments by overcoming the existing problem of conveyor belts. By using mobile conveyor belts, the robots have the ability to pass objects at a distance between mobile robots to improve the performance of foraging tasks by overcoming the long travel distance for delivery and congestion near the central collection zone. In addition, a mobile workstation can handle the tasks that transport the production of goods to users. By paralleling the production time and the movement, a mobile workstation can substantially shorten the time it takes to deliver products to customers.





## Contents

|     |  |    |
|-----|--|----|
| I   | Introduction . . . . .                                     | 1  |
| II  | Related Work . . . . .                                     | 3  |
| III | Technical Background . . . . .                             | 7  |
|     | 3.1 Swarm Robotics . . . . .                               | 7  |
|     | 3.2 Foraging Problem in Robotics . . . . .                 | 9  |
|     | 3.3 Object Sorting and Clustering . . . . .                | 16 |
|     | 3.4 Navigation . . . . .                                   | 22 |
|     | 3.5 Path Formation . . . . .                               | 24 |
|     | 3.6 Task allocation . . . . .                              | 25 |
|     | 3.7 Other tasks for swarm robotics . . . . .               | 29 |
|     | 3.8 The central-place foraging algorithm (CPFA) . . . . .  | 29 |
|     | 3.9 The multiple-place foraging algorithm (MPFA) . . . . . | 30 |
|     | 3.10 Simulators for Swarm Robotics . . . . .               | 31 |
|     | 3.11 Foot-bot . . . . .                                    | 33 |
|     | 3.12 Job Shop Problem (JSP) . . . . .                      | 33 |
|     | 3.13 Travelling Salesman Problems(TSPs) . . . . .          | 34 |
| IV  | A Mobile Conveyor Belt . . . . .                           | 36 |

|     |  |    |
|-----|--|----|
| 4.1 | Intorduction . . . . .   | 36 |
| 4.2 | A Mobile Conveyor Belt . . . . .   | 38 |
| 4.3 | Configurations of Mobile Conveyor Lines . . . . .                          | 39 |
| 4.4 | Reachability Analysis . . . . .  | 40 |
| 4.5 | Generating a Configuration for Reachable Points . . . . .                  | 48 |
| 4.6 | The Automatic Configuration Algorithm and the Overlapping Effect . . . . . | 48 |
| 4.7 | Experimental Evaluation . . . . .  | 50 |
| 4.8 | Summary . . . . .  | 53 |
| V   | Dynamic Robot Chains . . . . .   | 54 |
| 5.1 | Introduction . . . . .   | 55 |
| 5.2 | Foraging Tasks With Robot Chains . . . . .                                 | 56 |
| 5.3 | The Controller For Foraging Robots . . . . .                               | 60 |
| 5.4 | Relocation of Robot Chains . . . . .                                       | 62 |
| 5.5 | Experimental Configurations . . . . .                                      | 64 |
| 5.6 | Experimental Results . . . . .   | 64 |
| 5.7 | Summary . . . . .  | 66 |
| VI  | Dynamic Robot Chain Networks . . . . .                                     | 69 |
| 6.1 | Introduction . . . . .   | 69 |
| 6.2 | Foraging with Robot Chain Network . . . . .                                | 70 |
| 6.3 | Foraging Robots Behavior . . . . .   | 72 |
| 6.4 | Modification of Robot Chain Network . . . . .                              | 72 |

|      |  |    |
|------|--|----|
| 6.5  | The High-level Controller . . . . .                                | 73 |
| 6.6  | Experimental Configurations . . . . .                              | 74 |
| 6.7  | Experimental Results . . . . .                                     | 76 |
| 6.8  | Summary . . . . .  | 80 |
| VII  | Mobile Workstations . . . . .                                      | 81 |
| 7.1  | Introduction . . . . .   | 81 |
| 7.2  | Mobile Microwave Robots . . . . .                                  | 84 |
| 7.3  | The Scheduling Problem for a Team of Mobile Workstations . . . . . | 84 |
| 7.4  | Planning Algorithms . . . . .                                      | 87 |
| 7.5  | Experimental Evaluation . . . . .                                  | 91 |
| 7.6  | Summary . . . . .  | 93 |
| VIII | Conclusion and Future work . . . . .                               | 95 |
|      | References . . . . .   | 98 |

## List of Figures

|    |  |    |
|----|--|----|
| 1  | A conveyor line made by the Miniveyors's portable conveyor belt . . . . .  | 1  |
| 2  | A field delimitation of three key words: mobile robotics, multi-robot systems, and swarm robotics. Swarm robotics field is include in the Multi-Robot Systems field, whereas swarm robotics field is include in the Mobile Robot field. . . . .  | 7  |
| 3  | The foraging tasks of foot-bots in the simulator. The robots search for resources and grab them and bring to the depot. A grey area serves as a depot and resources are distributed in the white area. . . . .   | 9  |
| 4  | The illustration of the Foot-bots forming a dynamic chain for the navigation. The red robots form the chain and work as a beacon using light sensors. . . . .  | 11 |
| 5  | The example of the bucket bridge made by robots which introduced in [1]. Each robot transfer the goods to next robot on the bucket bridge made by robots. The last robot put the goods to the collecting zone. . . . .   | 12 |
| 6  | Communication topology model defined by the distance between robots: bold arrows represent the information flow, large circles are communication range. . . . .  | 14 |
| 7  | The illustration of the CPFA [2]. . . . .  | 29 |
| 8  | The illustration of DDSA. Utilizing the spiral search pattern, the robots should traverse a continuous plane. They will begin at a central collecting location and move outward. The targets are shown as little gray cylinder in a distribution that is only partially clustered. . . . . | 30 |
| 9  | Using the k-means++ clustering method, an example of a dynamically assigned depot. Depots (dark red solid circles) are positioned at the cluster centers. The targets (black squares) are divided into four distinct groups (red ellipses). . . . .  | 31 |
| 10 | The ARGoS simulator for swarm robotics. . . . .  | 32 |
| 11 | An example of Travelling Salesman Problem . . . . .  | 34 |



|    |  |    |
|----|--|----|
| 12 | A mobile conveyor belt. . . . .  | 37 |
| 13 | The design of the mobile conveyor belt. The mobile conveyor belt can move objects from starting point to end point. . . . .  | 38 |
| 14 | Case 1: $2\Delta_{\max} \leq H_{\max} - H_{\min}$ and $\Delta_{\max} \leq H_{\min}$ . . . . .  | 43 |
| 15 | Case 2: $\Delta_{\max} \leq H_{\min}$ and $0 \leq H_{\max} - H_{\min} \leq 2\Delta_{\max}$ . . . . .   | 44 |
| 16 | Case 3: $H_{\min} \leq \Delta_{\max}$ and $2\Delta_{\max} \leq H_{\max} - \Delta_{\max}$ . . . . .   | 45 |
| 17 | Case 4: $H_{\min} \leq \Delta_{\max}$ and $2H_{\min} \leq H_{\max} - \Delta_{\max} \leq 2\Delta_{\max}$ . . . . .  | 46 |
| 18 | Case 5: $H_{\min} \leq \Delta_{\max}$ and $0 \leq H_{\max} - \Delta_{\max} \leq 2H_{\min}$ . . . . .   | 46 |
| 19 | Case 6: $H_{\min} \leq \Delta_{\max}$ and $H_{\min} \leq H_{\max} \leq \Delta_{\max}$ . . . . .  | 47 |
| 20 | Exemplifications of the reachable sets in two cases; the Case 1 and Case 2. The examples of Case 1 have $H_{\min} = 5m$ and the examples of Case 2 have $H_{\min} = 16m$ . The remaining parameters are $H_{\max} = 20m$ , $L = 10m$ , $D_{\max} = 5m$ , $D_{\min} = 2m$ , and $\Theta_{\max} = 30^\circ$ . . . . .  | 48 |
| 21 | The illustration of the search space of the automatic configuration for $M = 4$ and $N \geq 3$ . . . . .   | 50 |
| 22 | The successful rates of the algorithm in a two-dimensional environment. . . . .  | 52 |
| 23 | The successful rates of the algorithm in a three-dimensional environment. . . . .  | 52 |
| 24 | Two mobile conveyor belts form a robot chain. . . . .  | 54 |
| 25 | The example of four robot chains deploy in the arena. The magenta dots are robots of the robot-chain and the blue dots are foraging robots. . . . .  | 57 |
| 26 | The robot use the lidar to get information. The red dotted line show the lidar's sensing range. . . . .  | 59 |
| 28 | The relocation of robot chain. The current robot chain (grey lines) relocates to a new location (red small circle). Robots (colored circles) explore the regions when they travel to the new location. An obstacle (blue rectangle) is discovered and two robot chains (green and purple lines) are computed. The last robot of robot chain 2 (purple lines) is closer to the new location than the robot chain 1 (green lines). . . . . | 63 |

|    |  |    |
|----|--|----|
| 29 | Foraging performance with varying numbers of robots in various areas in Experiment 2-1.  | 65 |
| 30 | The collision time of each swarm in Experiment 2-1. . . . .  | 66 |
| 31 | Foraging performance with varying numbers of robots in various areas in Experiment 2-2   | 67 |
| 32 | The example of four robot chains networks in the arena. The magenta dots are robots of the robot-chain, whereas the blue dots are foraging robots. . . . . | 70 |
| 33 | Foraging performance in Experiment 3-1. . . . .  | 76 |
| 34 | The collision time of each swarm in Experiment 3-1. . . . .  | 76 |
| 35 | Foraging performance in Experiment 3-2. . . . .  | 78 |
| 36 | The collision time of each swarm in Experiment 3-2. . . . .  | 78 |
| 37 | Foraging performance using different $t_{\text{protect}}$ in Experiment 4-1. . . . .   | 79 |
| 38 | Relocating performance using a different $t_{\text{explore}}$ in Experiment 4-2. . . . .   | 79 |
| 39 | A mobile microwave robot. . . . .  | 81 |
| 40 | A mobile printer robot. . . . .  | 82 |
| 41 | A scenario involving a mobile coffee making robot. . . . .   | 83 |
| 42 | A model of the mobile coffee making robot. . . . .   | 84 |
| 43 | The example of three mobile coffee making robot's jobs. . . . .  | 86 |
| 44 | A task graph for the jobs in Figure 43 . . . . .   | 88 |
| 45 | An optimal temporal plan for the three jobs in Figure 43. . . . .  | 89 |
| 46 | Running time according to different numbers of nodes . . . . .   | 93 |
| 47 | Plan length according to different numbers of nodes . . . . .  | 93 |
| 48 | Running times according to different numbers of nodes . . . . .  | 94 |
| 49 | Plan length according to different numbers of nodes . . . . .  | 94 |

# List of Tables

|   |  |    |
|---|--|----|
| 1 | Definitions of the symbols in Figures 14 to 19 and Table 2. . . . .    | 42 |
| 2 | The valid range of $\theta$ in all cases in Figures 14 to 19 . . . . . | 42 |
| 3 | Range of the parameters for experiments in 2D environments. . . . .    | 51 |
| 4 | The Configuration in Experiment 2-1 . . . . .                          | 64 |
| 5 | The Configuration in Experiment 2-2 . . . . .                          | 65 |
| 6 | The configuration of experiments 3-1 & 3-2 . . . . .                   | 75 |
| 7 | The configuration of experiments 4-1 . . . . .                         | 75 |
| 8 | The Configuration of experiment 4-2 . . . . .                          | 75 |

## I Introduction

Various mobile robot tasks involve moving objects from one location to another. However, few of the existing mobile robots are designed to efficiently transport a large number of objects. In the case of a mobile robot on a rescue mission, for instance, the robot performs a round trip to transport items from a hazardous place to a safe place. High-payload robots can complete the task in fewer round trips, but it makes more sense to use conveyor belts to transport a large number of items. Conveyor belts, which are widely used to transport and move materials and items, are useful for the majority of robotic system handling. The majority of existing conveyor belts are utilized indoors in a stationary position. Few conveyor belts are designed to be portable and deployable by humans in outdoor environments. As an example, the portable conveyors offered by Miniveyors are lightweight conveyor systems that facilitate the rapid and efficient transport of a variety of materials. These can aid robots in numerous moving object tasks, including rescue missions. However, because these conveyor systems must still be installed by human workers, it is challenging to install them in harsh environments such as disaster zones. Therefore, I propose to consider a conveyor system as an autonomous robot with its own mobility. This conveyor belt robot is referred to as a mobile conveyor belt.

The following is the research statement of this thesis:

If we endow mobile robots with the ability of passing objects at a distance between robots, how can a group of robots leverage this ability to enhance their performance in tasks that require them to work as a team?

A mobile robot is a robot that has the ability to move around and not be fixed to physical locations. The meaning of robot has the ability what robot can do something. The passing objects are that robots can move an object to another location or pass the object to other robots. A group of robots is some

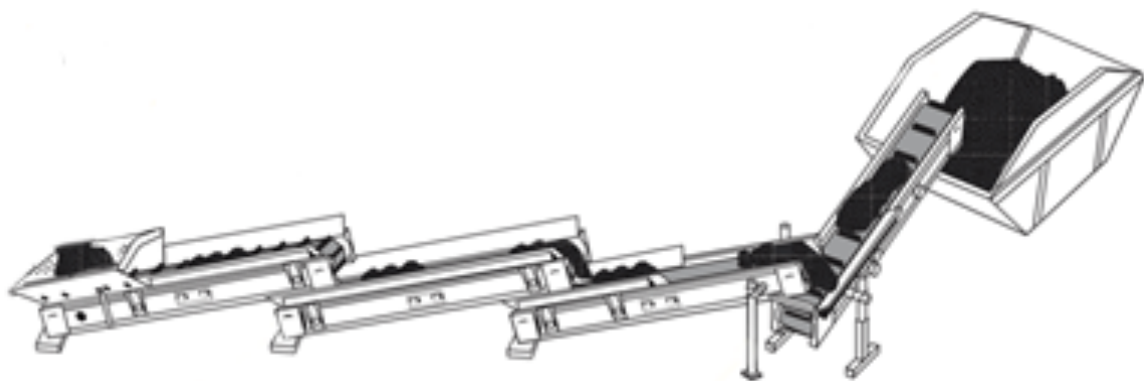


Figure 1: A conveyor line made by the Miniveyors's portable conveyor belt



number of robots in the system. Leverage is a benefit of doing something. Performance is a measurement of how well a system is accomplishing the system's goal. The task means a work performed by a robot, such as a passing object, foraging object, etc. A team is a group of robots that work together to achieve the goal of the system.

In this dissertation, I propose a mobile conveyor belt consisting of a conventional conveyor belt attached to a mobile platform (see Figure. 12) and study how to configure a conveyor line with a number of mobile conveyor belts automatically and without human interaction. One of the challenges of this system is how to link multiple mobile conveyor belts together to construct a conveyor line that transports items from one place to another. I provide a comprehensive set of equations to define the set of places that can be reached by a mobile conveyor belt, given its physical constraints, as well as a probabilistic algorithm to determine whether it is possible to use  $N$  mobile conveyor belts to connect a position to its destination on a flat surface. I apply the concept of a mobile conveyor line to the foraging problem to improve the foraging performance of the foraging robotic system. The mobile conveyor belt can reduce foraging robots' travel time and collision time. In the foraging problem, I present a new algorithm to form a mobile conveyor line called a dynamic robot chain. The algorithm is capable of generating a strategy for an environment with obstacles. In addition, I presented a new robot system called a mobile workstation that integrates mobile platforms with production machinery to improve productivity by overlapping production time and delivery time. I described a number of algorithms that let a number of mobile workstations perform the jobs.

## II Related Work

Due to the fact that robots cannot normally pass huge quantities of objects, I considered adding some robot capabilities to the conveyor belt so that the conveyor systems can be utilized in scenarios that no other robotic system can handle. There are few studies that address the overall configuration of the conveyor systems. Rather, the key points of these studies focus on hardware design improvement. Li and Li [3] used a hardware modeling program called AMESIM to assess the conveyor system's performance and discovered that adding flywheels to motors may substantially improve the performance of the system. Bindzar et al. [4] developed a three-dimensional mathematical model of a conveyor belt, which is used to assess its performance under stress loading. Pitcher [5] investigated the loss of strength in three basic kinds of connections between conveyor belts and concluded that some connection designs cannot function to its full capability. Nuttall [6] investigated the design of many powered belt conveyors, as well as spreading driving energy and stress control, to achieve a balance between locally applied driving energy and the related resistances. Donis [7] examined the optimal positioning of the conveyor belt weigher based on the stiffness of the belt.

When conveyors are linked together, they construct a kinematic chain akin to a snake-shape robot, a kind of hyper-redundant manipulator whose configuration problem is to identify the paths of the configuration space with collision-free pathways [8]. Related to a flexible conveyor train (FCT), the mobile conveyor lines are mining-related continuous haulage solutions. This conveyor system is more flexible than the mobile conveyor line examined in this research, but its automated configuration problems are also more complex.

Several modifications have been made to the conveyor belts. Mankge [9] and McNearny and Nie [10] presented a conveyor system working underground for use in mining environments. Especially, [9] examined constraint management of conveyor haulage systems using his simulation. Hou and Meng [11] researched the dynamic characteristics of a conveyor belt influenced by the conveyor belt type and demonstrated that the propagation speed of stress waves increases with increasing tensile load. Ananth et al. [12] proposed a design for a conveyor system that considers belt speed, gearbox options, belt width, pulley, and motors, among other factors. Karolewski and Ligocki [13] constructed the mathematical models of long conveyors belt that include features such as the belt's several operating phases and the wave behavior of the tape.

The design of conveyor systems also prioritizes energy efficiency. Zhang and Xia [14] adjusted operating settings to enhance the energy efficiency of belt conveyors. Fonseca et al. [15] presented an expert system method to conveyor selection in order to aid in the selection of conveyor equipment and shown that it outperformed human experts. Lauhoff [16] analyzed a proposal about the speed of conveyor belts to save energy and concluded the conventional filling levels are inadequate.

In central-place foraging, robots gather dispersed resources and transfer them to a central gathering zone [17, 18]. Hecker and Moses devised the stochastic central place foraging algorithm (CPFA) for robotic swarming systems. Flanagan et al. [19] significantly improved the gathering performance of CPFAs by sharing resource locations. Fricke et al. [20] introduced a distributed deterministic spiral search algorithm (DDSA) that ensures a thorough search of the whole arena. While the DDSA surpasses the CPFA in simulation, the CPFA performs somewhat better in practical experiments [21].

Lu et al. [22, 23] developed a multiple-place foraging algorithm (MPFA) akin to global courier and shipping system in which numerous distributed warehouses effectively distribute and gather materials. This algorithm is motivated by observed foraging behavior of polydomous colonies of the wasps and ants [24] with multiple sleep site of spider monkeys and nests [25]. The MPFA produces better foraging performance and shorter trip lengths compared to the CPFA. Lu et al. [26] enhanced the MPFA systems's foraging performance by using carrier robots called dynamic depots  $MPFA_{dynamic}$ , in which the dynamic depots can carry supplies directly to the center. However, resource-transporting robots still need to travel extensive distances. Grammatical Evolution is used by Ferrante et al. [27] to separate a foraging activity into finding and delivering activities. Pini et al. [28] proposed an approach for the static division of foraging robot swarms.

Lee et al. [29] expanded the research of MPFA with dynamic depots  $MPFA_{dynamic}$  in [26] via substituting mobile robot chains for the dynamic depot. In the past, robot chains were utilized to localize other robots by functioning as just a collection of fixed location beacons [30] or as navigation to inform the local information [31]. However, the robot chains are utilized to move materials via some delivery drones [32] or mobile conveyor lines [33].

The Euclidean Steiner tree problem is one famous NP-hard problem, but there is a polynomial-time approximation scheme such that can provide a near-optimal in polynomial time [34]. Parque [35] presented a method inspired by nature for computing the typology of obstacle-avoiding Steiner trees taking into account  $n$ -star topologies. [36] an accurate but inefficient method for Euclidean Steiner tree problems in obstacle spaces. Garrote [37] offered a heuristic approach for a problem of constructing Euclidean Steiner trees when the two-dimensional plane may be partitioned into polygonal parts. The Euclidean Steiner tree solver is implemented by the method described in [38] with changes to handle obstructions.

In this dissertation, I expand upon the research of MPFA with dynamic depots [26] via substituting dynamic depots with mobile robot-chains. In the past, the robot chain has been conceptualized as a series of virtually linked robots that assist in the localization of other robots by functioning as just a collection as fixed-location beacons [30] or as navigation to inform the local information [31]. However, in my cases, I form a link between robots of the robot chain using UAVs or conveyor belts to transfer the resources [33]. For example, robots can deliver resources between two moving platforms via UAV [32],

or resources are transported between two robots by throwing them [39].

In addition, I tried to improve the performance of service robots for delivering the items. The recent advancements in warehouse and industrial robots, particularly the automated guided vehicle (AGV) produced by Amazon Robotics (such as Kiva robotic systems), which has changed the system operations of huge distribution centers, largely inspired this concept. These warehouse robots like a kiva robotic system transport storage containers to human packers in packing area [40]. I believe further time savings may be realized if packing can begin as soon as a pod begins to move. Due of numerous physical restrictions, this concept may not be feasible in warehouse settings, but I examine it in several contexts, as with service robotic systems. The robotic system such as a Kiva system requires a scheduling algorithm, which utilize A\* search to build pathways in a two-dimensional grid while minimizing trip time. Kiva systems disclosed in [4] the techniques used for resource allocations and the solution of the relevant global optimization problem.

In recent years, there have been several manufacturing robot competitions. Robocup@Work is a notable event that focuses on the usage of robots in work-related contexts [14]. Mobile manipulators in Robocup@Work perform duties including automation, part handling, and manufacturing to general logistics (like as [16]). In the most recent RoboCup Logistics League competition, greater emphasis was placed on the planning of in-factory logistics solutions. In this robotic system competition, the system must retrieve raw materials from the input storage area, transport them between stationary equipment, operate the equipment, and deliver the final product. The IEEE Virtual Manufacturing Automation Challenge (VMAC) involves the operation of multiple AGVs to transport a variety of goods between numerous input locations and output locations in an industrial environment, like a warehouse setting. [15]. This notion of mobile workstations is motivated in part by the immobility of the devices in these events.

As a subset of the task planning problem, the pickup and delivery problems(PDPs) have been investigated in numerous contexts [10].A subtype of vehicle routing problems that fall into this category are problems that require the rapid pickup and delivery of objects or people [9]. These problems are NP-hard, like the traveling salesman problems (TSPs), and there is no effective way of optimally solving them. There are heuristic options like the one-to-one approach [12] and Load LIFO [11]. KARolewsKi [13] analyzed a generalized version of PDPs that includes several qualities seen in a range of different PDPs. Obviously, this problem is much more challenging than General PDPs, as I am simultaneously attempting to solve two tough problems (Task plannings and PDP). The taxi dispatch problems [8] are related to PDPs, excluding the possible waiting time of the passenger [41].

The planning problems for mobile workstations are similar to a job shop scheduling problems(JSPs), the well-known optimization problems involving the allocation of jobs or tasks to machinery while reducing the makespan. Many researcher already developed the efficient JSP solvers (such as [42] and

[43]). The distinction between JSP and mobile workstation's problem is JSP ignores the mobility of the machine. Although Beck et al. [44] offered a technique to express vehicle routing problems (VRPs) as a JSPs and vice versa, they did not attempt to integrate the VRPs and JSPs into a one problem. In JSPs, each job consists of a series of actions that must be done in sequential sequence, and each operation should be performed by a certain machine [45]. In the problem of planning mobile workstations, each subtask of the jobs might be partly allocated to several mobile workstations. In contrast to simultaneous motion and task planning, this research focused on path and task planning for mobile workstations rather than motion planning.

### III Technical Background

This section covers the background of the relevant technologies to this research. The objective of this research is to propose novel robotic systems that have an ability of passing objects at a distance between mobile robots. Therefore, I discuss the technologies associated with conveyor belts, swarm intelligence, and planning in this section.

#### 3.1 Swarm Robotics

Swarm Robotics is a topic of study that examines how systems run by numerous autonomous robots or multiple autonomous agents may complete collective tasks that cannot be accomplished by individual robots or are performed more successfully by robots working in groups. Dudek et al. [46] classified robot-executable activities as intrinsically single-agent, the tasks that may benefit from the usage of multiple agents, conventionally multi-agent, and requiring multiple agents. The discipline of swarm robotics systems focus on the last three categories, and previous research has demonstrated in a variety of application domains that utilizing a large number of agents to handle a task in a distributed manner permits working with agents that are significantly less complex at the individual level.

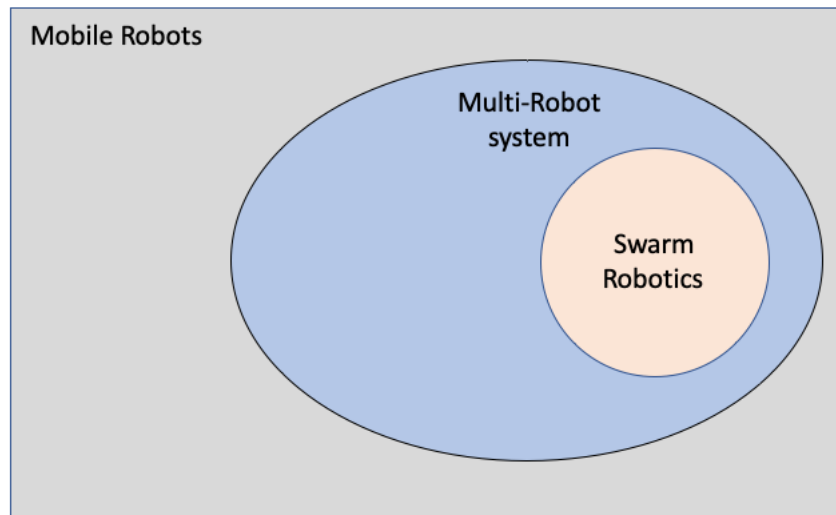


Figure 2: A field delimitation of three key words: mobile robotics, multi-robot systems, and swarm robotics. Swarm robotics field is include in the Multi-Robot Systems field, whereas swarm robotics field is include in the Mobile Robot field.

Şahin [47] has identified three desired properties as the primary motivation for swarm robotics research; robustness, scalability, and flexibility. Şahin developed a set of criteria for describing and categorizing research in swarm robotics from neighboring fields, including the following: Autonomous robots are robots that are capable of moving and interacting with their environment without the need for centralized control. The job at hand is one that a large number of autonomous robots can do collectively, indicating that the robotic system was developed with scalability in mind. The swarm consists of a limited number of homogeneous groups of robots, with the emphasis focused on vast numbers of



people who are identical to one another, rather than on different teams that have been centrally organized and assigned predetermined roles for each robot. The capabilities of a single robot, such as its sensing, computation, and communication skills, are insufficient when compared to the difficult job that must be completed collectively. The fact that each robot handled sensing and communication locally guaranteed that interaction between robots in swarms remained scattered and independent of coordinating mechanisms, which would have prohibited scalability. The notion of swarm robotics is based on the observed collective behavior of several kinds of live creatures in nature. This collective behavior happens when local interactions between people and their environment result in a group of autonomous agents accomplishing complex tasks in a dispersed way without the usage of a central control unit. Despite being seen as a limitation, the localisation of communication and interactions has a positive impact on the scalability and robustness of the system, and is thus chosen over global sensing and communication.

In swarm robotics, the term "swarm intelligence" refers to the superior capabilities of a swarm of agents over the capabilities of individual agents. Local events produced by swarm members during task execution translate into global behavior that often surpasses individual capabilities to the extent that many collective tasks can be accomplished by robots not expressly intended to do so. The global, macroscopic dynamics are supposed to originate through interactions between swarm members and their surroundings.

The capacity to accomplish global goals at the swarm level utilizing distributed algorithms operating at the individual level comes with a cost: it is often difficult to design the behavior of individual robots so as to optimize their contribution to the overall performance. Scientists with an interest in swarm robotics have undertaken extensive research on this topic and sought to resolve it via simulation, modeling, and learning. The simulation, in which a given multi-robot scenario is replicated in a virtual environment in which a computer program simulates robot capabilities (sensors and actuators) and interactions, enables the performance of a robot swarm to be evaluated through repeated runs of an experiment, eliminating or reducing the need for time-consuming experiments with real robots and facilitating algorithm optimization via a trial-and-error method. Simulation also permits the assessment of a robot's efficacy. With the use of such formulae, the effect of algorithm parameters can be examined immediately, and vital insights into the global dynamics of the swarm may be gained intuitively. Modeling is achieved by connecting individual-level control settings to swarm-level dynamics using mathematical methods. The process of changing algorithm parameters based on prior experience is called learning. In offline techniques, the parameter optimization process is included in the design of robot controllers. In online approaches, robots dynamically change their control parameters depending on their perception of the environment. The process of modifying algorithm parameters based on past experience is referred to as learning.

Several researchers in the area of swarm robotics have resorted to offline learning strategies such as artificial evolution and repetitive robot trials. In each experiment, fitness functions were evaluated. This function gives an evaluation of an algorithm's performance when given a cluster-level task. When starting a new iteration of the process, the initial values of the algorithm's parameters are utilized to decide which parameter values offer the best results and then to train the robots using those parameter values.

In a way similar to how creatures evolve in the natural world, robots are capable of modifying their behavior across several generations in order to attain their goal. Despite the fact that neural networks are the most prevalent kind of robot controller used in artificial evolution, a new research has examined the usage of alternative approaches, such as rule-based grammatical evolution, in artificial evolution.

### 3.2 Foraging Problem in Robotics

Collective foraging is a popular topic of research in the realm of swarm robotics. This study was inspired by the foraging behavior of ant colonies. Ants and other social animals use food sources effectively by leveraging local relationships. In an artificial swarm robotics system, a certain area is designated as the "nest" or "depot" for gathering things. The swarm's job is to find things all over the environment and bring them to the depot. Multi-foraging is an extension of foraging that requires bringing numerous kinds of materials to preset depots. This is a continuation of the initial action of foraging. This kind of work has several practical uses, including the removal of landmines and other dangers, search and rescue operations, and even the finding of other planets. Several studies have examined the dynamics of swarm energy derived from foraging. Things gathered by robots and carried to the depot, which are equivalent to food sources, supply energy to the swarm; nevertheless, search activity depletes energy. Each robot's control algorithm should determine when the robot should explore the environment for items to bring to the depot and when the robot should stay idle in order to optimize net energy income revenue. This part of the activity will not be discussed in this section since its value resides more in the dynamic assignment of tasks among the robots than in the actual foraging process.

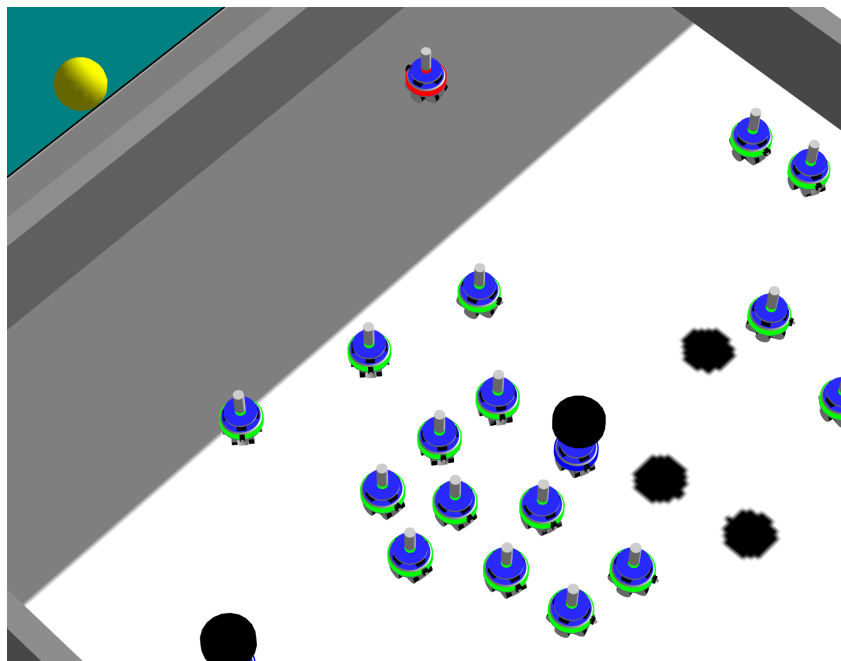


Figure 3: The foraging tasks of foot-bots in the simulator. The robots search for resources and grab them and bring to the depot. A grey area serves as a depot and resources are distributed in the white area.

The task of the foraging may be split into two main types of sub-tasks: robots are either finding the

resources in the problem space or transporting something to deliver back to a depot. The execution of each of these sub-tasks in a group of robots can be aided by robot cooperation mechanisms. Cooperation can also be beneficial for reducing the negative impacts of robot interference and so boosting the system's scalability. To achieve cooperation, individuals must interact with one another such that each robot's actions are influenced by those of the rest of the swarm. Such interaction can occur in several ways, including using a shared memory, making local changes to the environment, and exchanging information directly between agents. The subsections that follow go into each type of communication in greater depth, using examples from previous research.

### **Path Formation**

Path formation is the process of establishing one or more preferred pathways inside the foraging problem. The robots will follow the route to reach their target as efficiently as possible, whether they are searching for resources or transporting them. Robots progressively establish these routes as they engage in foraging, and they may vanish if the cause of their formation is no longer relevant, such as when there is no longer an item source. In studies where robots may interact with one another via shared memory or broadcast communications, suppose one of the robots conveys the path it travelled to reach a site of interest. Taking advantage of the information supplied by the first robot, the other members of the swarm may optimize their own routes using this data.

This concept is used in [48]: When a robot, which initially wanders randomly in search of resources, discovers a resource to transport to the depot, it remembers the route it travelled to get there in shared memory. This enables other robots to retrace their movements to locate the resource. Other searching robots alter their movements to follow the path read from the shared memory, and whenever they discover anything, they communicate with one another about their journey and what they discovered along the way. This method entails progressively creating, in shared memory, a map of preferred travel routes in order to maximize foraging activities. In [49], for instance, genuine robots were utilized to test out different real-world situations.

Similar to the approach stated in [48], shared trail information [50] will gradually be collected. In order to alleviate the interference issue that arises when the path from rich resources to the depot becomes crowded and robots going in opposite directions crash with one another, the route must be redesigned. The core algorithm for following routes has been updated such that robots heading toward a goal are repelled by paths that have been used to accomplish other goals and are attracted to paths that have been used to attain the same goal by other robots. For instance, a robot that is supplying resources and is traveling toward the depot will strive to avoid the path used by robots that are traveling to the resource.

Methods of pheromone transmission, based after the behavior of ants, include tagging the environment with markers that guide robots down the most time- and resource-efficient path to their goal. The system [51] employs a mechanism comparable to pheromone-based communication. Robots transferring a resource will leave "crumbs" on their route back to the depot, which will attract robots that are hunting for something. According to the study of Hamann and Worn [52], foraging robots are motivated

by a pheromone that is emitted by robots when they return to the depot to deposit an obtained material. This strategy produces a pheromone gradient in the arena, which robots follow in their search for resources; the quantity of pheromone produced by robots is largest where resources have been gathered and diminishes as they approach the depot. According to study mentioned in [53], robots will only emit a pheromone upon returning to their starting location if they see other objects in close proximity to the resource they are bringing.

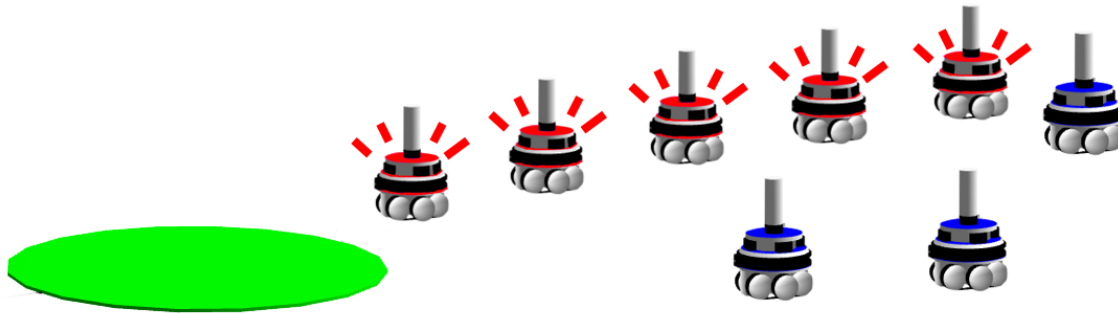


Figure 4: The illustration of the Foot-bots forming a dynamic chain for the navigation. The red robots form the chain and work as a beacon using light sensors.

In a number of research, robots have constructed group of interconnected persons that aid other robots in their search efforts. This is comparable to the scent trails that animals leave behind to help other species locate food sources or storage areas. According to [54], the depot emits a beacon signal that can be detected by robots up to a certain distance; a robot that reaches the beacon coverage area's limits stops searching for resources and emits a new beacon, thereby expanding the search area for the other robots, which may become beacon emitters if they reach the coverage area limits of the first robot's beacon signal's coverage area. A robot that approaches the boundaries of the beacon's coverage region is unable to navigate farther. Using this technology, group of robots that emit beacons are created, allowing searching robots to explore new regions while returning to the depot quickly. A group of robots in physical interaction with one another starts at the depot at [55]. Fundamental behaviors allow robots to do tasks such as building and growing the chain, following the chain (such as when returning to the depot), and performing excursions near the chain (to search for resource).

Each robot in [56] may either actively participate in the foraging process or act as a beacon. To construct groups, robots that function as beacons begin their journey at either a depot or a resource source. Each beacon has an integer representing its position in the group, with larger numbers indicating greater distances from the depot or resource. These robot chains provide a simulated gradient field that foraging robots may exploit to effectively reach their goal location. Upon detecting a local connection with beacon emitters, the foraging robots used the field to do so.

The aforementioned method is included into a following study [57] in an adaptive system capable of using many algorithms, dynamically switching from one algorithm to the next at the swarm level dependent on the features of the environment. This system has been dispersed. The robots start the

foraging process by conducting a virtual gradient-based algorithm, which is more efficient when resource sources are situated near to the depot. If this algorithm fails (indicated by all robots turning into beacons), a separate algorithm is initiated in which robots form a single chain originating at the depot and sweep a circular region centered on the depot. If this strategy, which is capable of exploring a larger area than the gradient-based method, is unsuccessful as well, the random walk algorithm is used as a backup.

### Cooperative Transport

This section explains the overview of cooperative tasks performed by multiple robots for transporting resources from the cluster to the depot. By limiting the working area of a single robot, the detrimental impacts of interference in congested regions like the depot may be avoided, and foraging performance can be improved. However, this sub-section does not include the transporting works of a large volume of resources by a number of robots, but it does focus on cooperative strategies that make the process of delivering an resources to the depot more efficient by using a group of robots.

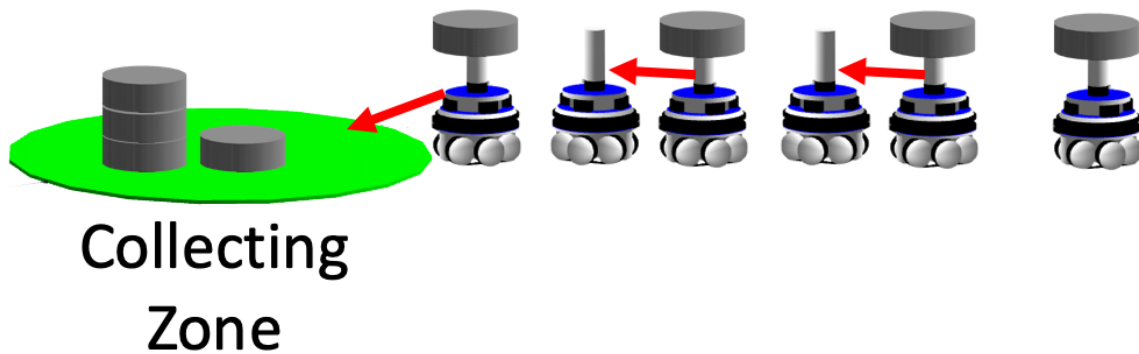


Figure 5: The example of the bucket bridge made by robots which introduced in [1]. Each robot transfer the goods to next robot on the bucket bridge made by robots. The last robot put the goods to the collecting zone.

Bucket brigade, in which things are handed from one robot to another until they reach the depot, is a well-studied cooperative transport system in swarm robotics. Drogoul et al. [1] modified the idea which is described in [51] by giving other abilities to the robots; Robots can inform the status using the light signal while carrying resources, and robots attracted resources-carrying robots while searching the resources for transferring the resources to resources carrying robots. Based on the additional abilities, robots can construct the chain from resources cluster to the depot, where the resources can be transport by the chain. They discovered that when interference is eliminated, foraging performance increases dramatically. In addition, Ostergaard [58] proposed a method that dropping the resource and inviting other robots to pick up and deliver to depot. The robot that discovered the resource requests that give resources to another robot to reduce interference of movement between robots.

An region division allocation is a process that similar to bucket brigade in which robots are allocated to separate work region. Goldberg and Matari [59] proposed a robotic system in which robots are separated into several robot groups that work in other region: one of the robot group works outside of the depot and transports collected resources to the depot's boundary, while another robot group transports collected resources from the depot's boundary to the depot. The area is separated into several of work zones that correspond to the group of robots [60]. Every robot has a working area that it explores for resources, with the exception of the robot in the last working zone, which takes the resource it finds and deposits it immediately in the storage facility. When the robot has gathered the necessary resources, it will be deposited at the boundary of the subsequent working area leading toward the storage facility. They demonstrated how robot invasions of contiguous regions, caused by either localization mistakes or the process for depositing captured objects, are a main cause of robot interference and establish the presence of a threshold number of foraging robots over which working effectiveness suffers.

In the scenario presented by Pini et al. [61], region division occurs dynamically as a consequence of work allocation among robots. Resources to be collected are concentrated in a single spot, and after transferring a resource, robots go back to the location where they discovered it; but, owing to localization errors or other factors, the system cannot ensure that robots can rapidly return to the resource source. Instead of returning to the depot, each robot transfers acquired materials for a certain distance, relying on other robots to finish the mission. Due to the fact that the number of localization mistakes increases as the robot moves, this process improves the resource-finding success rate for robotics. Based on an assessment of a cost function, each robot independently determines the distance to go. The cost functions calculate costs in accordance with the total cost of the job to transfer resources to the depot. Arkin et al. [62] proposed a transport system in which many robots may transport one resource at the same time; this mechanism can speed up the process of transferring the resource to the depot, allowing collaborating robots to complete the task more quickly. In addition, robots carrying resources employ an explicit communication system to announce their status so that other robots are drawn to them and assist them in bringing the resource to the depot.

### **Other Cooperation Strategies**

In addition to cooperative transportation and path formation, the literature also offers numerous instances of foraging cooperation mechanisms. Rybski et al. [63] used two communication methods in which robots activate a light sensor that attracts other robots; Both approaches aim to draw robots to the resource. In the first method, known as reactionary communication, a robot that is in the process of collecting a resource signals its status with the emission of light; as in the second method, known as deliberate communication, robots that detect a resource but is unable to collect it because they are already carrying another resource stop near the detected resource and signal its presence for a fixed time duration before returning to the detachment. These two approaches are used to communicate with each other. According to [64], a robot that is aware of the location of a set of resources may recruit another robot upon returning to the depot. The recruited robot follows the recruiter from the depot to the site of



the resources, thus decreasing the time spent seeking for the resources. The usage of direct communication in [65] facilitates access to the depot. Robots convey their proximity to the depot via a light signal that may be detected by nearby robots, hence increasing the depot's visibility. A message forwarding mechanism is constructed in [59] in order to eliminate the high level of interrobot interference that happens in the depot area. Using a distributed mechanism, resource-carrying robots advertise their status with specific signals and avoid concurrently reaching the depot area.

### Shared Memory between Swarm Robots

All robots that are part of a swarm robotics system are able to write and read information on a shared memory-equipped medium. Broadcast transmission is conceptually analogous to the method discussed above. Each robot in the swarm is able to communicate information with the other robots in the swarm, resulting in all robots having access to shared information. However, such systems may be useful for investigating the influence of shared information on foraging efficiency and developing recommendations for further communication techniques. In concept, these approaches have challenges with scalability and individual robot simplicity. Often, the most challenging component of a foraging operation is locating the several points of interest in the environment, such as the varied locations of supplies to collect and the cache. Due to the lack of a global positioning system and/or inaccuracies in estimating relative displacements, it is conceivable for a robot to forget how to return to a previously visited area of interest. Even if the robot has been designed to recall the path, this may occur. In other words, each robot contributes its imperfect knowledge to construct a shared map of the arena that, although being partial and flawed, is nonetheless superior than the representations each robot can construct alone [48, 49]. With a shared memory, robots are able to transmit their recent experience (such as the path they traveled to reach a certain spot) and aid other robots in discovering points of interest. The trail information possessed by foraging robots may be utilized not just to follow the same paths to the same destinations, but also to avoid pathways that are used to reach other target places that are being searched [50].

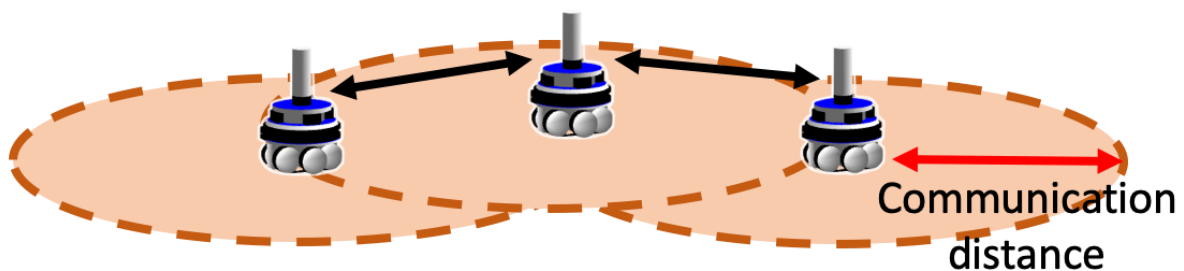


Figure 6: Communication topology model defined by the distance between robots: bold arrows represent the information flow, large circles are communication range.

### **Direct Communication between Swarm Robots**

Direct communication refers to a procedure in which robots can communicate directly with one another, frequently explicitly conveying data to express a certain condition. Typically, information may be communicated between adjacent robots using the notion of local communication, and these robots can then respond on the information received by altering swarm robots behavior to increase the performance of the foraging. Direct communication between robots can be reduce the effects of interference in the congested places like the collecting zone [59], to make it easier to locate a point of interest like a resource source [63, 64] or the depot [65], or to implement cooperative resource transport mechanisms [1, 58, 62]. Simple sensing of the relative location of neighboring robots or even touch sensing may be used to communicate.

### **Communication through the Environment**

Pheromones are chemical compounds generated by ants and other social animals with the purpose of marking their environment. Swarm members use pheromone to establish an indirect communication system that exploits the environment as a conduit for conveying information. Typically, pheromones can be recognized from short distances, and pheromone-mediated communication may assist swarm members overcome their limited sensing and communication capabilities. Similar to shared memory processes [51–53], pheromones may be used in foraging activities to create trails that help in identifying places of interest (such as rich food sources) in the environment. In robotic systems, if robots lack navigational skills that enable them to quickly identify the depot, the environment may be marked with a gradient field whose intensity increases as it approaches the depot in order to optimize the return route to the depot. Since the strength of pheromones in the environment is often dynamic and influenced by robot activity, static gradient fields cannot be considered a kind of pheromone transmission. To lead robots with limited sensory and navigation abilities, both of these strategies make tiny modifications to the surroundings. A method theoretically similar to pheromone communication is implemented when specific robots in a swarm take on the role of environment markers by assuming a fixed position and providing nearby foraging robots with an indication of the proximity to the depot and resource sources. This strategy employs static robots to store and transmit to foraging robots a value corresponding to the current pheromone information at their location. For instance, in [61], the pheromone information held by beacon robots varies over time depending on short-distance communications with foraging robots, and so follows the swarm dynamics similarly to physical pheromone released by foraging ants; In [54], beacon emitters indicate their distance from the depot, enabling robots transporting a resource to simply return to the depot by following a chain of beacon emitters. In certain research [59, 60], robots are allocated to distinct sections of the arena and collaborate by taking up and releasing supplies at these regions' borders. Using the resources that are being gathered, another method of communication across the environment may be established, such as when more than one robot transports a resource from its source to its [61]. For instance, when many robots are involved in moving a material from its source to its depot.

### **A Performance Measurement of Foraging Performance**

There has a two major measurement. 1) the required time to gather the fixed number of resources [51, 59, 60, 63]; and 2) the quantity of resources gathered in a given time as a function of swarm size [48, 50, 56, 58, 60, 61, 65]. In several researches, the cost of foraging robotics system (such as the energy consumption by robots) is measured, for instance by estimating the distance traveled by robots [59, 62]. Krieger et al. [64] provided a different weight to the distance traveled based on the notion that transporting a resource consumes more energy. Krieger gave a varied weight to the distance covered depending on whether the robot is carrying resources, based on the notion that carrying resources requires more energy. Special focus is given to the issue of inter-robot interference, which happens when many robots inhabit a confined location. Hoff et al. [61] added communication model between swarm robots in the cost measurement. This phenomenon may be quantified using the number of robot collisions [59] or the time spent in all instances when a robot seeks to complete a job but is impeded by another robot [64].

### **A Modeling of the Foraging Performance**

The majority of existing studies use either actual robot experiments or computer simulations to analyze a swarm's performance in the foraging task, whereas mathematical modeling is seldom used owing to its inherent difficulties. There are, however, some significant exceptions. For example, Rongier [66] shows how much foraging problem are similar to the scenarios described. Using stochastic Petri nets and Markov chain analysis, it is feasible to represent [51] Using these modeling tools, it is feasible to forecast performance measures such as the chance of delivering all resources put in the environment to the depot within a certain length of time using this model. Using a system of partial differential equations, reference [52] models a foraging scenario based on pheromones. These equations provide the estimation of the spatial density of robots in the search state and robots carrying a resource, as well as the quantification of the flow of resources to the depot based on the rate of transitions to the search state. Lerman and Galstyan developed a mathematical model that investigates the impact of inter-robot interference. The model revealed that there exists, in a non-cooperative system, an optimum number of robots that reduces the time necessary to convey the arena's supplies to the depot. Above this ideal number, the work provided by more robots exceeds their interference with one another. Lerman and et al. mentioned their work in the [67]. As a direct result, much work has been devoted to the development of collaborative interference-reduction systems.

### **3.3 Object Sorting and Clustering**

The phrase "object clustering" refers to operations that need dispersed objects to be gathered in a certain spot. In contrast to the foraging job, the item clustering task does not have a precise goal position for the gathered items; rather, the purpose of this study is to cluster the objects together in close vicinity. When there are several sorts of resources and clusters must be constructed for each type, I refer to this process

as sorting because the items are grouped based on their type. In this circumstance, the job is referred to as "sorting" since the items are grouped by type.

Similar to the foraging challenge, many item clustering situations imply that robots cannot identify distant resources. As a result of this shortcoming, the default behavior while searching for resources is a random pattern of wandering. Either the randomness of the robot's movement is directly programmed into the robot's controller, or it is the result of robots altering their trajectory in reaction to the presence of obstacles. As a result of robots transporting materials traveling at unpredictably fast speeds, clusters form at random locations. This happens when a robot lacks localization equipment and there are no designated delivery regions. Under clustering tasks, a robot is often characterized by short-distance sensing and short-term memorization: in exceptional instances, robots are able to identify resources at zero distance (such as when the robot touches the resource), but they cannot recall past resource interactions. In light of these constraints, each robot's choice on whether to gather or dump resources at any given time or place can only be determined (deterministically or using a probability factor) based on the limited representation of the environment that its capabilities permit. In order for the robots to perform correctly, this choice must be taken. In most cases, the algorithms that execute how decisions are made are based on intuitive considerations; but, in certain circumstances, artificial intelligence is used to determine the choice. It is challenging for interactions between robots operating in the same area to improve task execution performance when they are grouped together. This is due to the possibility of robots interfering with one another. It is conceivable that increasing the number of robots utilized for a certain task will not result in a linear increase in the system's overall performance. This is also true if more robots are utilized to complete the work. As a result, the vast bulk of research on clustering does not investigate how robots interact with one another. There is an exception, which is represented by situations in which robots have localization capabilities that enable them to form clusters in a "smarter" manner: in this case, coordination between robots becomes crucial, and communication mechanisms such as indirect communication [68] or direct communication [69] play a role in the global clustering dynamics. There are exceptions to this rule when robots have localization skills that enable them to form clusters.

The most of published research on clustering and sorting tasks, for instance, use robots incapable of localisation. In these research, the clustering of items is the consequence of random environmental interactions. In other research, robots are able to self-localize, and clustering is accomplished by the deployment of a more deterministic strategy.

### **Probabilistic Cluster Formation**

Due to the fact that robots cannot establish their position, can only sense adjacent items from a limited distance, and do not recall the location of previous clusters, it is impossible for there to be a precise place where objects must be grouped to fulfill optimality requirements. Nonetheless, emergent clusters may be produced by using local sensing capabilities, allowing optimality conditions to be met. Nature provides a plethora of intriguing behaviors that may be used as a foundation for the development of

artificial systems for swarm robotics tasks. Various cluster formation methods have been inspired, for instance, by the behavior of ants as they are sorting their brood, a fairly common occurrence in the real world.

Deneubourg and colleagues [70], which was published, is one of the most often referenced articles about this topic. They offered a basic way for automating the robots' implementation of object grouping. Randomly wandering across the arena, the robots retain a short-term recollection of the hurdles and challenges they have previously encountered. The robots' possibility of acquiring a new resource of a certain kind is related to the number of items of the same type that they have encountered in the past; greater quantities of objects of the same type encountered in the past correspond to lower probability values. A robot holding an object, on the other hand, carries the chance of losing it, and this risk rises as the robot progresses through the final phases and comes into touch with a growing number of similar objects. Using computer simulations to illustrate their thesis, the authors illustrated how this basic approach contributes to the formation of groupings of identical items.

In earlier simulation tests, it has been assumed that robots are capable of sensing their near surroundings. According to [71], robots and objects are placed on a two-dimensional grid of square cells, and a robot that can move in any of eight possible directions can detect the presence and type of objects in three of the eight cells surrounding its current position: the cell in the direction of its movement, as well as the cells on its left and right. This allows the robot to go from one cell to the next in any of eight conceivable ways. If the object's kind differs from that of the objects on the robot's left or right, it will be picked up by a robot that is now holding nothing. If the type of the object is the same as the objects to its left or right, the object is rejected.

According to Hartmann [72], an investigation was conducted into a scenario in which robots controlled by neural networks are placed on a two-dimensional grid of square cells and are able to identify the presence and type of objects in each of the eight cells surrounding their current position. The outputs of the neural controller enable robots to move forward and backward, turn left and right, pick up and drop objects, and move concurrently in all directions. Using a genetic algorithm to generate the settings of the neural controller enables robots to achieve outstanding results in grouping single-type and multi-type items, according to the findings of the researchers. The clustering procedure described in [73] is carried out using actual robots, each of which is equipped with a C-shaped front shovel that can concurrently move up to two distinct items. In both cases, robots will randomly alter their movement direction, but in the second scenario, they will release whatever is pushing them before changing their own movement direction. Robots will continue to move in a straight route until they encounter an obstruction or its shovel shows that more than two objects are being pushed. As a result of this essential process, robots are able to combine everything into a single group by combining all of the components.

Maris [74] describes a further set of tests conducted using physical robots. In this situation, a robot is unable to distinguish between objects that need to be grouped and other barriers, such as walls or other robots; as a result, an avoidance mechanism is engaged whenever any form of obstruction is spotted, prompting the robot to avoid the obstruction. Deactivating the frontal infrared sensor prevents robots from detecting the existence of tiny items in front of them; as a consequence, these things are

moved until another barrier in a direction other than the frontal direction is recognized. This is how the movement of clusterable objects is performed. Even though the authors note that slamming an item against a wall leads it to become "lost," their studies always conclude with a tiny amount of ungrouped objects as a consequence of this strategy. This is because the writers place the items at an angle against the wall. For instance, according to Holland and Melhuish [75], robots are able to push goods and identify obstacles. This clustering behavior is achieved by using an algorithm similar to that discovered [73]. Robots can also distinguish between several sorts of things and can simultaneously pull one object with one gripper and move backwards with another gripper. Using these abilities, the authors showed that robots can sort things into two distinct categories, forming an inner cluster of objects of the first kind that is surrounded by objects of the second type. This spatial arrangement is created when robots of the second kind drag items of the same type for a certain distance before releasing them. Using a varied pullback distance for various types of objects enables the object sorting approach described in [76] to be used to more than two separate categories of objects. As seen in Figure 1, the robots dynamically compute the pullback distance values for each type depending on the pace at which the different item kinds come into contact with one another during the clustering activity. This is carried out in order to attain optimal outcomes in terms of both type separation and cluster density. The authors employ an evolutionary technique to determine the parameters of the distance adaptation algorithm that are optimal for their specific conditions.

Vardy [77] employed virtual robots with visual skills to cluster things of many types that were distinguished by color. This enabled Vardy to do clustering on objects of many types that had been segregated by color earlier. Similarly to the studies described in [73], a C-shaped shovel is used to gather and dump objects. The forward and reverse motions are utilized to acquire both the object pickup and object deposit. Counting the amount of pixels of a certain color inside the range of vision of a robot is one approach for detecting an item. If the robot is holding an item of a specific color and the number of pixels of that color surpasses a specified threshold, the robot will release its grasp on the object. This article discusses two separate applications of the method: In the most basic algorithm, robots walk in straight lines when there are no obstacles in their route, and their interactions with things arise from random encounters. In the improved algorithm, however, robots employ their vision system to expedite the clustering activity by determining the direction of a target item to pick up or a cluster where to deposit the object being carried, and then moving in the direction of the target. In the simplest form, robots walk in straight lines when their route is unobstructed, and their interactions with objects are minimal.

Using robots equipped with a sensor capable of detecting the presence of another robot or an object in the line of sight, Gauci and his colleagues [78] estimated the movement direction and trajectory curvature of the robots. Each of a robot's left and right wheels is assigned a specified angular velocity in response to its surroundings. This defines both the robot's journey route and its trajectory's curvature. By squishing items together, it is possible to move them. A mechanism known as artificial evolution is used to obtain the most optimum values for the angular velocity of the wheels for each conceivable value of the sensor input (i.e. an item being detected, a robot being detected, and nothing being detected). It is



noteworthy to note that clustering may be performed (although with lesser efficiency) even if the sensors cannot recognize or discriminate robots from items. In their study, the authors proved that this strategy may be used to cluster objects if the robot sensors have a sufficiently enough range.

### **Deterministic Cluster Formation**

In the case that robots are endowed with the ability to position themselves in their environment, clustering may be accomplished more efficiently by transferring encountered objects to a fixed location, as opposed to relying on probabilistic methods. In order for the system to work successfully, it is vital to have a coordination mechanism in place since all of the robots must agree on a common location to group the goods. While each robot may initially select a random location and begin clustering there, all robots must eventually assemble in a single, shared cluster to guarantee that the job is properly performed. [69] is a way of obtaining agreement amongst robots based on direct communication: robots that come into touch with one other share their current cluster position and determine probabilistically whether or not to transfer to a place that is being utilized by another robot. In lieu of explicit communication, the robots in [68] are able to measure the size of the different clusters being formed and alter their preferred cluster placement based on the observed sizes, giving larger clusters preference.

As a consequence of the usage of a positive feedback mechanism, the clustering dynamics may be described as follows: an action done at a particular location increases the likelihood that the same action will be repeated in the future at the same location. This circumstance is an excellent illustration of stigmergy, which is a phenomenon seen in nature and imitated in artificial systems in which local changes in the environment induced by past actions influence the execution of later actions. During the clustering operation, the branding mechanism can be seen in two processes that are complementary to each other. This process is to add items to the larger cluster (more likely to contain more objects in the future) and to remove the objects from the smaller cluster (this will be done later by the cluster again to increase the possibility of shrinking). Wang and Zhang's implementation [71] found a "critical" size such that clusters less than this size tend to vanish (i.e., the chance that robots remove things from them is higher than the probability of adding new objects), and clusters bigger than this size tend to increase over time.

When robot is endowed with the capacity to self-localize in their environment, clustering may be performed more effectively by moving encountered objects to a predetermined location rather than using probabilistic approaches. This may be accomplished by relocating objects to a preset position. Due to the existence of a high number of robots in this situation, it is vital to design some kind of coordination mechanism, since all of the robots must agree on a single position to cluster the items. Therefore, while each robot may start the clustering process at a point of its choice, all robots must eventually converge to a single, common cluster in order for the job to be accomplished. When robots collide, they transmit their current cluster position and make a probabilistic decision to migrate to a spot already occupied by another robot depending on the information they get. This is how robots gain consensus using the method in the [69]. In [68], there is no overt or explicit communication; rather, robots may measure

the size of the different clusters being created and adjust their preferred cluster placement depending on the observed sizes, giving preference to bigger clusters.

The dynamics of clustering may be seen as the consequence of a process of positive feedback, in which the execution of an action in a particular area increases the probability that the same action will be performed at that location in the future. This is an instance of the natural phenomenon known as stigmergy, which may be duplicated in artificial systems. It is defined by the fact that the execution of future activities is dictated by local changes in the environment caused by previous activity. During the clustering task, stigmergic mechanisms may be identified in two complimentary processes: the addition of items to big clusters (which enhances the probability that further objects will be added in the future) and the removal of objects from small clusters. Both entail the addition of items to bigger groups (which increases the likelihood that those clusters will be shrunk again at a later time). Clusters that are lower than the critical size tend to dissolve over time (i.e., the likelihood of robots removing things from them is higher than the probability of adding new objects), but clusters that are bigger than the crucial size tend to grow over time. Wang and Zhang discovered this essential size for their implementation in [71].

### **A Performance Measurement of Object Clustering**

Performance measures of task that are frequently used to evaluate the efficiency of an algorithm include the number of objects or clusters that are being created [69, 74], the size of the cluster that is the largest [69, 79], the average cluster size [79], and the amount of time needed to finish the task [73].

The completion of a job may be defined as obtaining a situation in which there is one cluster per object type (where this is attainable with an acceptable probability) or where the number and size of clusters satisfy stated conditions. Before the assignment to be regarded complete, both of these requirements must be completed. In [71], a performance measure in which each item type may be allocated a different weight represents the proportion of tasks that were successfully done in the presence of objects of several types. The measure is determined by comparing the size of all clusters to the total number of items of a certain category. In [68], the fraction of a work that has been finished is calculated by considering just the greatest cluster of items in each category. This statistic closely resembles the one discussed before. Vardy et al. [68] employed a different statistic referred to as time-weighted completion to determine the temporal pattern of task progress that happened throughout an experiment. Using multi-type object clustering, the pace at which a task is completed depends not only on how items of the same kind are grouped together, but also on how objects of different types are segregated from one another. These concerns are reflected in metrics that contain a compactness component (which assesses the quality of clusters of identical things) and a separation component (which gauges the quality of space between clusters of identical objects) (that measures the degree to which items of different types are isolated from one another). As stated before, the presence of many robots operating concurrently does not often result in a superlinear gain in performance. Inter-robot interference, on the other hand, is likely to reduce the quantity of productive work that can be accomplished by each robot. As is intuitively understood, the number of such events per robot increases as the number of robots increases, and as a

result, the total number of interactions increases in a manner that is more than linearly proportional to the number of robots present [73].

### **A Modeling of Object Sorting and Clustering**

The work that was done by Martinoliti et al. [79], who developed a probabilistic model and applied it to two different clustering scenarios, is one of the few attempts that have been made to construct a comprehensive mathematical model of the dynamics of clustering. In the approach, a sequence of probabilistic events is used to describe occurrences such as robots entering the detection zone of a cluster, items being added to or removed from clusters, and other events of a similar kind. In the method for controlling the robot, the chance of each event happening is assessed by taking geometric factors into account. The model was able to accurately forecast the dynamics of the cluster, as shown by the model's creators (using metrics such as the average cluster size, the number of clusters, the size of the largest cluster, or the time required to create a single cluster). The problem of object clustering has been subjected to a comprehensive and formal investigation by Kazadi et al. [80]. The authors of the study discovered, through their research, that the ratio between these two probabilities, which is a function of cluster size due to the fact that both probabilities are generally dependent on cluster size and is symbolized by  $g$  in [80], is an essential property of object clustering scenarios that determines the system. This ratio is a function of cluster size because both probabilities are generally dependent on cluster size (expressed as a number of clusters). By making careful adjustments to the  $g$  function in the robot's implementation, it is feasible to obtain the clustering dynamics that are needed in a certain circumstance.

### **3.4 Navigation**

A collective navigation scenario is one in which localization skills with limited knowledge may approach an unknown target with the assistance of other robots. This is done by the robot's collaboration with other robots. Since the focus of this category is on situations in which a single robot must reach its destination while benefitting from the presence of other robots, scenarios in which many robots go to the same destination are not examined.

Given that each robot in a swarm has a limited grasp of its immediate environment, information sharing between robots is essential for a successful navigation strategy in a multirobot environment. This information is often provided in the form of a distance estimate to the target, which may be stated in a variety of forms, including Euclidean distance or hop count. Robots may interact in a variety of ways, including directly across short distances and indirectly via their surroundings. Since a robot cannot always assume the presence of nearby robots with which it can communicate or nearby messages left in the environment by other robots, the majority of robot controllers include a random component that causes the robot to move randomly until it receives information that enables it to optimize the navigation task. This is due to the fact that a robot cannot always assume the presence of other robots with whom it may interact or messages left by other robots in the environment. Once the first data has been acquired, the robot's behavior is typically more predictable, since it can then choose the most efficient path of

travel.

### **Static Routing**

set of landmarks whose location does not fluctuate while the robot is traveling is what is supposed to be interpreted as the route that a robot travels when it is seeking for a given region. Only trials in which the location of such markers in the environment is randomized dependent on the movement of robots in the swarm are considered in this category. Cohen recommended deploying a robot swarm to drive an agent to an indeterminate point in his 1996 article identified in [81] When a robot discovers a target, it instantly stops traveling and starts reporting its finding to the other robots in the vicinity. It will be much easier to reach a bigger area of the arena if the swarm merely replicates this way and utilizes it. The wireless signals provided by robots in [82] form a pheromone gradient, which is akin to the chemical pheromones produced by ant colonies. By following this gradient, any location with at least one robot within communication range may quickly arrive at the destination.

### **Dynamic Routing**

In the previous paragraph, the path of a robot is specified by a series of landmarks represented by robots or markers dropped by robots. Using robots as landmarks has the issue of devoting a significant amount of resources to a single robot; nevertheless, executing a strategy involving indirect communication between robots (communication through the environment) presents practical obstacles. These challenges may be avoided by the use of direct communication and non-static robot placement. The robot's path from its present position to the goal is determined using more advanced algorithms using these approaches. The robots that Sgorbissa et al. [83] equipped with limited sensing and communication capabilities navigated using the following method. When a robot recognizes the position of a target, it alerts other robots in its line of sight so that they may move closer to it. With this technique, chains of robots in line of sight send information on the estimated distance to the target, which is determined either directly (if the target is in line of sight) or by adding the distance to the next robot in the chain to the stated distance to the target. The method is compatible with mobile robots and does not need fixed robots. "Ghost" robots: If a robot seeking a target detects another robot transmitting information on the estimated target distance, and the transmitting robot moves in the environment such that it becomes invisible to the first robot or increases its distance to the target, then the first robot moves toward the previous location of the second robot, as if a "ghost" robot were still transmitting its target distance from the previous location. Ducatelle et al. [84] proposed solving the navigation problem with a dynamic routing algorithm in which the route between a robot and a target location (expressed as a sequence of intermediate nodes) is continuously updated and optimized based on the current configuration of the swarm. This technique is successful in congested areas with low robot density and intermittent communication between robots.

## **A Performance Measurement of Navigation System**

Comparing the time required for a robot to achieve its target [85] to the time required for a non-cooperative approach such as a random walk is typical practice [86]. If I want to assess the performance of a distributed algorithm, you may use non-cooperative methods, but if you want to study the performance of a navigation technique, you can discover the shortest path between two locations. This is corroborated by a number of studies, such as [83, 84] in which the average time required to reach the destination decreased as the number of robots grew. The reduced length of time required to reach the target's position comes at the expense of the swarm potentially using a large quantity of its resources. The cost is determined as the total amount of time devoted to the navigation method of a single robot by all members of the swarm in [86], and simulation tests demonstrate that it exceeds the average. As a result, it may be required to strike a compromise between navigation performance and swarm resource use while developing a distributed navigation algorithm.

### **3.5 Path Formation**

In the area of swarm robotics, "path formation" refers to the method by which robots build a route in the environment that links two locations, hence minimizing the amount of time required to travel between those locations. Because the path is often represented by a fixed or moving chain of robots, this process is also known as chain generation. Path construction is a common occurrence in the foraging habitat. This is due to the fact that robots often exchange places of interest and, as a consequence, gain from exchanging knowledge on how to go there. This section discusses research that focuses on situations involving two distinct objective sites and the necessity for robots to effectively navigate between them.

In past studies, the problem of route construction was addressed primarily via the use of pheromone-based, probabilistic, and evolutionary-based techniques. When communicating robots exchange local messages indicating their present location in reference to the route being built, a system comparable to pheromone communication is established. Extensive study has been conducted on pheromone transmission for the foraging job, where it is utilized to dynamically create optimal routes between the depot and plentiful food sources. This dynamic method allows robots to explore new parts of the environment until both places of interest are identified and an optimal route is built between them. Using probabilistic approaches, a robot chain is generated as a series of random events dictated by robots entering and departing the chain. In evolutionary approaches, robot behavior is modified based on a fitness function that measures the quality of a route between two sites. In [87], for instance, robots are controlled by a basic neural network whose parameters are generated using a genetic algorithm, and the fitness function rewards robots according to the number of times they go between two target sites. An other example of an evolutionary approach is the genetic algorithm.

#### **Stationary Robot Chains**

In order to establish a stationary robot chain, a succession of robots within sensing or communication range of their two neighbors must link two target sites. This causes the robots to operate as a chain. In

order to go from one site to another, a robot without a chain need just follow the same route as stationary robots. During their time spent foraging, Werger and Matari (cite: Werger 1996 Robotic) constructed a network connecting the depot to a food supply. Szymanski et al. [88] used the notion of a virtual pheromone to provide navigation information to a network of networked robots in order to determine the quickest path between two places. Their work was mentioned as "Szymanski2006distributed." After then, the robots in the near neighborhood interact through line-of-sight. The messaging algorithm enables each robot to determine the fastest path and communicate this information to the other robots in the network. The authors give proof that their strategy is effective in demanding conditions, including a labyrinth-like arena.

### **Robot Flows**

When robots routinely go between two goal locations, a positive feedback loop is triggered. As a result, route creation occurs when a robot follows a path, which increases the likelihood that future robots may travel the same (or a similar) route in the subsequent iteration. Numerous studies have been undertaken on foraging, and one of the processes discovered is a positive feedback loop in which several robots walk the same path to the depot or food source. This technique is used by both cooperative memory sharing [48] and pheromone transmission [51,52].

### **3.6 Task allocation**

Task allocation or division of labor refers to the capacity of each robot in a swarm robotics system to dynamically adjust the task it does based on its own local knowledge of the surroundings. With this feature, robotic systems may demonstrate effective work dynamics by altering the proportion of robots engaged in a given job (or not engaged in any activity) in response to the task's current demand or the anticipated reward from task completion. This allows the systems to optimize their efficiency. Although robots have limited sensory capabilities, which prevents them from directly measuring the global status of their surroundings, local interactions among the robots and their environment may be utilized to modify the behavior of individual robots in order to increase the swarm's overall efficiency. This may help robots collaborate more successfully. The great majority of work allocation and assignment systems are either based on thresholds, where robots switch operations when an observable quantity surpasses a predefined amount, or on probabilistic metrics.

#### **Threshold-based methods**

In threshold-based approaches, robots monitor some feature of their environment and adjust their behavior when the quantity exceeds a specific level. The observed amount may be global, in which case it pertains to a global condition of the environment that all robots are able to monitor, or it may be local, in which case it pertains to local environmental factors. It is conceivable that the threshold value will be modified. Due to the fact that homogenous robots use the same control algorithm and threshold value, it is feasible for a large number of robots to switch activities simultaneously when the threshold



is exceeded. This may result in undesirable oscillations in the swarm's dynamics. It is advised that I can use local numbers and/or flexible criteria. If the energy level in the storage facility falls below a certain threshold, robots will begin searching for resources to collect. Alternately, methods of task allocation may be determined based on the amount of time each robot spent in previous searches.

### **Probabilistic methods**

In probabilistic approaches, a robot's decision to change activities at any given time is made at random, based on a probability value that is often modified in response to external inputs. These probabilities are used to forecast the potential outcomes of future occurrences. Though all robots are operated by the same probability value, a large number of robots cannot switch activities simultaneously because each robot's control algorithm has a random component. This inhibits the switching of activities simultaneously. According to the concept that robots should only engage in an activity if there is a high possibility of success, each robot is capable of calculating probability values based on past experience gathered when engaging in a specific activity. These values are equivalent to the threshold values used in threshold-based techniques. One of the other tactics utilized in the foraging scenario is sensing a stimulus in such a manner that an action is more likely to occur when the related stimulus is stronger [89]. Another method that has been used is indirect sensing of the fraction of robots engaged in a particular activity compared to the effort necessary for that tasks [90]. The great majority of presently accessible works use distributed task allocation algorithms to assign tasks to robots that must seek and utilize scattered environment resources. Based on data acquired locally, the algorithms that control robotics allow each robot to estimate a global feature of the environment. Foraging is often included in case studies on the allocation of everyday tasks.

### **Foraging**

In the scenario involving foraging, task allocations are employed to determine when each robot should forage and when it should rest. The literature suggests two primary concepts for regulating robot activity: predicted success, which determines when robots seek resources, and stimulation, which decides when robots look for resources. Parker [91] uses motivating behavior to decide when to start and stop searching in swarm robotics. Impatience and acceptance control robot activity: Achieving a threshold value of impatiencetience causes a robot to start searching while failing to achieve one causes a robot to stop searching. The impatience of idle robots in the arena decreases when other robots in the vicinity signal that they are engaged in the search. On the other hand, the acceptance of searching robots increases, guaranteeing that a significant number of robots do not do the same task simultaneously. According to the [17], Liu used a higher amount of signals to regulate the robot's behaviors. Each robot keeps track of its seeking time threshold, or the maximum amount of time it may spend looking for resources before returning to the storage facility, as well as its rest time threshold (time between foraging activities). The rate of resource collection success, the frequency of robot collisions, and the level of social engagement all have an influence on these two variables, which all change over time (results of

the search activity of other robots). According to the authors, threshold variation is an adaptive mechanism that allows robots to adapt to their surroundings. [Bibliography required] The robots that enter the storage facility emit a pheromone that signifies the success of their search. Utilizing a genetic algorithm, the researchers developed a near-perfect mix of adaption parameters for each robot's time thresholds. When a seeking robot approaches the end of its assigned time without finding any resources, it returns to its nest. Previous successes and failures in the collecting of resources impact the possibility that a robot will leave the depot to hunt for resources while it is in the depot. Specifically, everytime a search operation concludes (successfully or unsuccessfully), this likelihood is raised or lowered proportionately to the number of successive successes or failures. In this way, each robot is able to learn how difficult it is to seek resources in the environment, and the total number of robots foraging across the planet rises according to the quantity of resources accessible to collect. The researchers observed that this strategy outperformed a more conventional alternative in which the probability of quitting the company was held constant. Due to the updating of the probability value based on the outcomes of the foraging, robots with a greater rate of success during the foraging process are less likely to be resting than robots with a lower rate of success. This conclusion is explained by the fact that even minute modifications in the robots' mechanical design might affect their foraging ability [92].

According to the findings of Campo and Dorigo [93], an army of robots that forage for food might swiftly refill their energy reserves. The decision algorithm of each robot uses this equation to evaluate the efficacy of the many possible actions, and then updates its control parameters (which are the probability that it will execute particular actions while foraging) based on the findings. The authors show that robots are capable of dynamically learning the optimal values for their control parameters, even when unforeseen changes in the environment's features result in a performance loss. They think that the decline in performance might be linked to a phenomenon known as the "memory effect," which impairs the capacity to adjust appropriately to new environments. An increase in stimulation leads to an increase in the number of robots actively doing their duties. Resting robots' foraging activity is stimulated by an increase in stimulus, resulting in an increase of food at the depot and a drop in stimulation. Since food depletes over time if no new food is delivered to the depot, a rise in stimulus stimulates foraging activity in resting robots, resulting in an increase in food at the depot and a reduction in stimulation. Using something called an activation threshold, Krieger and Billeter [64] were able to control the robots when their energy levels went below a specific point. When this occurred, resting robots would immediately begin seeking for supplies. Using a heterogeneous strategy in which each robot has its own threshold, it is feasible to simultaneously halt the search of all currently resting robots. Every every robot in the cite-castle2013 job makes a probabilistic determination based on the response threshold parameter. This parameter, which controls the robots' input sensitivity, is defined as follows: Foraging may be broken down into two subtasks: gathering and storing the goods discovered. Brutschy et al. explain [90] which can transports a resource from a source zone to a task interface region, where it waits for another robot engaged in the other sub-task and then transfers the resource to it. Likewise, if no other robots are presently working on the other subtask, a storage robot will wait at the task interface zone. Robots that assess the amount of waiting time and switch probabilistically between two subtasks while waiting at the

task interface may get dynamic task allocation. The amount of time spent waiting while doing each of the two subtasks is included into the probability value calculation. This data is merged with the amount of time currently spent waiting. Using an autonomous design process that is based on grammatical evolution, a swarm of robots may be able to dynamically divide into two specialized groups. This would enable them to take advantage of an environment feature that supports division of work.

### **Other Tasks Allocations**

Agassounon et al. [94] developed a mechanism for assigning tasks. In this strategy, robots gather resources that have been randomly distributed into a single cluster. The amount of resources that are dispersed across the environment reduces with time, meaning that the likelihood of each robot finding resources to gather (and, therefore, their job efficiency) decreases over time. If, after a particular period of time, a robot's search activity has not yielded any resources, the tasks allocation system will instruct it to cease its search and shift it to a place where it may rest. This method increases the swarm's efficiency by reducing the number of robots doing unnecessary tasks, while ensuring that the main purpose is fulfilled (clustered all resources). In [95], robots are tasked with discovering and eating one of two randomly placed types of resources. The purpose of the swarm is to distribute robots among resource categories according to the total amount of resources available in each category. Robots outfitted with camera vision systems are capable of identifying around resources and other robots. The authors' distributed control technique for this situation requires the robots to periodically analyze their surroundings and probabilistically modify the current task (i.e. change the sort of resources to consume). According to simulation test findings, the robot swarm is capable of adjusting the percentage of robots assigned to each task to match the quantity of each kind of resource available in the environment.

### **A Performance Measurement of Task Allocations systems**

The behavior of swarm robots will be guided by work allocation algorithms based on their anticipated utility. Assume that robots use energy at a rate proportionate to the jobs they are doing, that completing tasks generates energy income, and that the goal of the swarm is to maximize the net energy income. Foraging swarm efficiency has been measured by comparing the quantity of resources gathered to the time spent searching for them [95], the net energy income to the amount of environmental energy, and the average time spent by robots retrieving a resource [96]. Castello et al. [89] in order to adapt the usage of the average deviation of the energy level as a measure of performance. [64] employs the lowest energy level recorded during an experiment as a measure of swarm resilience, while [95] adaptive uses the average variation in energy level as an indication of performance. When it comes to tasks that need the cooperation of several individuals, the swarm aims to do them as quickly and efficiently as possible. Counting the number of robots currently working on a certain project may be used to assess resource use. In [94], the distribution efficiency of the swarm is measured by comparing the average cluster size, a metric of job completion, to the average number of active workers. This indicates how well the swarm is able to distribute available resources.

### 3.7 Other tasks for swarm robotics

This section include several other tasks for swarm robotics with some examples. 1) Odor source localization; Finding an odor's source in the environment is the odor source localization issue. 2) Object assembly; A swarm handling the task of constructing buildings from items found in the environment in object assembly and construction challenges. 3) Morphogenesis and self-assembly and ; Morphogenesis is the transformation of organisms into new forms. Self-assembly occurs when autonomous robots can physically join with each other via simple local interactions. The well-known swarm-bots project is a research program that resulted in the creation of s-bots, which are physical robots. Through the use of a gripper and a connecting ring, these s-bots may interact with one another [97].

### 3.8 The central-place foraging algorithm (CPFA)

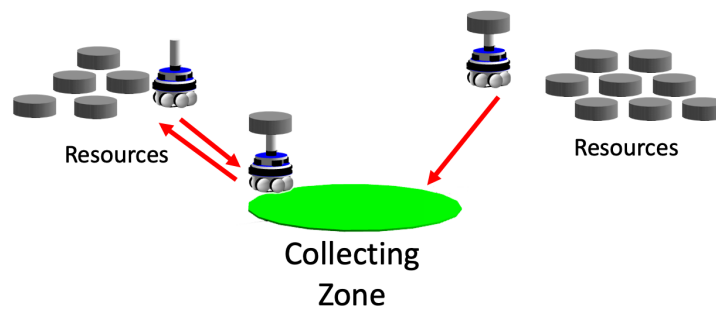


Figure 7: The illustration of the CPFA [2].

A robot swarm foraging algorithm known as a central-place foraging algorithm (CPFA) is one that compensates for the diminishing returns observed during the whole collecting process. The robots' actions are fashioned after those of a particular type of desert seed harvester ants that are limited to brief foraging periods during which not all available prey may be gathered [98], [99]. Therefore, robots are intended to gather as many targets as feasible, but not optimally. Hecker et al. [100] imitate ant behaviors that regulate memory, communication, and locomotion, as well as an evolutionary process that shapes these behaviors into foraging methods that promote error-tolerance, flexibility, and scalability under variable and complicated environmental circumstances. The source for this article is Beyond Pherom (2015). Individual robots are more likely to remember or relay the positions of grouped objects. This enables the robot swarms to locate their targets more effectively. Swarms that have adapted to a particular environmental condition will use foraging strategies that involve the appropriate movement patterns and communication strategies. In prior studies, evolving swarms effectively foraged by maximizing resource collection during a one-hour experimental window. However, resource consumption rates tend to fall dramatically when just a small, unevenly distributed share of remaining resources remains.

A distributed deterministic spiral algorithm (DDSA) is a kind of CPFA that generalizes the spirals search pattern to apply to robot swarms [101] Before the DDSA can specify the interconnecting spirals for a collection of robots, it must first estimate the distance that each robot must travel along each edge of its unique spiral 9. For the purpose of calculating the spiral paths, it is necessary to know the following

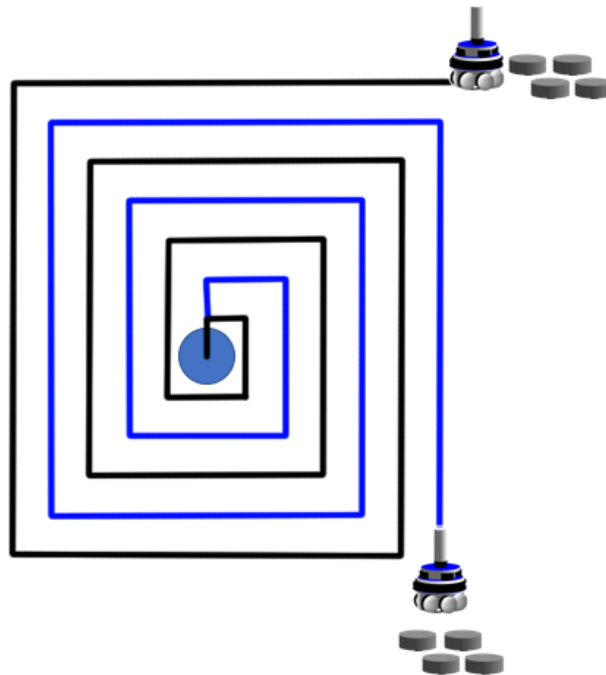


Figure 8: The illustration of DDSA. Utilizing the spiral search pattern, the robots should traverse a continuous plane. They will begin at a central collecting location and move outward. The targets are shown as little gray cylinder in a distribution that is only partially clustered.

information: 1) the total number of robots, so that there is sufficient space for all of them; 2) the target detection range of the robots, so that there is no space between the spirals; 3) how far into the spiral the robot is, given that the spiral grows larger over time; and 4) the index of each robot in a specific order. The ARGoS [102] is a swarm robot simulator which can be used to implement DDAS. ARGoS support for high-fidelity ODE physics engines enables the accurate detection of collisions between robots. 320 new states are created every second by the two-dimensional physical solvers while the simulation is running.

### 3.9 The multiple-place foraging algorithm (MPFA)

In addition to the immune system, the major inspiration for the multiple-place foraging algorithm (MPFA) was the observed behaviors of groups of monkeys and insects. For example, polydomous Argentine ant colonies consist of many depots that occupy hundreds of square meters [24] and [103]. Within the MPFA, a group of robots may be organized to do complex real-world tasks jointly. Collective foraging is one such activity in which robots search for, pick up, and dump things inside a collecting zone. Compared to the central-place foraging algorithm (CPFA), foraging performance decreases as swarm size and search regions increase: more robots result in more interrobot collisions, and larger search regions result in longer travel durations. Lu et al. [26] introduced the multiple-place foraging

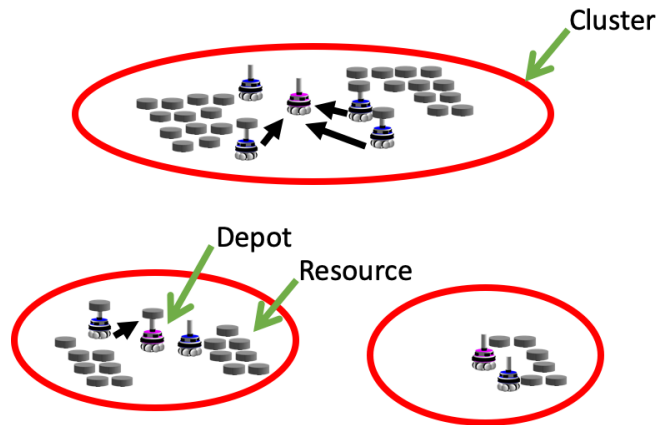


Figure 9: Using the k-means++ clustering method, an example of a dynamically assigned depot. Depots (dark red solid circles) are positioned at the cluster centers. The targets (black squares) are divided into four distinct groups (red ellipses).

algorithm with dynamic depots (also written as  $MPFA_{dynamic}$ ) in an attempt to address these concerns. Initially, the Depots are specialized robots that are placed around the search area. Each depot has the potential to transport a variety of targets. The positions of recently found local targets by robots are utilized to direct the movement of depots to their centers. The geographically dispersed design shortens the time required for robots to shift themselves and reduces the frequency of robot collisions. I describe robot swarms that act like foraging ants using the  $MPFA_{dynamic}$  strategy and a genetic algorithm to enhance their behavior. The ARGoS robot simulator is used to model these robot swarms. Robots using the  $MPFA_{dynamic}$  are faster at locating and collecting targets than those utilizing the CPFA or the static MPFA. The dynamic MPFA performs better than the static MPFA even when the static *depots* are appropriately positioned using global information, and it performs better than the CPFA even when the dynamic *depots* transport targets to a central location. In addition, the  $MPFA_{dynamic}$  grows up more efficiently, therefore the advantage over the CPFA and the static MPFA is even more apparent in expansive regions (50 x 50 meters). When simulated error is incorporated, the performance of all algorithms declines, but the MPFA maintains a performance edge over the other choices. According to this study, dispersed agents that dynamically adapt to the local information in their surroundings provide a more flexible and scalable foundation for swarm robotics than previously thought.

### 3.10 Simulators for Swarm Robotics

The swarm robotic systems research requires a substantial number of physical robots, making it difficult for many research groups to support [77]. Using the currently-being-created computer simulation, it is aimed to visually examine the structures and algorithms. Even though the study of real robots is the final objective of the research, it is often extremely beneficial to do simulations before going on to examine real robots. Simulations are often more efficient, affordable, and user-friendly than their true counterparts. They are also easier to install. In this section, an overview of widely used simulation



systems is provided.

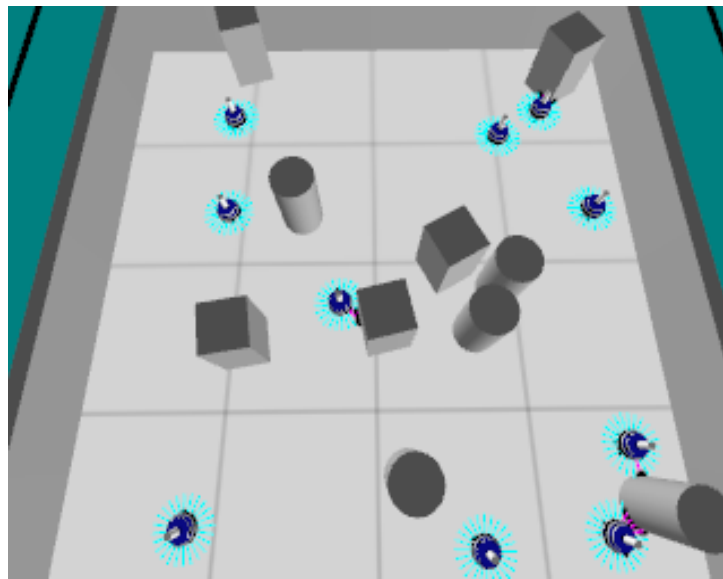


Figure 10: The ARGoS simulator for swarm robotics.

### **ARGoS**

ARGoS [102] is a multi-robot simulator with a multi-physics engine that simulates massive heterogeneous swarm robotics in real time. ARGoS offers a high degree of both flexibility and efficacy as a consequence of its modularity in terms of design, parallelism in execution, and composability of objects. ARGoS is the most effective technique for modeling the physics of hundreds, or even tens of thousands, of robots. In contrast to other simulators, each ARGoS object is referred to as a plug-in entity, and it is both easy to develop and simple to utilize. In this method, various physics engines may be employed in a single experiment, and robots can migrate from one to another in a manner that is imperceptible to the naked sight. According to the research, ARGoS can simulate around 10,000 wheeled robots with their whole dynamics in real time. ARGoS has the extra capacity of being included concurrently in the simulation.

### **Player/stage**

The Player Project [104] produces one of the most well-known simulators and aims to give free software for use in robot and sensor research. The Player Project is employed extensively. The Player project is a robot server that provides researchers with total control and access to the robotic platform, sensors, and actuators. Stage [105] is a scalable simulator that interfaces with Player and can simulate up to one thousand mobile robots in a two-dimensional map simultaneously. In Stage, physics are simulated in a fully kinematic way, and noise is not considered.

## Gazebo

Gazebo [106] is a robot simulator that adds three-dimensional outside scenes to stage. It replaces the simplistic physics engine in Stage with an ODE-based physics engine that generates realistic sensor input. In addition to its native interface, Gazebo offers its users with the more familiar Player interface. This approach allows the controllers designed for stage to be used in Gazebo and vice versa.

## ÜberSim

The Carnegie Mellon University ÜberSim [107] is a simulator developed to provide rapid validation of a program prior to its uploading to real-world robot soccer environments. ÜberSim makes use of the ODE physics engine for more realistic movement and interaction. Even though it was originally designed for soccer robots, the C programming language may be used in the simulator to construct custom robots and sensors, and the program can be sent to the robots over TCP/IP.

### 3.11 Foot-bot

Foot-bot was developed as part of the Swarmanoid Project [108] at the Collège Polytechnique Fdrale de Lausanne in Lausanne, Switzerland. The foot-bot is a differential drive robot with a 13-centimeter-diameter and 28-centimeter-tall circular chassis. The maximum velocity is 30 centimeters per second. Through the use of a docking ring and a gripper, it is possible to connect it to more foot-bots or hand-bots, allowing it to be mechanically modular. Additionally, the base of the foot-bot is equipped with eight infrared sensors. These sensors detect reflected gray tones and serve as proximity sensors all around the robot's spherical chassis. Additionally, the foot-bot is equipped with 13 LEDs, 12 of which are located on the circular ring and one on the top. The foot-bot is also equipped with two cameras, one of which is positioned on the front of the bot and the other of which has a 360-degree field of vision. It has both a short range sensor that can detect distances between 40 and 300 millimeters and a long range sensor that can measure distances greater than 300 millimeters (200 - 1500 mm). The Range and Bearing approach is used to verify that data is correctly sent between foot-bots. The foot-bot is a good robot for swarm applications because to its high degree of sensing skills, mechanical flexibility, interaction with other robots and the environment, small size, and ability to switch batteries on the spot. These characteristics make it an ideal contender for such applications. With the aid of the foot-bot, experiments involving autonomous and decentralized swarms, recharging robots, and self-assembling robots may be conducted. This research employs proximity, range and bearing, positioning, LED, and ground sensors and actuators, respectively.

### 3.12 Job Shop Problem (JSP)

The notion was first designed for the scheduling of work in a job shop, however it now has much more uses than just that one case. In the domains of computer science and operations research, the job-shop problem (JSP) is an optimization challenge [109]. We given  $n$  jobs,  $J_1, J_2, \dots, J_n$ , each of which has a

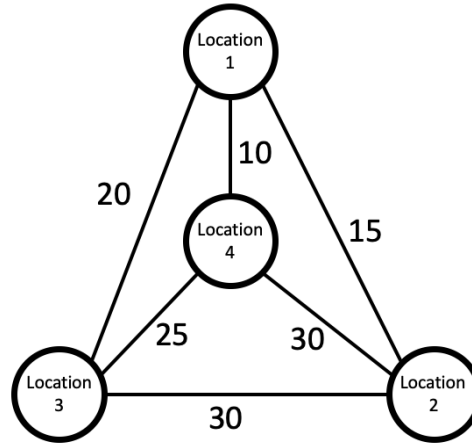


Figure 11: An example of Travelling Salesman Problem

distinct processing time and must be scheduled on  $m$  machines, each of which has a different processing speed, all while striving to lower the makespan, which is the time it takes to complete the full schedule. This is an instance of a general scheduling issue (that is, when all the jobs have finished processing). In the job-shop scheduling system, each work is divided into a series of operations denoted  $O_1, O_2, \dots, O_k$  that must be completed in a certain order. Each task can only do one operation at a time, since each activity requires its own dedicated machine. The laid-back environment of the flexible work shop, where any activity may be performed on any of a variety of machines, is a favorite hobby of many.

The disjunctive graph [110] and [111] is one of the most used models for expressing job-shop scheduling problems. The following is a mathematical formulation of the problem: Let  $J = \{J_1, J_2, \dots, J_n\}$  and  $M = \{M_1, M_2, \dots, M_m\}$  be two sets of jobs and machines, respectively. Each  $M_i$  in the problem represents a machine, and each  $J_i$  represents a task. Let  $\mathcal{X}$  represent the sequence of all tasks allocated to each machine, so that each work is performed by precisely one machine;  $x \in \mathcal{X}$  can describe as a  $n \times m$  matrices in which column  $i$  specifies the tasks that machine  $M_i$  will do in sequential order. As an example, the matrix.

$$x = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}$$

indicates that machine  $M_1$  will do the three tasks  $J_1, J_2, J_3$  in the order  $J_1, J_2, J_3$ , whereas machine  $M_2$  will perform the tasks in the sequence  $J_2, J_3, J_1$ . The cost function  $C$  may be understood as "A processing completion time", and the objective is to discover an assignment of tasks  $x \in \mathcal{X}$  for which  $C(x)$  is minimal.

### 3.13 Travelling Salesman Problems(TSPs)

The travelling salesman problems(TSPs) are the optimization problems that solving the question "Given a list of locations and the distance between each pair of locations, what is the shortest route to visit all locations?". This problem an NP-hard problem in an optimization problem. TSPs can be modelled as an un-directed weighted graph that starts and ends at a specific vertex after each vertex has been visited

exactly once. If there is a solution to the problem, the model is the complete graph, where each pair of vertices is connected by an edge.

## IV A Mobile Conveyor Belt

A conveyor belt is commonly used to transport huge volumes of things swiftly and efficiently. I presented a novel conveyor system called a mobile conveyor line that can configure itself automatically to transport goods to a specified location. I study the reachability of a mobile conveyor line and present an algorithm to verify the reachability of a specified location, as well as the algorithm to offer a configuration for connecting multiple mobile conveyor belts to reach the locations. This system can be suitable for the locations where it is difficult to install a conveyor line, such as disaster zones. One challenge of a mobile conveyor system is how to form the conveyor line by multiple mobile conveyor belts that transports goods from start location to end location. Some locations cannot be reached regardless of how the conveyor belts are connected, due in part to the physical limitations of a mobile conveyor belt (e.g., they cannot tilt at a wide angle) and in partly due to the insufficient number of mobile conveyor belts. In addition, as the number of mobile conveyor belts rises, the number of possible configuration grows exponentially, causing computational difficulty in generating the correct configuration to organize the mobile conveyor lines.

### 4.1 Introduction

Many engaging activities for mobile robots include transporting goods between locations. However, mobile robots are not designed for efficiently transporting a *large* number of goods. As an example, suppose I need to move many valuable items out from a disaster area in a rescue mission. Robots such as those who took part in the DARPA Robotics Challenge have a limited payload, and hence they must make numerous round trips to relocate many items to safer places. The robots with a high payload (e.g., Clearpath's Grizzly robotic utility vehicle) can complete the task in a less time, it is better to deploy conveyor system to help robots move the goods: the conveyor systems are responsible for moving the objects, while the robots are concerned with loading and unloading objects to and from the conveyor belts. Everywhere, conveyor technology is utilized to transport materials along factory assembly lines or as a route of transportation (such as escalators). Therefore, conveyor systems can play a significant role in robotic systems for logistics that emphasize high throughput.

Today, the majority of conveyor belts are intended for stationary use in interior environments (such as conveyor systems in assemble lines.). A small number of conveyor belts are labeled as "portable" and can be transported by employees to outdoor places. The portable conveyors of Miniveyors, as depicted in Fig. 1, can reportedly transport up to 20 tons per hour, making them appropriate for building sites, mining, landscaping, archeology, and catastrophe debris clearance.<sup>1</sup> These portable conveyor systems enable the efficient and rapid transfer of a wide variety of items, making them an ideal companion for robots on rescue missions. However, these portable conveyor system still have to be installed by human, making deployment in severe conditions such as disaster zones challenging. Therefore, I proposed to

---

<sup>1</sup><http://www.miniveyor.com/miniveyor.html>



Figure 12: A mobile conveyor belt.

regard conveyor belts as robots and research conveyor belts with their own movement. I refer to these robots as "mobile conveyor belts." In this research, I offer a mobile conveyor belt, which consists of a conventional conveyor belt attached to a movable platform (see Fig. 12)). The goal of this research is to study how to have a number of mobile conveyor belts independently organize themselves without human assistance.

A challenge part of this mobile conveyor system is how to connect multiple mobile conveyor belts to form a conveyor line that can transfer goods from one location to a given location. Some locations simply cannot be reached regardless of how the conveyor belts are connected, , be partly due in part to the physical limitations of mobile conveyor belts (e.g., they cannot tilt at a wide angle) and in part to a lack of mobile conveyor belts. Moreover, as the number of mobile conveyor belts increases, the number of possible configurations of the mobile conveyor belts grows exponentially, making it computationally difficult to generate the configurations to form a conveyor line without unnecessary mobile conveyor belts. To address these concerns,

- I formulate a complete set of equations to characterize the set of places that a mobile conveyor belt can reach given its physical constraints the hardware;
- I propose a probabilistic algorithm for determining whether it is feasible to link a position to a destination on the flat surface using  $N$  mobile conveyor belts;
- I offer a method for generating a proper configuration for all conveyor belts if a probabilistic algorithm demonstrates the existence of such a configuration.; and
- I present a heuristic methods to increase probability that algorithms in 3D settings will discover a solution.



This section is organized as follows. Section 4.3 defines the automatic configuration problem, and Section 4.4 examines the reachability of a mobile conveyor belt, and Section 4.5 explains how to offer a configuration for conveyor lines. Section 4.6 describes the configuration generation algorithm and presents heuristics to speed up the algorithm. In Section 4.7, I presented an experiment result to evaluate the algorithm. Finally, I conclude in Section 4.8.

## 4.2 A Mobile Conveyor Belt

As seen in Fig. 12, a mobile conveyor belt consists of a movable platform with a conveyor belt. A mobile conveyor belt has three parts: a mobile part, an adjustable part, and a conveyor part. The conveyor part is connected to the adjusting part. The mobile part is a mobile platform that uses wheels to move around for adjusting the location of the mobile conveyor. The adjusting part has two functionality as height-adjustment and pitch adjustment. 1) In Fig. 12, a height adjustment is implemented to lift a conveyor part using x-lift, but it can be implemented in various ways. 2) The pitch adjustment is implemented by a servo motor located at the junction of a adjusting part and a conveyor part. The conveyor slope can be controlled by the servo motor. The conveyor part is a conveyor belt that transports goods in a continuous stream from one location to another.

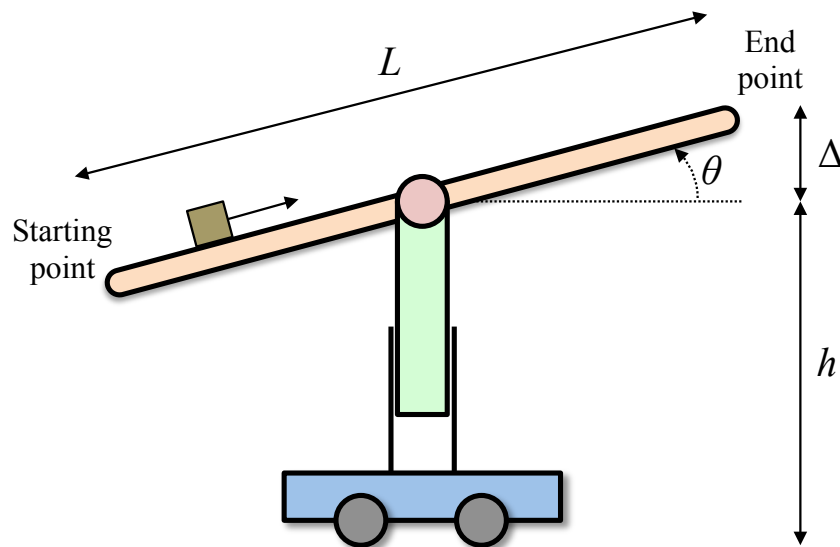


Figure 13: The design of the mobile conveyor belt. The mobile conveyor belt can move objects from starting point to end point.

I have mathematically defined mobile conveyor belts as shown in Fig. 13. Let  $\theta$  represents the *pitch angle* between the conveyor part and the horizontal plane,  $L$  represents the length of the conveyor part, and  $h$  represents the height of the conveyor part's center from the ground.

The mobile conveyor belts are pitch-adjustable with a minimum pitch angle  $\Theta_{\max}$  and a maximum pitch angle  $\Theta_{\max}$  (i.e.,  $-\Theta_{\max} \leq \theta \leq \Theta_{\max}$ ). The mobile conveyor belts have the same maximum pitch angle in both counterclockwise and clockwise directions. Let a *starting point* is the point where the objects enter the conveyor part, and an *end point* is the point where the objects exit the conveyor part.

Depending on the pitch angle, a difference in height between a starting point and an endpoint occurs. In addition, the mobile conveyor belts are height adjustable with a maximum height  $H_{\max}$  and a minimum height  $H_{\min}$ . (i.e.,  $H_{\min} \leq h \leq H_{\max}$ ). However, the length of the conveyor part is not adjusted.

### 4.3 Configurations of Mobile Conveyor Lines

A mobile conveyor line comprises of  $N$  number of mobile conveyor belts interconnected at their terminus to organize the conveyor line. In the mobile conveyor line, each mobile conveyor belt can specifications;  $L$ ,  $H_{\min}$ ,  $H_{\max}$ , and  $\Theta_{\max}$  can be different for each mobile conveyor belt. An *end point* of one conveyor belt and a *starting point* of another conveyor belt are connected together by some mechanisms. If there is no mechanism to physically join two conveyor belts together, two conveyor belts can be connected by stacking one conveyor belt over another, as shown in Fig. ???. In the latter scenario, I assume that objects can be dropped from one conveyor belt to another conveyor belt by some distance. Let  $D_{\min}$  and  $D_{\max}$  be the minimum and maximum dropping distance that objects can withstand, such that  $D_{\min} \leq d \leq D_{\max}$  where  $d$  is the dropping distance. If there are some mechanisms to join conveyor parts of mobile conveyor belts or if objects cannot be dropped, I assume that  $D_{\min} = D_{\max} = 0$ .

Let  $\tau_1, \tau_2, \dots, \tau_N$  be the  $N$  mobile conveyor belts that constitute a mobile conveyor line on the flat surface. Let  $\phi_i$  and  $(x_i, y_i)$  represent the heading of the  $\tau_i$  and the coordinates of the center location of  $\tau_i$ , respectively. Since a mobile conveyor belt cannot rotate without rotation of a mobile part, the heading of the conveyor part is the same as the heading of the mobile part. Let  $\theta_i$  and  $h_i$  be the pitch angle and the height of  $\tau_i$  as depicted in Fig. 13. To simplify the analysis, I assumed all mobile conveyor belts which consists of the same mobile conveyor line are identical. A *configuration* of  $\tau_i$  can be represented by 5-tuple  $(x_i, y_i, \phi_i, h_i, \theta_i)$ . The coordinate of the *starting point* of  $\tau_i$  is  $(x_i^{\text{start}}, y_i^{\text{start}}, z_i^{\text{start}}) = (x_i - (L/2) \cos(\theta_i) \cos(\phi_i), y_i - (L/2) \cos(\theta_i) \sin(\phi_i), h_i - (L/2) \sin(\theta_i))$ , and the coordinate of the *end point* of  $\tau_i$  is  $(x_i^{\text{end}}, y_i^{\text{end}}, z_i^{\text{end}}) = (x_i + (L/2) \cos(\theta_i) \cos(\phi_i), y_i + (L/2) \cos(\theta_i) \sin(\phi_i), h_i + (L/2) \cos(\theta_i) \sin(\theta_i))$ . But the analysis can be easily generalized to the case of mobile conveyor belts as follows.  $L$  be a  $L^i$  that is the length of the conveyor part of  $\tau_i$ ,  $\theta_i$  be the pitch angle between  $-\Theta_{\max}^i$  and  $\Theta_{\max}^i$  where  $\Theta_{\max}^i$  be the maximum pitch angle of  $\tau_i$ .

A goal of a mobile conveyor line is to move objects that enter the scene from an *entry point* go out to *exit point*. Let  $(x^{\text{entry}}, y^{\text{entry}}, z^{\text{entry}})$  is a coordinate of *entry point* and  $(x^{\text{exit}}, y^{\text{exit}}, z^{\text{exit}})$  is a coordinate of *exit point*. An The purpose is to determine  $\langle (x_i, y_i, \phi_i, h_i, \theta_i) \rangle_{i=1..n}$  configurations for  $n$  mobile conveyor belts, where  $1 \leq n \leq N$ , such that the following constraints are satisfied:

- C1)  $x_i^{\text{end}} = x_{i+1}^{\text{start}}$  and  $y_i^{\text{end}} = y_{i+1}^{\text{start}}$  for  $1 \leq i < n$  (i.e., the x-coordinates and y-coordinates of the end point of a belt is equal to that of the starting point of the next belt);
- C2)  $x_1^{\text{start}} = x^{\text{entry}}, y_1^{\text{start}} = y^{\text{entry}}, x_n^{\text{end}} = x^{\text{exit}},$  and  $y_n^{\text{end}} = y^{\text{exit}}$  (i.e., the x-coordinates and y-coordinates of the starting point and the end point of the conveyor line are equal to that of the entry point and the exit point, respectively);
- C3)  $D_{\min} \leq z_i^{\text{end}} - z_{i+1}^{\text{start}} \leq D_{\max}$  for  $1 \leq i < n$  (i.e., the vertical gap between two consecutive belts cannot be too large or too small);

- C4)  $D_{\min} \leq z^{\text{entry}} - z_1^{\text{start}} \leq D_{\max}$  and  $D_{\min} \leq z_n^{\text{end}} - z^{\text{exit}} \leq D_{\max}$  i.e., the vertical gap between the entry point and the conveyor line as well as the vertical gap between the exit point and the conveyor line cannot be too large or too small);
- C5)  $H_{\min} \leq h_i \leq H_{\max}$  and  $-\Theta_{\max} \leq \theta_i \leq \Theta_{\max}$  for  $1 \leq i \leq n$  (i.e., the height and the pitch angle must be within their limits); and
- C6)  $z_i^{\text{start}} \geq 0$  and  $z_i^{\text{end}} \geq 0$  for  $1 \leq i \leq n$  (i.e., the belts cannot go under the flat surface).

where

- $x_i^{\text{start}} = x_i - (L/2) \cos(\phi_i) \cos(\theta_i)$ ;
- $y_i^{\text{start}} = y_i - (L/2) \sin(\phi_i) \cos(\theta_i)$ ;
- $z_i^{\text{start}} = h_i - (L/2) \sin(\theta_i)$ ;
- $x_i^{\text{end}} = x_i + (L/2) \cos(\phi_i) \cos(\theta_i)$ ;
- $y_i^{\text{end}} = y_i + (L/2) \sin(\phi_i) \cos(\theta_i)$ ; and
- $z_i^{\text{end}} = h_i + (L/2) \sin(\theta_i)$ .

#### 4.4 Reachability Analysis

Due to a length of a conveyor part and dropping distance, mobile conveyor belts cannot be able to connect from a starting point to some endpoint. Given a entry point of the object stream, let  $C$  represent a set of *all* connectable end points and  $R$  represent a set of reachable points that satisfy the dropping distance criteria for the points below  $C$ . (i.e.,  $R = \{(x, y, z') : z - D_{\max} \leq z' \leq z - D_{\min}, (x, y, z) \in F\}$ ). If a point is absent from the  $R$ , it indicates that it is not a reachable point. If a point is not a reachable point where a moving conveyor cannot move an object from a starting point to an endpoint and then drop it at that location in order to move it. It implies that it is impossible to form a mobile conveyor line to transport objects from its starting point to the end point and then drop to that point.

As I shall demonstrate in Section 4.6, computing the reachable set of a mobile conveyor belt can assist assess whether a particular conveyor line arrangement is viable. In this section, I described the reachable set of the belt using a complete set of equations.

To discuss the reachable set, I would first analyze the 2D reachable set on the x-z plane with  $y = 0$  and  $\phi = 0$ . This 2D reachable set can be readily extended to 3D by rotating it along the vertical line passing through the starting point of the mobile conveyor belt. This result would show the 3D reachable set of the moving conveyor belt as a torus-shaped area. When the configuration of  $i$ -st mobile conveyor belt is  $(x_i, y_i, \phi_i, h_i, \theta_i)$ , the mobile conveyor belt's starting point is  $(x_i^{\text{start}}, y_i^{\text{start}}, z_i^{\text{start}}) = x_i^{\text{start}} = x_i - L/2 \cos(\theta_i), y_i^{\text{start}} = 0, z_i^{\text{start}} = h_i - L/2 \sin(\theta_i)$  and the end point of the mobile conveyor belt is  $(x_i^{\text{end}}, y_i^{\text{end}}, z_i^{\text{end}}) = x_i^{\text{end}} = x_i + L/2 \cos(\theta_i), y_i^{\text{end}} = 0, z_i^{\text{end}} = h_i + L/2 \sin(\theta_i)$ .

I assumed that the starting points of the mobile conveyor line is fixed at  $(0, 0, z^{\text{start}})$  for easy analysis. When the configuration of 1-st mobile conveyor belt is  $(x_1, y_1, \phi_1, h_1, \theta_1)$ , the end point of the mobile conveyor belt is  $(x_1^{\text{end}}, y_1^{\text{end}}, z_1^{\text{end}}) = x_1^{\text{end}} = L \cos(\theta_1), y_1^{\text{end}} = 0, z_1^{\text{end}} = h_1 + L/2 \sin(\theta_1)$ . The set of

reachable points of 1-st mobile conveyor belt is  $R_1 = \{(x, y, z') : x = x_1^{end}, y = y_1^{end}, z_1^{end} - D_{max} \leq z' \leq z - D_{min}, (x, y, z) \in F\}$ . Therefore, the starting point of 2-st mobile conveyor belt is  $(x_2^{start}, y_2^{start}, z_2^{start}) = x_2^{start} = x_1^{end}, y_i^{start} = 0, z_i^{start} = z_1^{end} - D$  where  $D_{min} \leq D \leq D_{max}$ . If the configuration of 2-st mobile conveyor belt is  $(x_2, y_2, \phi_2, h_2, \theta_2)$ , the end point of 2-st mobile conveyor belt is  $(x_2^{end}, y_2^{end}, z_2^{end}) = x_2^{end} = x_2^{start} + L \cos(\theta_2), y_i^{end} = 0, z_i^{end} = z_2^{start} + L \sin(\theta_2)$ . Moreover, the end point of the  $i$ -st mobile conveyor belts is  $(x_i^{end}, y_i^{end}, z_i^{end}) = x_i^{end} = x_i^{start} + L \cos(\theta_i), y_i^{end} = 0, z_i^{end} = z_i^{start} + L \sin(\theta_i)$ .

When the starting point is determined in the x-z plane, the pitch angle  $\theta$  determines the endpoint of the mobile conveyor belt. In other words, a 2D reachable set of the mobile conveyor belt determine by the pitch angle. The following equation represents a reachable set at the pitch angle  $\theta$ :

$$R'_i(\theta) = \{(x_i^{start}, 0, z_i^{start}) : x = x_i^{start} + L \cos(\theta), z_i^{start} + L \sin(\theta) - D_{max} \leq z \leq z_i^{start} + L \sin(\theta) - D_{min}, z \geq 0\} \quad (1)$$

A reachable set  $R'_i(\theta)$  is part of a vertical line that varies in length and position depending on the value of  $\theta$ .

Let  $\theta \in [\theta_a, \theta_b]$  represent the valid range of  $\theta$  satisfying all physical limitations for the mobile conveyor belt. Then the reachable set is

$$R_i = \bigcup_{\theta_a \leq \theta \leq \theta_b} R'_i(\theta) \quad (2)$$

The valid range of  $\theta$  is determined by the physical constraints' values such as  $H_{min}$ ,  $H_{max}$ ,  $\Theta_{max}$ ,  $L$ , and  $z^{start}$ . In analysis, the sets of different valid range of  $\theta$  can be divided into six cases. In Figures 14 to 19 the six cases show how  $H_{min}$ ,  $H_{max}$ ,  $L$ , and  $\Theta_{max}$  relate to each other. The valid range of  $\theta$  depends on the value of  $z^{start}$  in each instance, as shown in Table 2.

In the table, the symbols are defined in Table 1. In Table 1  $\Delta_{max}$  is a distance between the center of a mobile conveyor belt and the starting point of a mobile conveyor belt,  $\Theta_{upper}$  represents an angle at which the mobile conveyor belt is at the highest point,  $\Theta_{lower}$  represents an angle at which the mobile conveyor belt is at the lowest point, and  $\Theta_{floor}$  is the lower bound of  $\theta$  is limited by the ground. These symbols are defined in Table 1.

To comprehend why the cases in Table 2 are comprehensive, it is necessary to comprehend how the six instances Figures 14 to 19 are categorized. In Case 1 and Case 2,  $H_{min}$  is sufficient to prevent the conveyor belt from contacting the ground. The red lines show a mobile conveyor belt when the height of the mobile conveyor belt is  $H_{max}$  at the lowest pitch angle and maximum pitch angle; Point A is a starting point of a mobile conveyor belt at the minimum pitch angles and Point B is the starting point of the belt at the maximum pitch angles, respectively. Similarly, the blue lines depict a mobile conveyor belt where the height of the mobile conveyor belt is  $H_{min}$  at the maximum or minimum pitch angles; Point C is a starting point of a mobile conveyor belt at the minimum pitch angles and Point D is the starting point of the belt at the maximum pitch angles, respectively., respectively.

In Case 1, the maximum  $\theta$  is constrained by  $H_{max}$ , but the lowest  $\theta$  is unlimited without  $\Theta_{max}$  when  $z^{start}$  exists between Point A and Point B (i.e.,  $z_B \leq z^{start} \leq z_A$  where  $z_A$  is the z-coordinates of Point A and  $z_B$  is the z-coordinates of Point B). As we know, physically there is a max pitch angle  $\Theta_{max}$ , so

Table 1: Definitions of the symbols in Figures 14 to 19 and Table 2.

$$\begin{aligned}\Delta_{\max} &= (L/2) \sin(\Theta_{\max}) \\ \Theta_{\text{lower}} &= \arcsin((H_{\min} - z^{\text{start}})/(L/2)) \\ \Theta_{\text{upper}} &= \arcsin((H_{\max} - z^{\text{start}})/(L/2)) \\ \Theta_{\text{floor}} &= -\arcsin(z^{\text{start}}/L)\end{aligned}$$

Table 2: The valid range of  $\theta$  in all cases in Figures 14 to 19

| Case | Range of $z^{\text{start}}$          | Range of $\theta$  |
|------|--------------------------------------|--|
| 1a   | $z_B \leq z^{\text{start}} \leq z_A$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\text{upper}}$        |
| 1b   | $z_C \leq z^{\text{start}} \leq z_B$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\max}$                |
| 1c   | $z_D \leq z^{\text{start}} \leq z_C$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\max}$         |
| 2a   | $z_C \leq z^{\text{start}} \leq z_A$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\text{upper}}$        |
| 2b   | $z_B \leq z^{\text{start}} \leq z_C$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\text{upper}}$ |
| 2c   | $z_D \leq z^{\text{start}} \leq z_B$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\max}$         |
| 3a   | $z_B \leq z^{\text{start}} \leq z_A$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\text{upper}}$        |
| 3b   | $z_E \leq z^{\text{start}} \leq z_B$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\max}$                |
| 3c   | $z_F \leq z^{\text{start}} \leq z_E$ | $\Theta_{\text{floor}} \leq \theta \leq \Theta_{\max}$         |
| 3d   | $z_O \leq z^{\text{start}} \leq z_F$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\max}$         |
| 4a   | $z_E \leq z^{\text{start}} \leq z_A$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\text{upper}}$        |
| 4b   | $z_B \leq z^{\text{start}} \leq z_E$ | $\Theta_{\text{floor}} \leq \theta \leq \Theta_{\text{upper}}$ |
| 4c   | $z_F \leq z^{\text{start}} \leq z_B$ | $\Theta_{\text{floor}} \leq \theta \leq \Theta_{\max}$         |
| 4d   | $z_O \leq z^{\text{start}} \leq z_F$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\max}$         |
| 5a   | $z_E \leq z^{\text{start}} \leq z_A$ | $-\Theta_{\max} \leq \theta \leq \Theta_{\text{upper}}$        |
| 5b   | $z_F \leq z^{\text{start}} \leq z_E$ | $\Theta_{\text{floor}} \leq \theta \leq \Theta_{\text{upper}}$ |
| 5c   | $z_B \leq z^{\text{start}} \leq z_F$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\text{upper}}$ |
| 5d   | $z_O \leq z^{\text{start}} \leq z_B$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\max}$         |
| 6a   | $z_F \leq z^{\text{start}} \leq z_G$ | $\Theta_{\text{floor}} \leq \theta \leq \Theta_{\text{upper}}$ |
| 6b   | $z_O \leq z^{\text{start}} \leq z_F$ | $\Theta_{\text{lower}} \leq \theta \leq \Theta_{\text{upper}}$ |

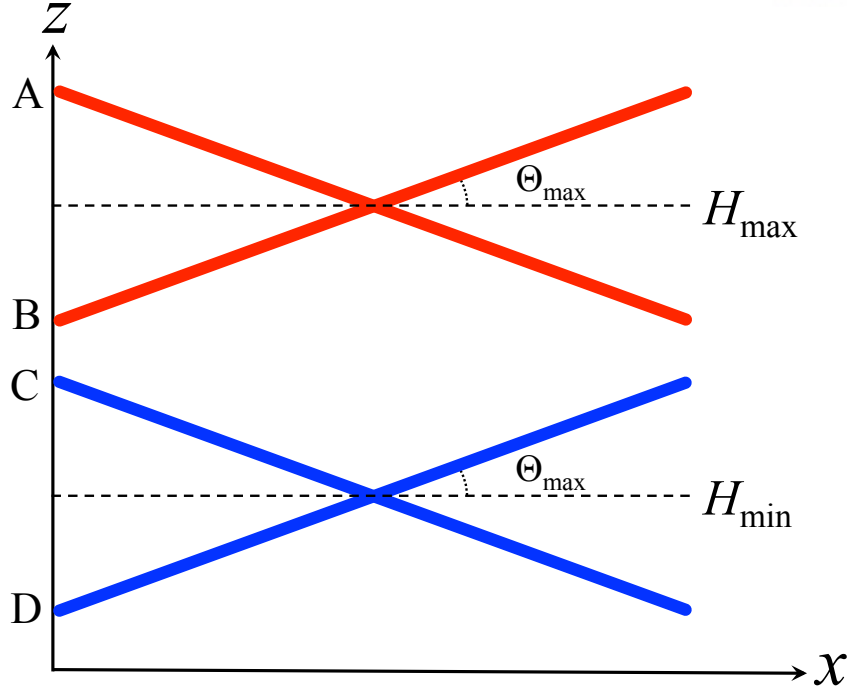


Figure 14: Case 1:  $2\Delta_{\max} \leq H_{\max} - H_{\min}$  and  $\Delta_{\max} \leq H_{\min}$

there is a limitation that a pitch angle of a conveyor part of a mobile conveyor belt cannot go higher than the pitch angle. Consequently, a valid range of  $\theta$  is  $[-\Theta_{\max}, \Theta_{\text{upper}}]$ , where  $\Theta_{\text{upper}} = \arcsin((H_{\max} - z^{\text{start}})/(L/2))$  represents the angle where a mobile conveyor belt is elevated to its maximum height. Similarly, when  $z^{\text{start}}$  is between Points C and D, the lowest  $\theta$  is constrained by  $H_{\min}$ , but the highest  $\theta$  is unconstrained except for  $\Theta_{\max}$ . Hence, a valid range for  $\theta$  is thus  $[\Theta_{\text{lower}}, \Theta_{\max}]$ , where  $\Theta_{\text{lower}} = \arcsin((H_{\min} - z^{\text{start}})/(L/2))$  represent the the angle at which the conveyor belt is at its minimal height. Moreover, if  $z^{\text{start}}$  is exited between Points B and C, the pitch angle is not constrained by the height, hence the acceptable range of  $\theta$  is  $[-\Theta_{\max}, \Theta_{\max}]$ .  $z^{\text{start}}$  cannot be smaller than  $z_D$  or larger than  $z_A$  as the height would exceed the limits. These results are described in Table 2 for in Cases 1a, 1b, and 1c.

In Case 2,  $z^{\text{start}}$  is between Point B and Point C, and when the range of the height is smaller than  $2\Delta_{\max}$ , the pitch angle can no longer be chosen freely; it is constrained by the highest height and minimum height. The valid range of  $\theta$  is  $[\Theta_{\text{lower}}, \Theta_{\text{upper}}]$ , when  $z^{\text{start}}$  is between Point C and Point B (i.e.,  $z_B \leq z^{\text{start}} \leq z_C$  where  $z_B$  is the z-coordinates of Point B and  $z_C$  is the z-coordinates of Point C. Apart from this, all other cases are the same as Case 1. The valid range of  $\theta$  is  $[-\Theta_{\max}, \Theta_{\text{upper}}]$ , when  $z^{\text{start}}$  is between Point A and Point C. The valid range of  $\theta$  is  $[\Theta_{\text{lower}}, \Theta_{\max}]$ , when  $z^{\text{start}}$  is between Point B and Point D (i.e.,  $z_D \leq z^{\text{start}} \leq z_B$  where  $z_D$  is the z-coordinates of Point D and  $z_B$  is the z-coordinates of Point B.) These results are described in Table 2 for Cases 2a, 2b, and 2c. Even if the viable height range is lowered to zero, Case 2 remains valid. Therefore, there are no more cases to examine when  $\Delta_{\max} \leq H_{\min}$ .

However, the mobile conveyor belt can touch the ground when  $\Delta_{\max} \geq H_{\min}$ . Then the minimum



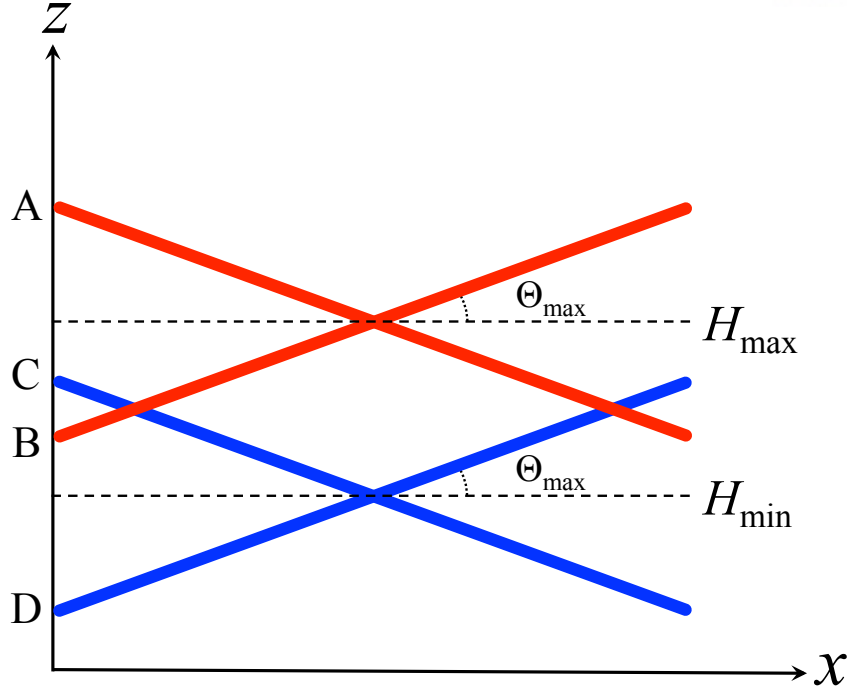


Figure 15: Case 2:  $\Delta_{\max} \leq H_{\min}$  and  
 $0 \leq H_{\max} - H_{\min} \leq 2\Delta_{\max}$

pitch angle depends on two critical points at the intersections of two Cyan lines in Fig. 16.

When the pitch angle is  $-\Theta_{\max}$ , the starting point is Point E and the end point is touching the ground; and when the height is  $H_{\min}$  the starting point is Point F and the end point is touching the ground; and the distance between  $H_{\max}$  and  $H_{\min}$  is  $2\Delta_{\max} \leq H_{\max} - H_{\min}$ . In these cases, there has four different valid range of  $\theta$  based on the  $z^{\text{start}}$  in Cases 3a, Cases 3b, Cases 3c, and Cases 3d in Table 2 ;3a)  $z_B \leq z^{\text{start}} \leq z_A$ , 3b)  $z_E \leq z^{\text{start}} \leq z_B$ , 3c)  $z_F \leq z^{\text{start}} \leq z_E$ , and 3d)  $z_0 \leq z^{\text{start}} \leq z_F$  where  $z_E$  is the z-coordinates of Point E,  $z_F$  is the z-coordinates of Point F, and  $z_0$  is the point where the z coordinate is zero. Due to the minimum height and the ground, there are no restrictions on the lower bound of  $\theta$  for Cases 3a and Cases 3b such as  $z_E \leq z^{\text{start}} \leq z_A$ . Thus the minimum  $\theta$  is  $-\Theta_{\max}$ . The minimum  $\theta$  is  $\Theta_{\text{floor}} = -\arcsin(z^{\text{start}}/L)$  when  $\theta$ 's lower bound is constrained by the ground in case 3c. When Cases 3d, the lower limit of  $\theta$  is once again constrained by the lowest height; hence,  $\Theta_{\text{lower}}$  is the least  $\theta$ .

In Case 4, like Case 3, the mobile conveyor belt can touch the ground. However, difference between Case 4 and Case 3 is the location of Point B. In Case 4, the  $\Delta_{\max} \leq H_{\max} - H_{\min}$  and  $H_{\max} \leq \Delta_{\max}$ , so that  $z_E$  is bigger than  $z_B$ . In Case 4, there has four different valid range of  $\theta$  based on the  $z^{\text{start}}$  in Cases 4a, 4b, 4c, and 4d in Table 2 ;4a)  $z_E \leq z^{\text{start}} \leq z_A$ , 4b)  $z_B \leq z^{\text{start}} \leq z_E$ , 4c)  $z_F \leq z^{\text{start}} \leq z_B$ , and 4d)  $z_0 \leq z^{\text{start}} \leq z_F$ .

Due to the minimum height and the ground, there are no restrictions on the lower bound of  $\theta$  for Cases 4a such as  $z_E \leq z^{\text{start}} \leq z_A$ . Thus the minimum  $\theta$  is  $-\Theta_{\max}$ .

Hence the minimum  $\theta$  is  $\Theta_{\text{floor}} = -\arcsin(z^{\text{start}}/L)$ , the lower bound of  $\theta$  is limited the ground for

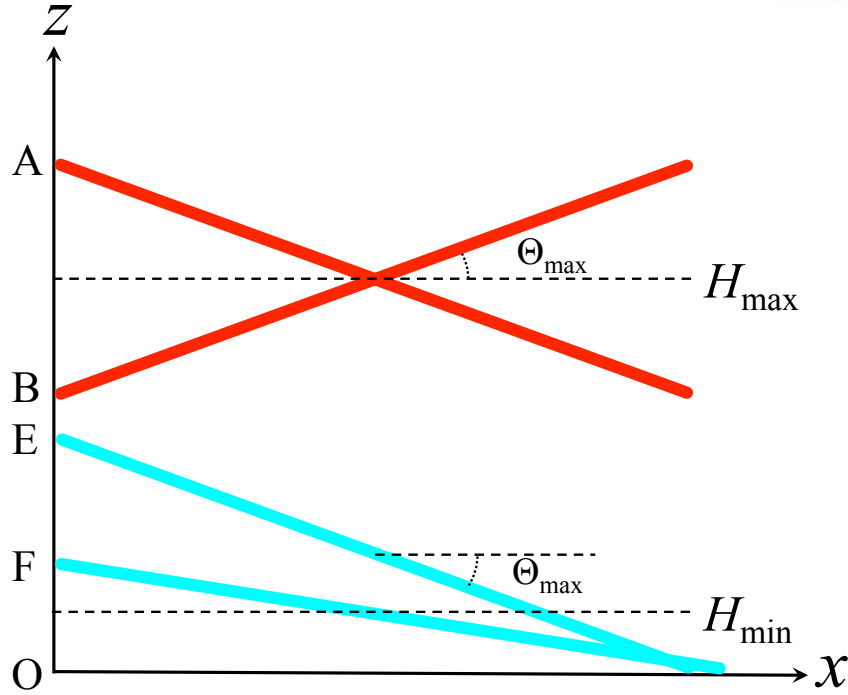


Figure 16: Case 3:  $H_{\min} \leq \Delta_{\max}$  and  
 $2\Delta_{\max} \leq H_{\max} - \Delta_{\max}$

the Cases 4b and Cases 4c.

When the Cases 4d, the lower bound of  $\theta$  is once again constrained by the minimum height; hence,  $\Theta_{\text{lower}}$  is equals the minimum  $\theta$ . All of the preceding are also true in Case5; the only variation being the position of Point B, which results in various combinations of the upper bounds and lower bounds of  $\theta$ .

Case6 is a special case in which  $H_{\max}$  is insufficient to prevent the belt from touching the ground even at the maximum height. There is one essential value in this instance: Point G is the origin when  $h = H_{\max}$  and the end point reaches the ground (the magenta line in Fig.ref:fig:case6). Obviously,  $z^{\text{start}}$  cannot be greater than  $z_G$ , or else the end point would be below the earth. The higher limit of  $\theta$  is always constrained by  $H_{\max}$ ; hence,  $\theta \leq \Theta_{\text{upper}}$ . The lower limit of  $\theta$  is dependent on whether or not  $z^{\text{start}} == z_F$ .

As we can see in Figures 14 to 19, the line  $z = H_{\max}$  gradually approaches the line  $z = H_{\min}$  from Case 3 to Case 6. The Case 6 is the last case that must be considered since it has the condition  $H_{\max} = H_{\min}$  and cannot be less than  $H_{\min}$ .

Given  $H_{\min}, H_{\max}, \Theta_{\max}, L$ , and  $z^{\text{start}}$ , we can determine the lower bound and the upper bound of  $\theta$  from in Table 1 and Fig. 14 to 19. Using  $\theta$  bounds in Eq. 2, we can get the reachable set. Fig. 20 shows following instance of the reachable sets in Case 1 and Case 2 (the orange regions). Due to the constraint  $z \geq 0$  in Eq. 1, I may need to chop a reachable set below the x-axis in order to ignore the points that are below the ground (e.g., Case 1c in Fig. 20c).

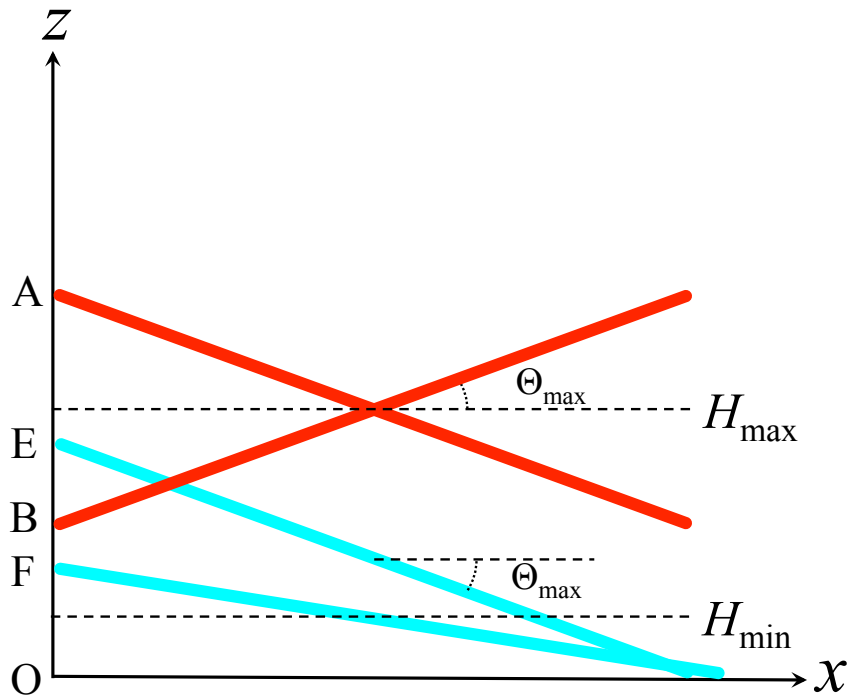


Figure 17: Case 4:  $H_{\min} \leq \Delta_{\max}$  and  
 $2H_{\min} \leq H_{\max} - \Delta_{\max} \leq 2\Delta_{\max}$

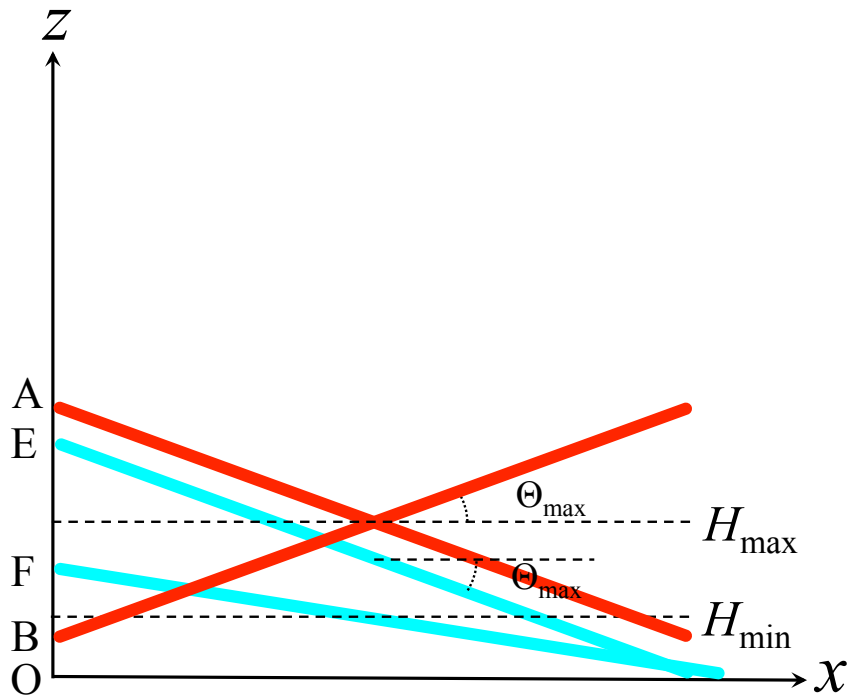


Figure 18: Case 5:  $H_{\min} \leq \Delta_{\max}$  and  
 $0 \leq H_{\max} - \Delta_{\max} \leq 2H_{\min}$

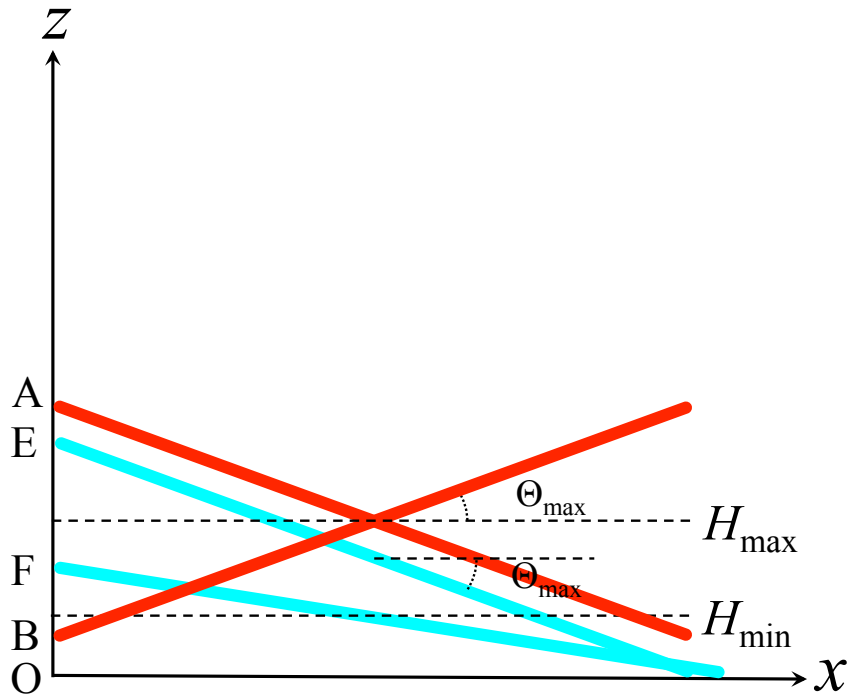
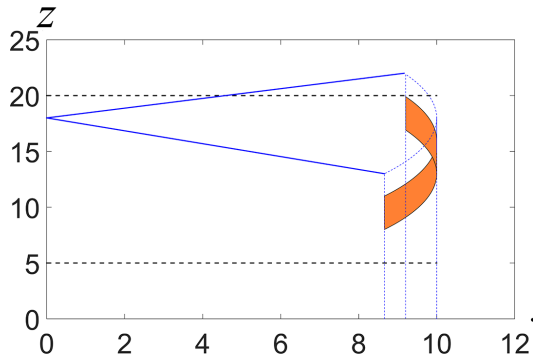
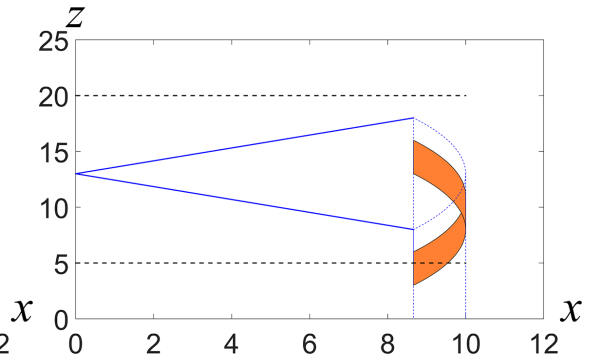


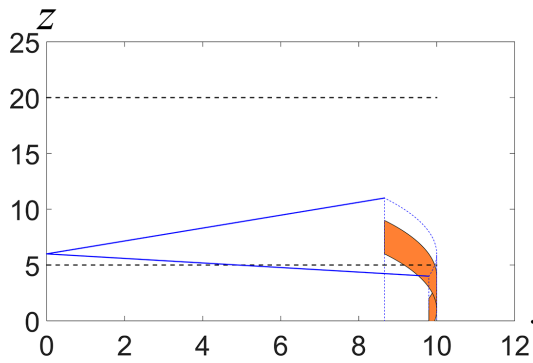
Figure 19: Case 6:  $H_{\min} \leq \Delta_{\max}$  and  
 $H_{\min} \leq H_{\max} \leq \Delta_{\max}$



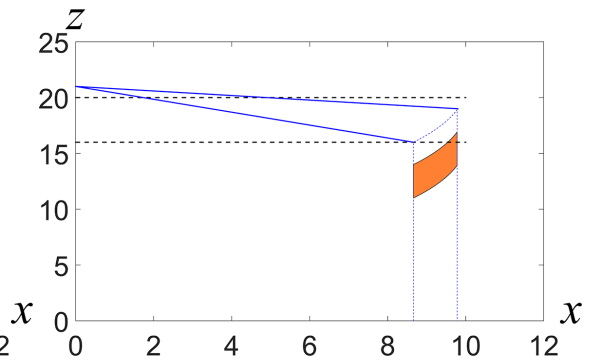
(a) Case 1a:  $z^{\text{start}} = 18$



(b) Case 1b:  $z^{\text{start}} = 13$



(c) Case 1c:  $z^{\text{start}} = 6$



(d) Case 2a:  $z^{\text{start}} = 21$

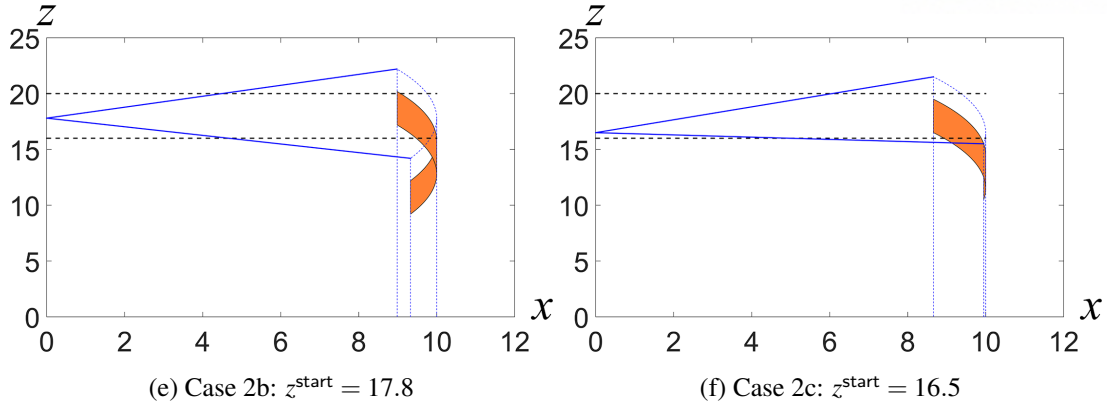


Figure 20: Exemplifications of the reachable sets in two cases; the Case 1 and Case 2. The examples of Case 1 have  $H_{\min} = 5m$  and the examples of Case 2 have  $H_{\min} = 16m$ . The remaining parameters are  $H_{\max} = 20m$ ,  $L = 10m$ ,  $D_{\max} = 5m$ ,  $D_{\min} = 2m$ , and  $\Theta_{\max} = 30^\circ$ .

#### 4.5 Generating a Configuration for Reachable Points

Given a location  $(x, 0, z) \in R$  that is reachable from  $(x^{\text{start}}, 0, z^{\text{start}})$ , I want to compute the conveyor belt's configuration  $(x, 0, 0, h, \theta)$  for reaching  $(x, 0, z)$ . Since the starting point is given, I have  $h = x^{\text{start}} + (L/2) \sin(\theta)$  and  $x = (L/2) \cos(\theta)$ . Therefore, all we need to find  $\theta$  given  $(x, 0, z)$ .

Consider (1) the vertical line segment  $[z + D_{\min}, z + D_{\max}]$  at  $x$ , and (2) the arc of the circle centered at  $(x^{\text{start}}, 0, z^{\text{start}})$  with an angle range of the maximum and minimum  $\theta$ , and a radius  $L$  according to Table 2 and Fig. 14 to 19.

There may be one or two crossings between the arc and line segment; these are the end locations which a mobile conveyer belt may reach while fulfilling the dropping distance limit to reach  $(x, 0, z)$ . Consider the point  $(x, 0, z')$  to be one of the end points. Let  $(x, 0, z')$  be one of the end points. Then the pitch angle  $\theta$  is  $\arcsin((z' - z^{\text{start}})/L)$ , and this gives us the configuration of the conveyor belt to reach  $(x, 0, z)$ .

This calculation can be easily extended to 3D environments with an arbitrary starting point  $(x^{\text{start}}, y^{\text{start}}, z^{\text{start}})$  and any point  $(x, y, z)$  at the donut-shaped reachable set in 3D space. I begin by transforming the issue to the x-z plane and determining the pitch angle  $\theta$  as described previously. Then the configuration is  $(\frac{x+x^{\text{start}}}{2}, \frac{y+y^{\text{start}}}{2}, \arctan(\frac{y-y^{\text{start}}}{x-x^{\text{start}}}), (L/2) \sin(\theta) + z^{\text{start}}, \theta)$

#### 4.6 The Automatic Configuration Algorithm and the Overlapping Effect

Based on the analysis of reachable set in previous section 4.4, I devised an algorithm to produce a configuration for a multiple mobile conveyer belts. Given  $N$  number of mobile conveyer belts, an entry point and an exit point, the algorithm returns a configuration of  $N$  conveyer belts to form a conveyer line that can connect an entry point  $p^{\text{entry}} = (x^{\text{entry}}, y^{\text{entry}}, z^{\text{entry}})$  to an exit point  $p^{\text{exit}} = (x^{\text{exit}}, y^{\text{exit}}, z^{\text{exit}})$ .

---

**Algorithm 1** The automatic configuration algorithm.

---

```

1: procedure FINDCONFIG( $p^{\text{entry}}, p^{\text{exit}}, N, M$ )
2:    $R_1^0 := \{(x^{\text{entry}}, y^{\text{entry}}, z) : z \in [z^{\text{entry}} - D_{\text{max}}, z^{\text{entry}} - D_{\text{min}}]\}$ 
3:    $Q^0 := \{R_1^0\}; n := \text{nil}$ 
4:   for  $i := 1$  to  $N$  do
5:      $U_i := \cup \{R : R \in Q^{i-1}\}; Q^i := \emptyset;$ 
6:     Randomly select  $M$  points  $p_1, \dots, p_M$  in  $U_i$ 
7:     For each  $p_k$ ,
8:       Generate  $R_k^i$  from  $p_k$ ;  $Q^i := Q^i \cup \{R_k^i\}$ 
9:       if there exist  $k^*$  such that  $p^{\text{exit}} \in R_{k^*}^i$  then
10:         $n := i$ ; Break
11:      end if
12:    end for
13:    if  $n \neq \text{nil}$  then    // solution found
14:       $p_n^* := p^{\text{exit}}$ 
15:      for  $i := n$  down to 1 do
16:        Identify  $k_i^*$  and  $R_{k_i^*}^i$  such that  $p_i^* \in R_{k_i^*}^i \in Q^i$ 
17:        Find  $p_{i-1}^*$  which generated  $R_{k_i^*}^i$ 
18:        Compute  $(\theta_i, h_i)$  for the  $i$ 'th belt to reach  $p_i^*$ 
19:        when  $p_{i-1}^*$  is the starting point of the belt.
20:        Compute  $(x_i, y_i, \phi_i)$  using  $p_{i-1}^*$  and  $p_i^*$ .
21:      end for return  $\langle (x_i, y_i, \phi_i, h_i, \theta_i) \rangle_{i=1..n}$ 
22:    else
23:      return "No solution"
24:    end if
25: end procedure

```

---

Algorithm 1 is the pseudo-code of automatic configuration algorithm, and its search space is shown in Fig. 21. The algorithm begins from  $p^{\text{entry}}$  and considers adding a mobile conveyor belt to the conveyor line one by one. The algorithm randomly compute  $M$  reachable sets as a one set that the newly added mobile conveyor belts may reach, until a reachable set contains  $p^{\text{exit}}$  (Line 9–10).

The set of reachable sets following the addition of the  $i$ 'th conveyor belt is denoted by  $Q^i$ . Initializing  $Q^0$  to include a single accessible set, which corresponds to the vertical line segment below  $p^{\text{entry}}$  (Line 2–3), was first step.

For  $i \geq 1$ ,  $Q^i$  is calculated by 1) calculating the union  $U_i$  of all accessible sets in  $Q^{i-1}$  (Line 6), 2) selecting  $M$  random points from  $U_i$  (Line 7), and 3) determining the reachable sets of the selected points according to Section 4.4 and adding them to  $Q^i$  (Lines 8–9). If a reachable set with  $p_{\text{exit}}$  is found by adding  $n$  conveyor belts, the algorithm searches backwards for the sequence of accessible sets that leads to  $p_{\text{exit}}$ , as shown in the red regions of Fig. 21 (Line 14–15).



The starting points of the  $n$  conveyor belts are the points  $p_0^*, p_1^*, \dots, p_{n-1}^*$  where the reachable sets are produced (e.g, the yellow dots in Fig. 21)

Since the algorithm stores all  $p_k$  and  $R_k^i$  in Line 9, no computation is required to calculate  $p_{i-1}^*$  in Line 16. Then I can get the configuration of multiple mobile conveyor belts using  $p_0^*, p_1^*, \dots, p_{n-1}^*$  as specified in Section 4.3 (Line 18–19). I can check that produced configuration satisfies the six constraints in Section 4.3.

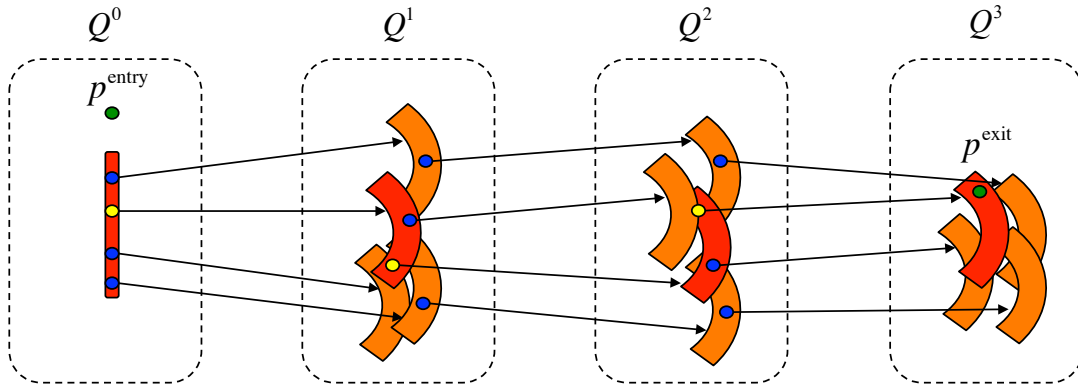


Figure 21: The illustration of the search space of the automatic configuration for  $M = 4$  and  $N \geq 3$ .

The algorithm 1 is functional in both 2D and 3D contexts. In 2D settings, all conveyor belts are oriented on a 2D plane, and, as shown in 21, there are many overlaps between accessible sets. In fact, it holds true because the maximum pitch angles and maximum lowering lengths are often relatively short, restricting the available starting positions of the subsequent conveyor belt to a restricted area. I refer to this phenomenon as the *overlapping effect*. In Line6–7, the method uses this effect by sampling *uniformly* from the union of the set of reachable sets. As we shall explain in Section 4.7, searching with  $M$  sampling points concurrently is superior than  $M$  independent searches, each with a single sample point, since the overlapping area is less likely to be sampled several times owing to the overlapping effect. However, when  $M$  increases, the return diminishes.

In 3D situations, however, the impact of overlap is less pronounced. Due to the additional dimension’s flexibility, the algorithm works badly if I do not direct the search towards the exit point. A solution to this issue is to increase the probability of selecting sample locations that are closer to the exit point. Specifically, in Line6, I choose  $M \times K$  points at random from  $U_i$  for a constant  $K$ . Then, I give each location a weight equal to  $W$  minus the distance from the exit point, where  $W$  is a big constant. Then, I randomly choose  $M$  points from these  $M \times K$  points based on their weights, so that points with a higher weight (i.e., those that are closer to the exit point) have a greater probability of being chosen. As I will demonstrate in the next part, these heuristics may assist in finding a solution in 3D situations.

## 4.7 Experimental Evaluation

I conducted 2 experiments to evaluate the performance of automatic configuration algorithm in 2D and 3D spaces. In addition, I assumed that the ground is flat ground in the both 2D and 3D spaces. In

the 2D experiment, for each  $1 \leq N \leq 20$  number of mobile conveyor belts, I randomly generated 100 *solvable* problems by setting up the parameters using a uniform probability distribution in the Table 3. In Table 3, all units except  $\Theta_{\max}$ 's are in centimeter. Apart from other parameters, I set  $\Theta_{\max}$  to a fixed

Table 3: Range of the parameters for experiments in 2D environments.

| Parameters      | Minimum value | Maximum value |
|-----------------|---------------|---------------|
| $H_{\max}$      | 400           | 600           |
| $H_{\min}$      | 200           | 400           |
| $L$             | 200           | 400           |
| $D_{\max}$      | 30            | 50            |
| $D_{\min}$      | 5             | 10            |
| $\Theta_{\max}$ | $30^\circ$    | $30^\circ$    |

value of  $30^\circ$ . For each set of parameters, the problem generator determined the entry point  $p^{\text{entry}}$  by setting  $x^{\text{entry}} = y^{\text{entry}} = 0$  and choosing  $z^{\text{entry}} \in [H_{\min} - (L/2)\sin(\Theta_{\max}), H_{\max} + (L/2)\sin(\Theta_{\max})]$  at random. Starting from  $p^{\text{entry}}$ , I connect  $N$  number of mobile conveyor belts by randomly choose a dropping distance  $d \in [D_{\min} D_{\max}]$  and a pitch angle  $\theta \in [-\Theta_{\max}, \Theta_{\max}]$ . The algorithm also ensured that the end points of mobile conveyor belts can not go under the ground. Since in the experiment, I mentioned that the grounds are flat, the system did not allow any part of the mobile conveyor belt to go under the ground. The system then selected an exit point  $p^{\text{exit}}$  in the descending distance at the terminus of the last mobile conveyor belt. Then the pair  $(p^{\text{entry}}, p^{\text{exit}})$  is an instance of a problem for which a solution must exist (i.e., a configuration exists to link  $p$  to  $p^{\text{exit}}$ ).

I repeated the algorithm 100 times for each problem using four different values of  $M$ , which is the number of points chosen as 1, 2, 4, and 8. In each repetition, I gave the algorithm 50 milliseconds to discover a solution and restarted it if it failed. If the algorithm cannot discover a solution within the allotted time, it fails to find a solution. I calculated the success rates of 100 executions and presented the data in Fig 22. Note that each data point in Fig. 22 represents the mean of 10000 values, while the error bars represent the 95% confidence intervals. As anticipated in Section 4.6, the algorithm's success rate increases as  $M$  increases. In all circumstances, however, success rates drop as the number of conveyor belts involved in issue creation grows. In particular, the slope of the success rate drop changes based on the magnitude of  $M$  when  $N$  exceeds 10.

Repeatedly, I conducted the same experiment in the three-dimensional space. I produced 100 problems for each  $N \in [1, 20]$  like previously experiment. In contrast, the algorithm was required to choose  $\phi$  when the system want to add a new mobile conveyor belt. To ensure that the departure point is suitably far from the entrance point, the system picked the coordinates  $\phi \in [\phi' - 90^\circ, \phi' + 90^\circ]$ , where  $\phi'$  is the last conveyor belt's heading angle. I ran the algorithm 200 times for 4 distinct values of  $M$ . Half of the iterations employed the heuristics described in Section 4.6 to bias the search toward the exit point, while the other half looked in all directions. I also decreased the execution time limit to 20 mil-

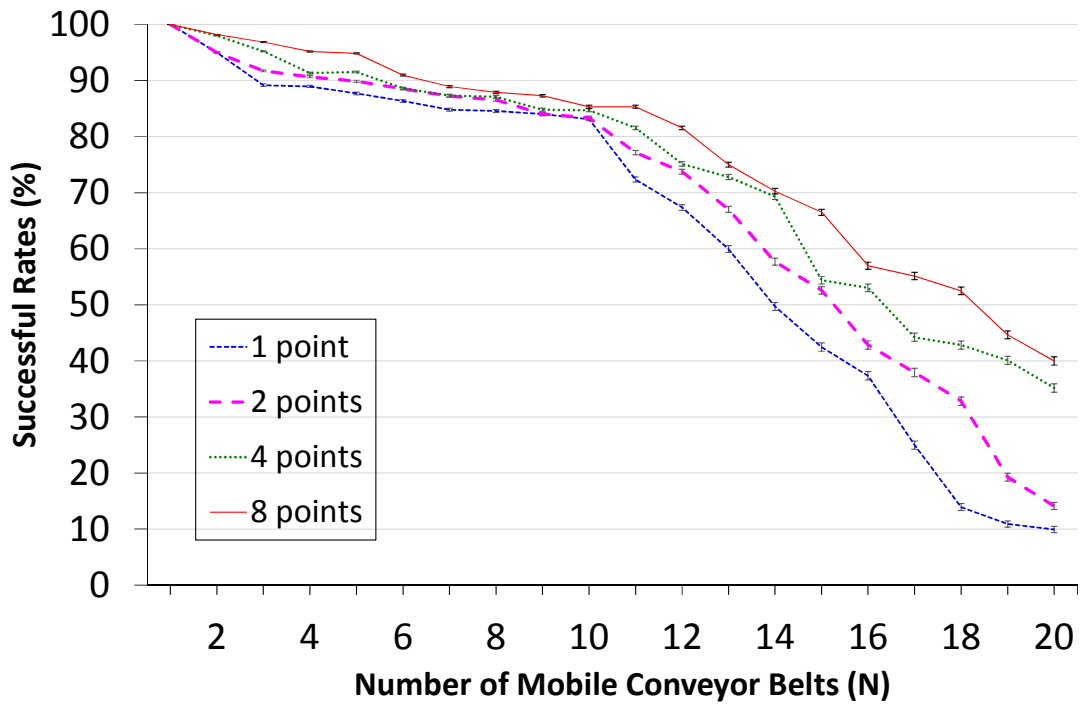


Figure 22: The successful rates of the algorithm in a two-dimensional environment.

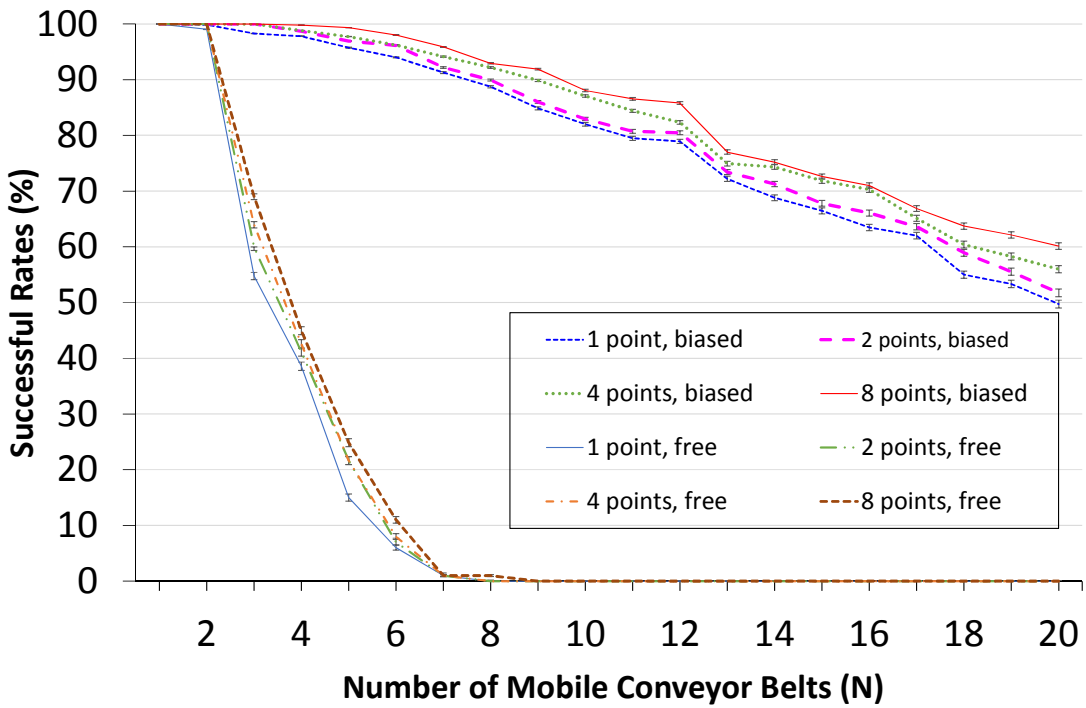


Figure 23: The successful rates of the algorithm in a three-dimensional environment.

liseconds. Fig.ref 23 illustrates the outcomes of this experiment. Without heuristics to guide the search, the algorithm cannot locate a solution when  $N > 10$ . Clearly, the heuristics were of great assistance in finding a solution. In addition, the overlapping effect is modest in the 3D environment, therefore the success rate disparities for various  $M$  values are not as great as in the 2D experiment. I also find that the success rate in 3D is greater than in 2D; however, this is owing to the fact that the distances between the endpoints and the starting point are lower in 3D.

#### 4.8 Summary

This section introduced the concept of mobile conveyor belts, which consist of a typical conveyor belt hooked to a mobile platform. The main objective of the system is to link multiple mobile conveyor belts to build a mobile conveyor line reaching to a specified location. Using a limited number of mobile conveyor belts, I investigated the topic of automated setup of a conveyor line. Conveyor belts are effective at transporting a huge number of goods; hence, they may play a bigger part in robotic systems used in rescue operations and logistical sectors. A comprehensive set of equations describing the accessible set of a movable conveyor belt is among the important findings. Based on the computed reachable set, the algorithm utilized efficient probabilistic approach for generating a configuration of the mobile conveyor belts. In the experiment, I change the number of choosing points to compare the overlapping effects for checking the performance of the configuration algorithm. In the experimental results, the result show the overlapping effect, which claims that the sets of reachable points are frequently overlapped. Moreover, the experiments demonstrated that in 3D environment it is necessary to help the searching process towards the exit point.

## V Dynamic Robot Chains

I consider applying a concept of a mobile conveyor belt to foraging tasks, which can move resources over long distances. I present robots have an ability to link other robots to organize a robot chain via which resources can pass through the link. The a team of robots can cooperate to gather resources. The objective of foraging task is to find and transport resources to a designated central collection zone quickly. According to a previously proposed method as a MPFA with dynamic depots, the foraging performance related to the size of search area and the number of robots. Moreover, the performance of the foraging system decrease as search arena and swarm size get larger: depot robots can travel long-distances to transfer supplies to the central collection zone, and more robots produce greater congestion on their journeys.

I presented a new expansion to multiple-location foraging including the dynamic deployment of the multiple robot chains. Each robot chain links the central collecting zone to a foraging area. The robots in the robot-chain can transfer the objects through the link to reduce the travel distance of the foraging robots. Instead of transporting material by each robot, materials are transmitted straight from foraging areas to the center via robot chains, avoiding congestion in the central collection zone. Additionally, dynamic robot chains may reposition themselves to gain access to resources while avoiding impediments. In the robot simulator ARGoS, I simulate swarms robotic system to measure the performance of the robotic systems. The trials demonstrate that the robot system with dynamic chains outperform the robot system with dynamic depots and experience less congestion. Additionally, dynamic robot chains may reposition themselves to gain access to resources while avoiding impediments. In the robot simulator ARGoS, I simulate the robot swarms. The results demonstrate that robots system with dynamic chains outperform robot system with dynamic depots and have less congestion.



Figure 24: Two mobile conveyor belts form a robot chain.

## 5.1 Introduction

Swarm robotics system has been successfully used to the management of foraging operations, in which the swarm's objective is to seek for items that are dispersed in an arena and return them back to a "central depot" *central depot* [112, 113]. Minerals, water, electricity, and building materials are examples of common resources that are often deposited in clusters at unidentified locations throughout a broad territory. The major goal of existing foraging swarm robotic systems is to offer an efficient decentralized search-and-gather foraging algorithm that insect-like swarm robot systems may utilize to acquire materials [114–116]. The foraging system suggested by Lu et al. was known as "multiple-place foraging systems." This system employs "dynamic robots" as assistive robots, which aid in delivering materials to a central depot [23, 26, 117]. A dynamic depots are mobile robots that works as movable depots but has a limited storage capacity for retaining items. Foraging robots are capable of transporting the resources they collect to nearby dynamic depots. Periodically, these mobile depots will return to the central depot, where they will deposit the materials collected by the foraging robots. Over time, dynamic depots are able to adjust themselves in response to foraging robots' requirements.

Utilizing dynamic depots may substantially improve collection efficiency. However, foraging robotic systems with dynamic depots may have congestion at the central depot when the dynamic depots return to the central depot to unload the gathered resources, especially if there are multiple dynamic depots and foraging robots [26]. The congestion creates bottlenecks that may rapidly impair the system as a whole. I thus propose replacing dynamic depots with dynamic robot chains, which are simply sequences of mobile robots capable of long-distance resource transfer. Two consecutive robots in a robot chain may create a connection that permits the transfer of resources from one robot to the next. As illustrated in Fig 24, one proposed implementation of linkages is based on mobile conveyors [33]. A connection may also be created via various ways (e.g., rope tows). Assume, as shown in Fig. 24, that certain robot chains link to the central depot. A foraging robot may transfer the resource to the last unit in a close chain of robots. The resource will subsequently be transferred to the central storage through the robot chain. The robot chains are dynamic because the robots are movable; as a result, the robot chains may move such that the last robot is closer to uncollected materials clusters.

The using the robot chains in the foraging swarm robotics can reduce a congestion near the central depot because my system ensures that one side of robot chains is connected to the collecting zone as a central depot. In contrast to dynamic depots, there is no need to compete with other robots while traveling to the central depot. Moreover, the robot chain may continually and uninterruptedly transfer resources to the central storage. While congestion of foraging robots may still occur at the last robots of robot chains, it is not concentrated at the central depot and is instead diffused over numerous robot chains. The experimental findings presented in Secection 5.5 indicate that robot chain-based multiple-place swarm foraging systems are more efficient than dynamic depots.

In central-place foraging, scattered resources are collected in a huge arena and transported to a central collection zone [114, 115]. The stochastic central-place foraging algorithm (CPFA) is presented for swarm robotics system, and the details of the implementation are described in the work [116]. The

foraging performance of CPFA can be improved significantly by distributing information on resources location [118]. A distributed deterministic spiral search algorithm (DDSA) [119] demonstrated a search method that ensures a thorough search of the entire area, and each spots is visited once. The simulation demonstrates that the DDSA overcomes the CPFA, but, in the real-world, the CPFA shows slightly better performance than the DDSA [120].

Since resources are located far distance from the central depot impose long travel distances, the multiple-place foraging algorithm (MPFA) [22, 23] modified by observed foraging behavior in the polydomous colonies of Argentine wasps and ants [24, 121] with pider monkeys with multiple sleep site and multiple nests [122]. The MPFA produces better foraging performance and shorter trip lengths compared to the CPFA. The MPFA produces high foraging performance and short travel distance compared to the CPFA. In other work [26], the foraging performance of the MPFA is improved further by introducing the a carrier robot called depot robots MPFA<sub>dynamic</sub>, in which depot robots can deliver resources to the central depot directly [123]. However, resource-transporting robots still need to travel extensive distances. The work in [124] demonstrated a static partitioning strategy that can provide an effective foraging robot swarm. Based on the Grammatical Evolution method, The work in [125] describes that the foraging task is divided into searching and delivering tasks automatically.

In this section, I build on the system of MPFA with dynamic depots in [26] by introducing mobile robot chains to optimize the delivering tasks. In the past, the robot chain has been thought as a sequence of virtually linked robots that helps the localization of other robots by operating a group of robot as the stationary beacons [30] or as a navigation to inform environment's information [31]. However, in these cases, I form a link between robots of the robot chain using UAVs or conveyor belts to transfer the resources [33]. For example, robots can deliver resources between two moving platforms via UAV [32], or resources are transported between two robots by throwing them [39].

## 5.2 Foraging Tasks With Robot Chains

I consider a group of robots cooperating to do foraging tasks which the goal is to gather resources in huge area called an *arena*. There are various types of gathering operations. In these section, I present new foraging system with robot-chain. The foraging system works for the following problem. In Fig. 32, I show one example of a foraging task that I trying to solve by operating a team of robots on these environment. The robots search resources and gather the resource in collecting zone called a central depot. The central depot is a collecting zone in the arena which placed at the arena's center. The goal of the robots performing the foraging task is to gather resources as many as possible in the center within a limited time. The system counts the number of resources that arrived at the central depot within the time limit. It means that the system does not count the number of resources on foraging robots and the links of robot-chains. I assume that the resources are spread into numerous resource clusters whose locations were originally unknown. Robots must investigate the environment in order to discover the location of resource clusters. After detecting the resource clusters, the robots cooperate together to gather the resources and delivery the resources return to the depot. In the arena, there has a number



of obstacles is existed. Obstacles are polygonal regions through which no robot or link cannot pass. Obstacles in arena are hindered the movement and visibility of robots. While exploring the arena for finding the resources, robots need to avoid collision with obstacles and other robots. The positions and shapes of obstacles are initially unknown, but robot sensors can detect them. Each robot processes sensing information for identified obstacles or robots or resources, and shared the processed information to other robots. Through the shared information, robots can know the location of obstacles or resources in advance without facing the obstacles and the resources.

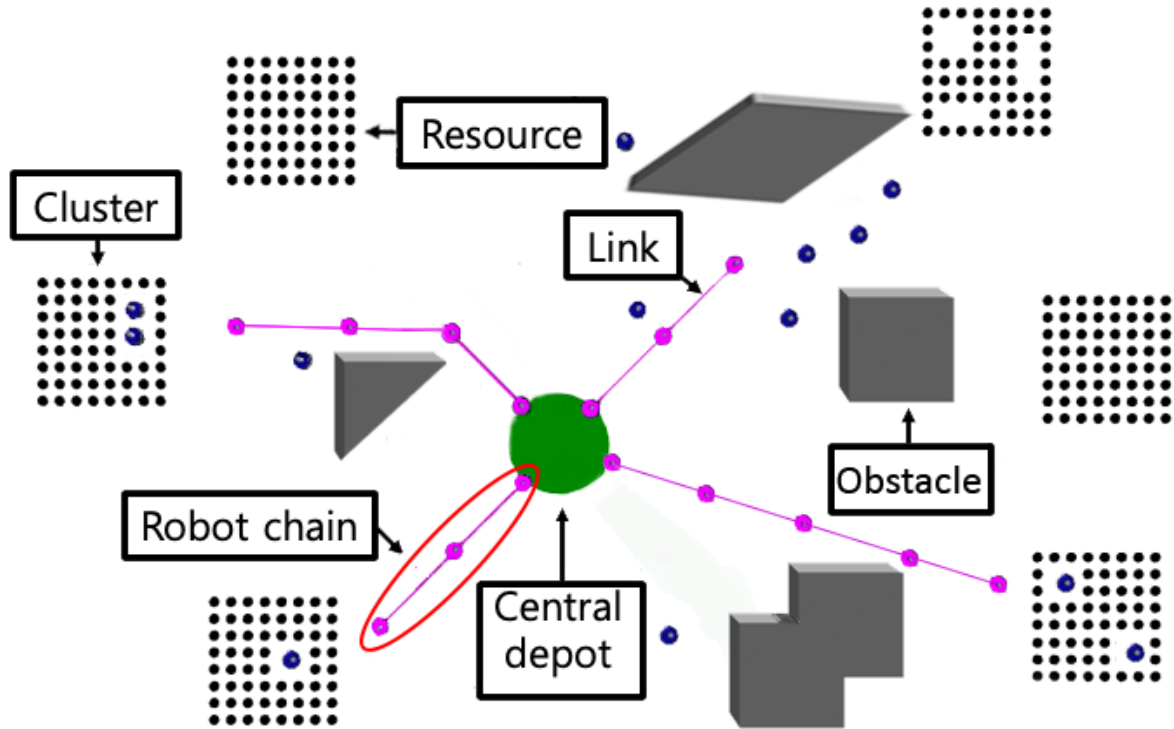


Figure 25: The example of four robot chains deploy in the arena. The magenta dots are robots of the robot-chain and the blue dots are foraging robots.

I have defined a simple robot model as follows. A robot is an omnidirectional mobile robot having a resource-holder with limited store capacity, resource detecting device, lidar sensor, gripper, resource detecting device, wireless communication device, and link-forming component. At any one moment, each robot may be in one of four states: 1) the mobile state, 2) the resource-collecting state, 3) the resource-dumping state, or 4) the robot-chain state. In the mobile state, a robot's lidar and resource sensing device are activated and it may move freely. However, the link forming component and the gripper are disabled.

When a robot in a mobile state reaches a resource cluster, the robot may change a state in which it cannot move while using its gripper to gather a resource from the cluster. Utilizing a gripper, a robot put the resource in the resource holder. While gathering resources, robots are unable to go to other regions. When a robot transfers resources to a resource holder, the robot may alter its state dependent

on the next action, such as a move or other action. After gathering a resource, the robot returns to its mobile form. When a robot's resource holder is not empty, the robot may collect resources from either the central depot or the last robot of a robot chain. A robot in a mobile state with a non-empty resource holder can only enter the resource-dumping state when it reaches the central depot or the last robot in the robot chain. During the stage of resource dumping, the robot can deposit all resources in the resource holder of the last robot in a chain or in the central depot. The status of the robot changes from resource-dumping state to mobility state once it has dumped its resources. At this moment, the robots' resource holders become empty. When a robot in a mobile state is instructed to construct a robot chain with other robots, the robot enters a robot-chain state in which it cannot move, its sensor is deactivated, and the link forming component begins establishing connections with the nearby robots in the robot chain. When the foraging robot is in a mobile condition and encounters a robot in a robot-chain, the foraging robot must avoid a collision. Currently, only the robot that is foraging performs the avoidance movement; neither robot performs a bilateral avoidance movement.

When entering the robot-chain state, the resource-holder should be empty. Before entering the robot-chain stage, the robot must allocate all of its resources to either the last robot in the chain or the central depot. In swarm robotics, there are several communication model types. The robot presented here uses wireless connectivity for transmissions. The robot was outfitted with a wireless connection device for information exchange. Robots may utilize the wireless communication devices to relay information about resource clusters and impediments they meet, regardless of their status. The robot has two functions depending on the tasks it is doing. At any one moment, each robot may assume one of two emphroles: 1) foraging robots or 2) robot-chain robots. The objective of foraging robots is to gather, whereas the purpose of robot-chain robots is to convey materials. Depending on their function, robots may adopt the following states: As a foraging robot, a robot may transition between mobile, resource-gathering, and resource-dumping states. As a robot-chain robot, a robot is capable of switching between mobile and robot-chain states. Note that a robot in a mobile condition is capable of changing its function at any moment.

As described in Section 5.1, there are several mechanical ways for connecting two robots. A connection between robots enables the robot to deliver the resources to another robot. Therefore, the robot chain transports the resource across the links from one end of the robot chain to the central store. All robots in the robot chain must be linked through robot-to-robot connections for the resource to reach the depot. In this paragraph, I explained the link model of the robot chain that I used in the system. I hypothesize that the behavior of these systems can be quantitatively modeled.  $\langle d_{\max}, d_{\min}, C, v \rangle$ , where 1)  $d_{\max}$  is a maximum distances between the robots; 2)  $d_{\min}$  is the minimum distances between the robots 3)  $C$  is the capacity of the link, which is the maximum number of concurrently moving resources; and 4)  $v$  is the speed of moving resources on a link. A distance requirement is necessary for the two robots to establish a connection. Because connections may be built in a number of ways, I determine the minimum and maximum distance between which two robots can establish a connection. Depending on whatever method is used to connect the connection, the minimum distance and maximum distance might be specified accordingly. In addition, the connection mechanism determines the capacity of the link and

the pace at which resources may be sent across it. Due to the fact that the frequency of congestion in the last robot in the robot chain might vary based on the speed and capacity of the link, this was also accounted for in the link model. Each robot-chain robot is a member of a single robot chain, and all robot-chain robots must simultaneously assume robot-chain states to form a robot chain. One end of the robot chain must be located inside the central depot, and the robot at this end is the first robot in the robot chain. When the first robot in a chain of robots receives resources, it is able to dump them at the central depot. The system determines that resources have been gathered when they arrive to the central depot. The robots may place resources in the previous robot's resource-holder at any moment, provided that the space of resource-holder is left. Similarly, the  $i$ 'th robot in a robot chain can transfer the resource in the own holder to a next robot as the  $(i-1)$ 'th robot in the robot chain at any time, provided that 1) the number of resource does not over the link's capacity and 2) the quantity of resources on the link does not over the amount of unoccupied space in the next robot's resource-holder. After resources have been placed on a link, it will finally enter the resource holder of the  $(i-1)$ th robot. The 1th robot deposits the resources it obtains instantly in the central storage. The system-determined resources are gathered when the first robot deposits the resources at the central depot.

As seen in Fig. 26, the lidar is utilized to identified the edges of the other robots and obstacles within lidar's detection range. The robots are able to differentiate the other robots' edges from the obstacles' edges depending on the current position of the other robots as broadcast from those robots. If the position of an edge does not match the edge of other robots, the edge must belong to obstacles. The robot share the edges of obstacles to the other robots via broadcasting information. The resource detection device has its own sensing range and can detect the exact locations of any resource within its range. When the resources existed within the resources detection range, the robot share the information of the resources detection to other robots. Therefore, other robots can use shared information to decide next actions.

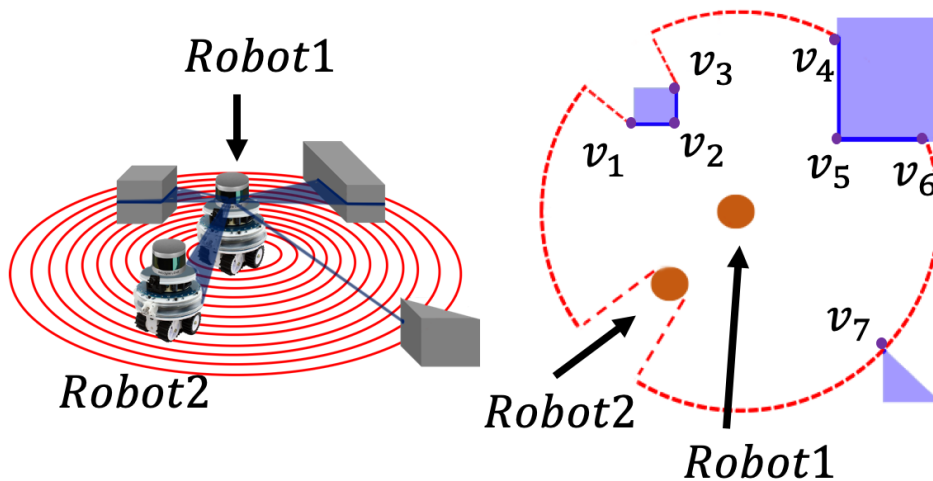


Figure 26: The robot use the lidar to get information. The red dotted line show the lidar's sensing range.

### 5.3 The Controller For Foraging Robots

Each robot is initially given a role, and each robot-chain robot belongs to one of the initial robot chains. The robot-chain robots will first begin forming robot chains that reach various area sites. The goal is to put the last robots in the robot chains far from each other. In the experiments, the system forms 4 robot chains, and this system will try to form robot chains such that the last robots are close to the corners of the arena. Initially, the direction of each robot chain is different by 90 degrees. The way to form the initial robot chain is identical to the method of relocating robot chains, which will be described in Section 5.4.

As I shall see, the locations of a robot chain are updated from time to time in accordance with the conditions described in Section 5.4 and other factors. Foraging robots near the last robot of a robot-chain, on the other hand, will collect resources and place them in the resource-holding robot closest to the last robot of a robotic chain. The controller of the foraging robots is the same as the one in [26], except that our foraging robots have to avoid obstacles. In summary, the controller operates in the following ways. In order to collect resources, a foraging robot constantly remembers the position of the previous resource cluster it visited. After visiting the cluster, robot using gripper to put the resources to regather own resource holder. When gathering resources, the foraging robot places the materials in the resource-holder of the last robot in the robot chain that is closest to the foraging robot for dumping the resources. It should be noted that foraging robots can also select to deposit their resources directly into the central depot if it is closer than the final robots in any robot chain they are following. After dumping the resources, the robot returns to the place of the gathered resources until the cluster is depleted of resources. As soon as the resource cluster becomes depleted, the foraging robot will begin exploring the arena in search of more resource clusters. The exploration strategy is the same as that used in [26], which can be found here.

The fact that the location of robots are shared with each other, the system ensures that foraging robots have no difficulty discovering the last of a robot chain's robots. A foraging robot, on the other hand, should find a way to reach the last robot of the robot-chain while avoiding obstacles. *Obstacle map* are shared data structures that store information about the obstacles that are detected by all robots in the arena. An obstacle map partitions a two-dimensional map into regions of three different type: 1) Obstacle regions, 2) Empty regions, and 3) Unknown regions. The unknown regions are the unexplored regions that any robot did not visit before. The empty regions are the area that is free of obstacles and is explored by the robot. The obstacle regions are the area occupied by obstacles that robots cannot pass through. These regions are shown in Fig. 27b where Unexplored regions are represented as the black regions, empty regions are represented as the white regions, and the obstacle regions are represented as magenta regions. As demonstrated in Fig. 26, the entire obstacle map is initially comprised of a single unknown zone that is then updated based on the sensing information obtained by the lidars of the robots. The information from the lidar may be utilized to determine the corners and edges of obstacles, as well as the empty regions where there is no obstruction to be detected. Through the process of expanding the area of empty regions and highlighting the obstacles regions that are encompassed by the detected edges,

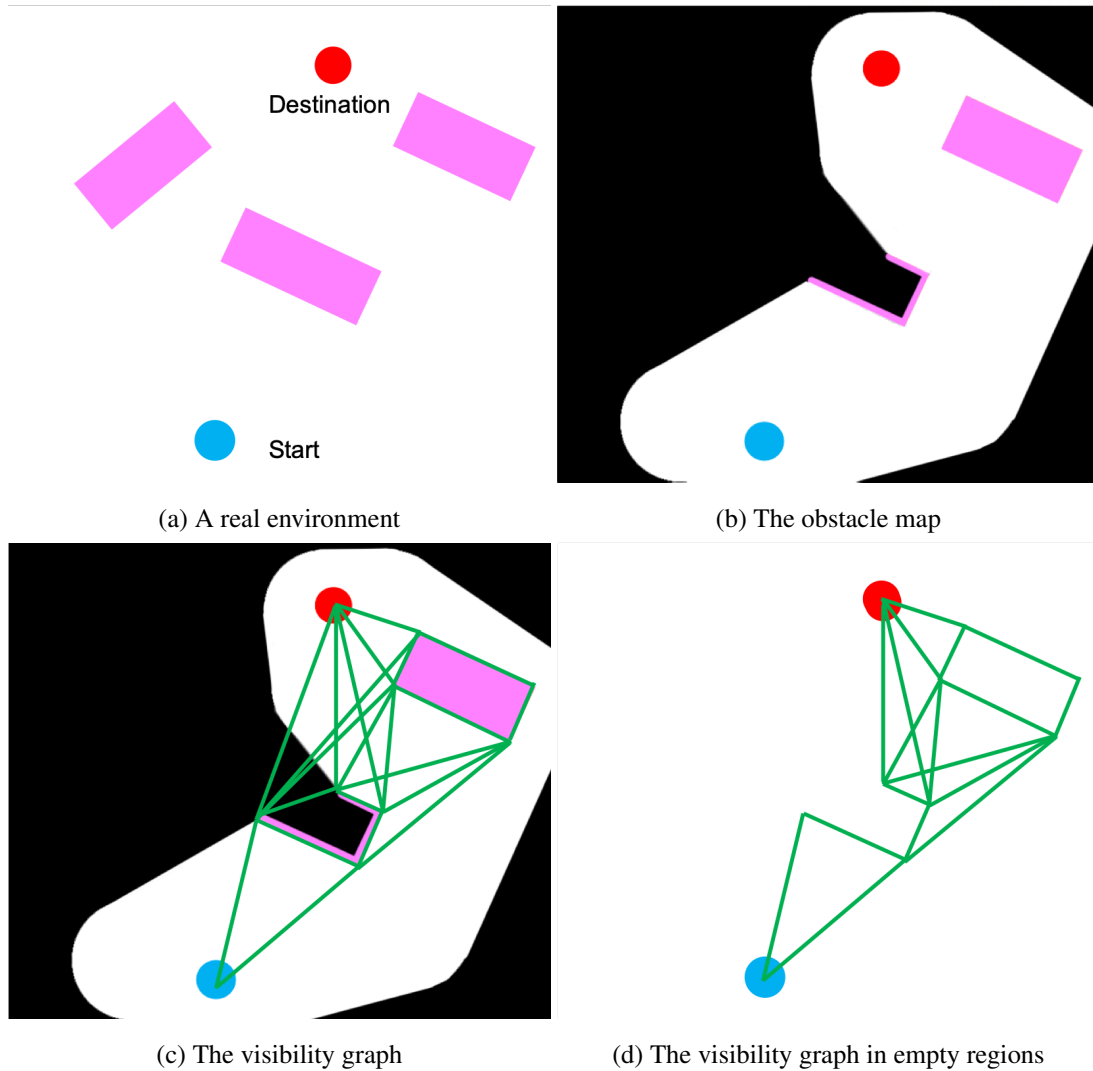


Figure 27: The visibility graphs in an obstacle map.

the robots are able to incorporate all of the information into the obstacle map. The robots communicate wirelessly with one another in order to share the obstacle map.

When a foraging robot wants to travel to a site, it will first generate the visibility graph based on the obstacle map (Fig. 27c). In the visibility graph, the algorithm removes all edges, which are not in the empty regions (Fig. 27d). Based on the visibility graph, the algorithm found the shortest path from current location of a robot to the destination of the robot.

After then, the foraging robot will go along the shortest route between its present position and the visibility graph's target. But, I also urge the foraging robot to periodically go into uncharted locations to examine them. I used epsilon greedy exploration approach to determine when a foraging robot should explore unfamiliar places. There is a slight chance that a foraging robot would pick this route over the shortest path in the empty region if it visits a node in the visibility graph that is incident to an edge that reaches an unknown region. When the robots explored the unknown area, the robots share the obstacles map to other robots. The foraging robot will then recalculate the shortest route based on the updated

visibility graph and other information about the unknown region. When a foraging robot's exploration meets a dead end, it will return to the previous node by the shortest route. If the foraging robot cannot locate the shortest way to the destination in the empty region from its present position, it will randomly explore the unknown region until it discovers a path to the target location.

#### 5.4 Relocation of Robot Chains

By relocating, dynamic depots may drastically shorten the time required for foraging robots to carry resources [26]. Similarly, a robot chain should reposition its last robot so that it is closer to resource clusters. Through the relocation of the robot-chain, the robot-chain can provide a short distance between the last robot of the robot-chain and the resource cluster to the foraging robots. However, relocating the robot chain takes time, so doing relocating too often can be reduce the performance of the system. In this case, the considerations are whether and how to relocate a robotic chain.

This system decides whether a robot chain should relocate only after a certain amount of time  $t_{protect}$  has passed after the last relocation, such that the system would avoid the relocation too frequently. After a certain amount of time  $t_{protect}$ , The robot chain will determine whether its usage rate stays high. Utilization of a robot chain is the pace at which resource-gathering robots transfer resources to the last robot's resource-holder. As long as the rate exceeds a particular threshold  $\alpha_{protect}$ , the foraging robots may continue to utilize the robot chain without being relocated. Although the relocation of the chain reduces the travel distance, if the foraging robots are constantly dumping resources to the robot-chain quickly, there is no need to stop other foraging robots and start relocating the robot-chain. The system will determine whether a more optimal site exists for the robot chain. Firstly, a goal position is computed for a robot-chain. When relocation begins, the relocation process explained in the next section seeks to position the final robot as near as feasible to the goal site. The following equations may be used to determine the desired location  $(x, y)::$

$$x = \frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i} \text{ and } y = \frac{\sum_{i=1}^N w_i y_i}{\sum_{i=1}^N w_i}. \quad (3)$$

where  $w_i$  is the estimated number of remaining resources in the resource cluster  $i$  in a set  $\mathcal{R}$  of resource clusters,  $N$  is the total number of locations where robots have detected resources in  $\mathcal{R}$ , and  $(x_i, y_i)$  is a the coordinate of  $i$ .  $\mathcal{R}$  is a set of non-empty resource clusters near the last robot of the robot chain.  $\mathcal{R}$  contains all non-empty resource clusters when determining the target point in the last relocation. In addition, new resource clusters will be added to  $\mathcal{R}$  (e.g, the next known resource cluster on the robot chain's right side that isn't taken into account while computing the last target point). If the target location is too close to the last target location, there is no need for relocation. When the relocation of the robot chain starts, foraging robots have a waiting time because the foraging robots cannot use the robot chain until the relocation of the robot chain is finished.

If the system determines that a robot chain should not be relocated, the system will wait for a short time  $t'_{protect}$  before making a new decision. If not, the robot chain will cease gathering resources from foraging robots and wait until all resources presently on the robot chain have been delivered to the

central depot. All robot chain links are then disconnected, and all robot-chain robots transition to the mobile state. (see Fig. 28).

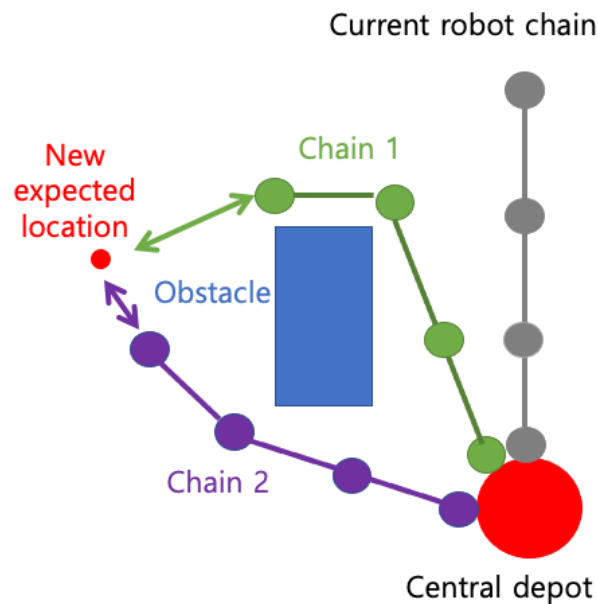


Figure 28: The relocation of robot chain. The current robot chain (grey lines) relocates to a new location (red small circle). Robots (colored circles) explore the regions when they travel to the new location. An obstacle (blue rectangle) is discovered and two robot chains (green and purple lines) are computed. The last robot of robot chain 2 (purple lines) is closer to the new location than the robot chain 1 (green lines).

The robots will then try to form a new robot chain such that the last robot is as near as feasible to the desired position. Due to barriers in uncharted places, the perfect design of the new chain could be pre-calculated, and the robots must hunt for the best locations as they explore the area. If any new locations of robots in the robot chain are inside of barriers, the robots continue to travel to those places and explore the areas around the new sites for a period of time known as the exploration time  $T_{\text{explore}}$ . The primary objective of a relocation method is to preserve the best configuration of a robot chain during the exploration process and to ensure that robots have sufficient time to return to the previous best configuration before the permitted exploration time expires. A configuration of a robot chain is, more accurately, a series of positions and orientations of the robots on the robot chain. Initially, the optimal configuration is the one that existed prior to initiating reallocation. The system expands the visibility graph by linking the central depot and the target location to the graph's nodes, which is obtained from the current obstacle map's visibility graph. Then, in the unoccupied areas, locate the quickest route between the central depot and the destination.

On the basis of this shortest route, the system compute the locations of the robots that may be placed on or near the shortest path to construct a robot chain in the vacant region along the shortest path beginning at the central depot. All robots will be relocated to these new places as soon as feasible, since they represent the optimal arrangement going forward. The robots will then be able to expose further



vacant areas and impediments in the empty region, and the visibility graph will be updated appropriately. The robots must monitor the length of time  $t_{\text{return}}$  required to return to their locations in the present optimal configuration. When the remaining exploration time equals  $t_{\text{return}}$ , the robots instantly cease exploring, return to their current best configuration, and form a robot chain. When robots investigate uninhabited places, the system generates a list of potential robot chains linking the new site to the central depot. The optimal configuration is the robot chain with the least distance between the new site and the final robot location.

## 5.5 Experimental Configurations

To check the performance of this robot chain algorithm, I ran two experiments sets using ARGoS. I analyzed the findings statistically to see whether the foraging performance changed predictably with various setups. In both tests, I compared dynamic robot chain algorithm  $RC_{\text{dynamic}}$  (i.e., dynamic robot chains) with two MPFA with dynamic depots algorithms, ( $MPFA_{\text{dynamic}}^3$  and  $MPFA_{\text{dynamic}}^{16}$ ).

As described in [26], 4 dynamic depots are spread and may migrate to new sites dynamically in the two MPFA algorithms. The capabilities of depots change between the two MPFA algorithms: 3 in the  $MPFA_{\text{dynamic}}^3$  and 16 in the  $MPFA_{\text{dynamic}}^{16}$ .

I developed the  $RC_{\text{static}}$  algorithm, which prohibits the movement or relocation of robot chains. The number of clusters is twenty, and their shapes are  $5 \times 5$ ,  $5 \times 10$ , and  $10 \times 10$  for 500, 1000, and 2000 resources, respectively. To analyze the foraging performance of the algorithms, I increased the amount of resources, the number of robots, and the size of the arena in the first set of trials. A portion of robots were used to initiate the robot chains. Table 4 provides an overview of the experimental setup.

Table 4: The Configuration in Experiment 2-1

|  |                      |                      |                         |
|--|----------------------|----------------------|-------------------------|
| Arena Size( $m \times m$ )               | $10 \times 10$       | $20 \times 20$       | $40 \times 40$          |
| Number of resource                       | 500                  | 1000                 | 2000                    |
| Number of robots                         | 20, 30, 40<br>50, 60 | 40, 50, 60<br>70, 80 | 60, 80, 100<br>120, 140 |
| Foraging time (minute)                   | 30                   |                      |                         |
| % of robots for the initial robot chains | 30%                  |                      |                         |

In the second experiment, I analyzed the performance of foragers in area with varying numbers of obstacles. Obstacle are randomly oriented, simulated box measuring  $0.5m \times 0.5m \times 0.5m$ . I examined four distinct numbers of obstacles: 4, 8, 16, and 32. In addition, I adjusted the cluster size to  $5m \times 10m$  with 20 clusters.

## 5.6 Experimental Results

The performance of foraging is measured by the quantity of resources gathered and transported to the central gathering zone. When checking the performance of the system, it excludes the number of re-

Table 5: The Configuration in Experiment 2-2

|                            |                |
|----------------------------|----------------|
| Arena Size( $m \times m$ ) | $20 \times 20$ |
| Number of resource         | 1000           |
| Number of robots           | 40, 60, 80     |
| Foraging time (minute)     | 30             |
| Number of obstacles        | 4, 8, 16, 32   |

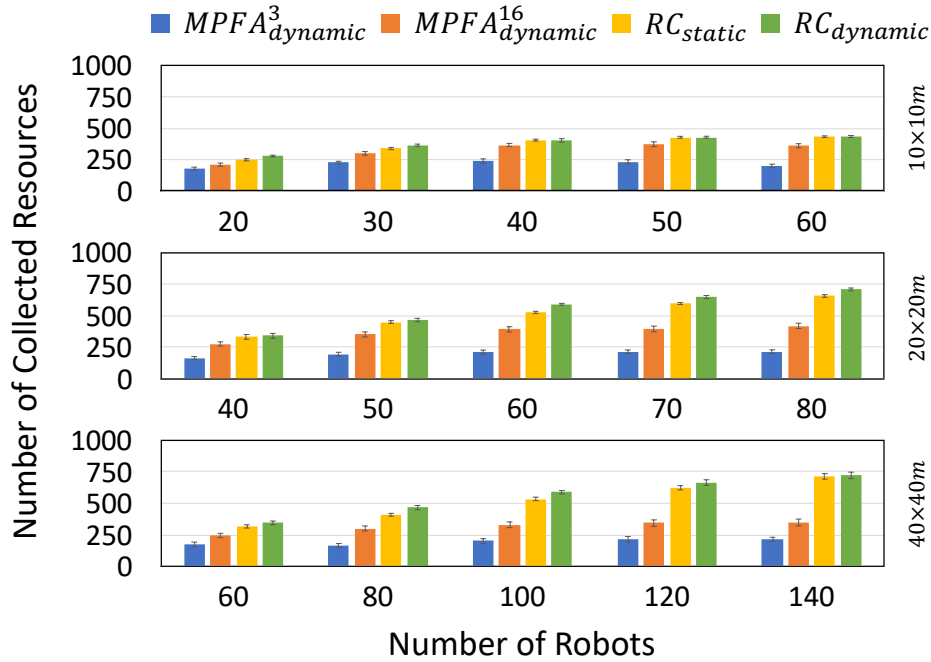


Figure 29: Foraging performance with varying numbers of robots in various areas in Experiment 2-1.

sources held by foraging robots or robot chains. Only the number of resources that arrived at the depot at the time of termination is counted. The collision-time is the robot's spending time required for avoiding colliding with one another and the area's boundary. Based on the collision time, the experiment's robot congestion may be determined. I studied the findings statistically to see whether the foraging performance changed predictably with various setups. Each data set shown in the graphs is an average of 30 runs, and each error-bar represents 95% confidence intervals.

Fig. 29 illustrates the foraging performance of 4 different algorithms in Experiment 2-1.

The two robot chain algorithms are more efficient than the two MPFA algorithms. Except for MPFA<sup>3</sup><sub>dynamic</sub>, all performance increases as the number of robots and area size rise. Both robot chain algorithms expand more rapidly than their respective MPFA counterparts. The findings reveal that the performance of RC<sub>dynamic</sub> is 236% greater than MMPFA<sup>3</sup><sub>dynamic</sub> and 106% greater than MPFA<sup>16</sup><sub>dynamic</sub> when the number of robots is 140 and the area of the arena is 40m × 40m.

Experiment 2-1 examines the collision time of each 4 method in Figure 30. The collision time in the MPFA with large capacity dynamic depots is 37% longer than in the RC<sub>dynamic</sub>. The depots with a

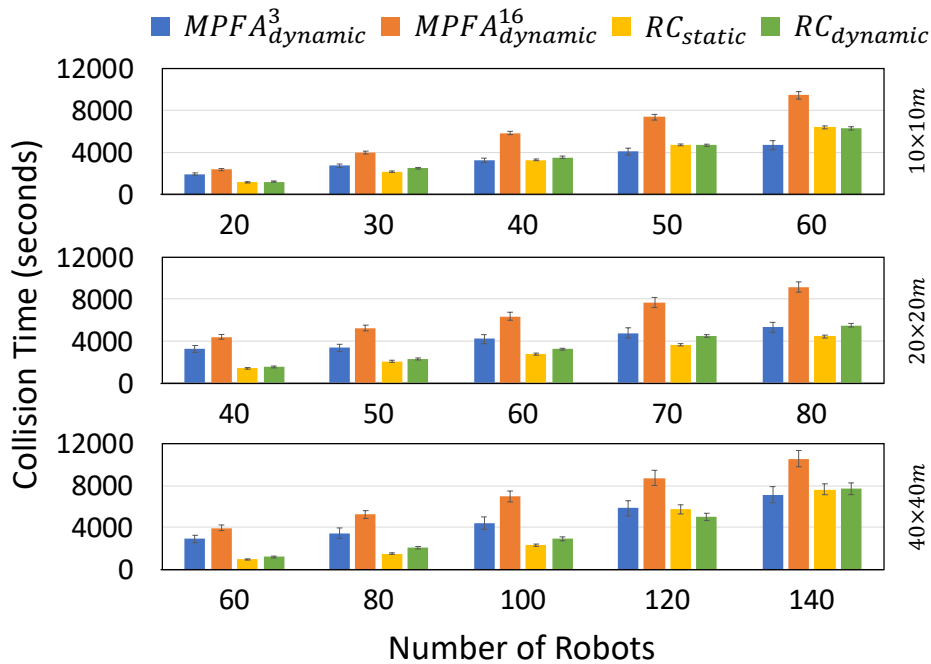


Figure 30: The collision time of each swarm in Experiment 2-1.

capacity of 16 have a longer collision time than depots with a capacity of 3. The difference in collision time between the two robot chain algorithms is not readily apparent.

Experiment 2-2’s Fig. 31 shows the foraging performance of all four algorithms with varying numbers of randomly dispersed obstacles. As the number of hurdles grows, so do all performances. Using 40 robots,  $MPFA_{dynamic}^{16}$  beats the other three methods. It is 40 percent, 44 percent, 47 percent, and 46 percent, respectively, more expensive than  $RC_{dynamic}$ . The performance of the remaining three methods does not vary much. When the number of robots reaches 60,  $RC_{static}$  and  $RC_{dynamic}$  perform marginally better than  $MPFA_{dynamic}^{16}$ . Compared to the performance of forty robots, their performance improves more rapidly than the two MPFA algorithms. When the number of robots reaches 80, the improvement in  $RC_{static}$  and  $RC_{dynamic}$  becomes more pronounced. In addition,  $RC_{dynamic}$  performs better than  $RC_{static}$  when the number of obstacles is 8, 16, or 32.

## 5.7 Summary

In this part, I demonstrated that by allowing robots in a robot swarm to move objects at a distance, they are able to do foraging tasks more efficiently as a group. The most essential characteristic is the construction of robot chains that are capable of overcoming the two most fundamental limitations of existing dynamic depot foraging systems. One is congestion in the middle of the collecting zone. Another aspect is the duration of the delivery trip. To do this, I designed a novel robot chain-based foraging swarm system. I presented the robot chain control algorithm and explored the dynamic robot chain relocation method. In multiple-location foraging tasks, these studies reveal that, given the same number of robots, dynamic robot chains perform better than dynamic depots.

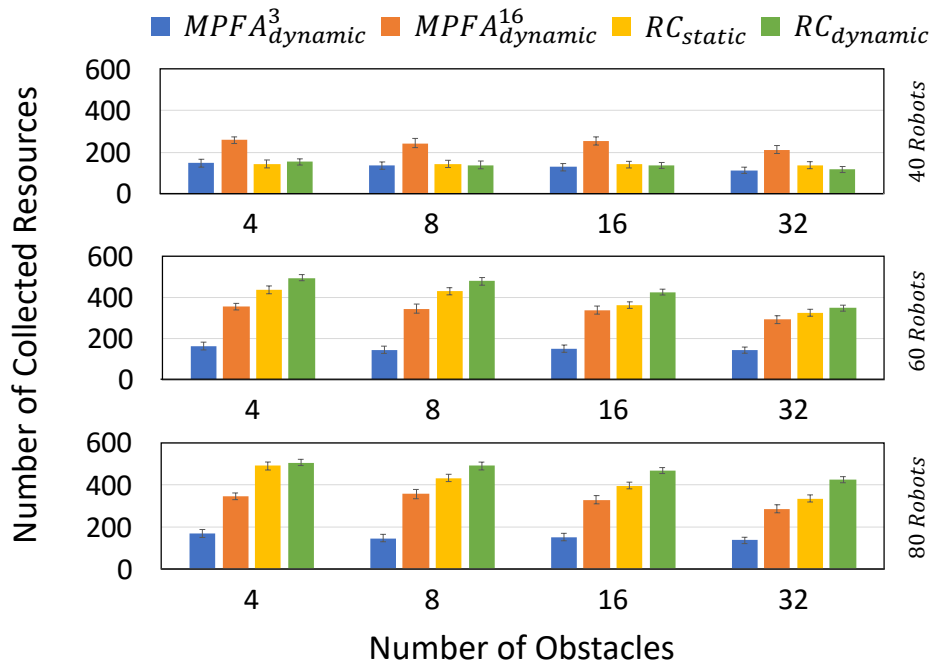


Figure 31: Foraging performance with varying numbers of robots in various areas in Experiment 2-2

In this section, I proved that by enabling robots in a swarm robotics for transferring goods at a distance between robots, the robots are able to do foraging tasks more effectively collectively. The most important feature is the forming of robot chains that are capable of overcoming the two most fundamental limitations of existing MPFA with dynamic depots. One is congestion around the collecting area’s center location. Another factor is the lengthy delivery journey distance. To this objective, I developed an unique robot chain-based foraging swarm system. I provided the robot chain algorithm for controlling robots and discussed the dynamic robot chain relocation technique. These tests demonstrate that, given the same number of robots, dynamic robot chains perform better than dynamic depots in multiple-location foraging tasks.

The following explains the higher performance of robot chain algorithms. In MPFA<sup>3</sup><sub>dynamic</sub> and MPFA<sup>16</sup><sub>dynamic</sub>, when the capacity of the depot robots’ resources is as low as 3, the depots must regularly visit the central depot. Therefore, the foraging robots must wait longer for the return of the stores. RC<sub>static</sub> and RC<sub>dynamic</sub> do not have this issue since they may continually obtain resources from foraging robots after robot chains are built; consequently, foraging robots have a high utility rate. In RC<sub>dynamic</sub>, the robot chains move just a few times, and are thus operational for the most of the time. When robot chains migrate, RC<sub>dynamic</sub> experiences somewhat more collisions; nevertheless, the relocation updates robot chains to better places for gathering more resources.

In Fig. 30, the majority of collisions in MPFA<sup>3</sup><sub>dynamic</sub> and MPFA<sup>16</sup><sub>dynamic</sub> are caused by congestion around the depot robot and the central depot. Due to the huge number of robots in close proximity to the depot robot or the central depot, it takes longer for the robots to unload resources at the depot robot or the central depot. In addition, the depot robot’s capacity is the difference in collision time be-

tween  $MPFA^3_{dynamic}$  and  $MPFA^{16}_{dynamic}$ . The diminished capacity leads to an increase in the number of resource delivery trips, and therefore, accidents. When the foraging robots are awaiting the depot robot, there is no chance of collision between the robots that want to discharge supplies to the depot robot. Consequently, the  $MPFA^3_{dynamic}$  exhibits lower collision times than the  $MPFA^{16}_{dynamic}$ . The collision at the central depot may be avoided nearly entirely in  $RC_{static}$  and  $RC_{dynamic}$ ; instead, most collisions occur at the ends of the robot chains. Since there are four robot chains, collisions are dispersed equally among them, resulting in reduced congestion. When the robot chains are moved in  $RC_{dynamic}$ , some collisions occur around the central depot. When moving to new sites, robots clash with one another to establish a new robot chain. When a depot goes to the central collecting zone to transfer resources, foraging robots in its vicinity are idle and await its return. The depot with a capacity of three moves more often than the depot with a capacity of sixteen. Consequently, there is less collision, and the robots must wait longer.

When obstacles are placed in foraging areas, all foraging performance is disturbed and rapidly decreases.  $RC_{static}$  performs almost identically to  $RC_{dynamic}$ . It suggests that the relocation is insufficiently effective for small swarm numbers and tiny venues.  $RC_{dynamic}$  has the potential to perform much better as the number of robots increases.

This study demonstrates that MPFA with dynamic robot chains may significantly increase the foraging performance of robotic systems compared to current methods. 1) It is more efficient than existing CPFA and MPFA with limited depot capacity; 2) it reduces robot collision time; and 3) the system can recognize and avoid obstacles.

## VI Dynamic Robot Chain Networks

The objective of a foraging swarm robot system is to find and collect resources in an unknown arena as quickly as possible. To avoid the congestion near the central depot, I previously described an extension of the multiple-place foraging system in which robots can form robot chains that deploy dynamically to reduce the distance by receiving resources at the robot chains instead of the central collection zone. However, a robot chain can only reach one target location at a time, and congestion can occur at the end of the robot chain. This section presents an extension to dynamic robot chains called *robot chain networks*, which extends robot chains with branches, each of which reaches different resource clusters. I formulate the problem of finding the smallest robot chain networks as the Euclidean Steiner tree problem and explain how Steiner trees can be utilized to optimize the efficiency of the foraging operations. I implemented the foraging swarm robot system in the swarm robot simulator called ARGoS. The experiments showed that the system deployed robot chain networks could avoid obstacles and collect more resources when compared with the original robot chain design.

### 6.1 Introduction

An important application of swarm robotics is foraging, which is the search for and return of scattered resources like as minerals, water, and fuels to a collecting zone termed *central depot* [112, 113]. Most extant foraging swarm robot systems use decentralized search-and-gather foraging techniques. [23, 26, 114–117]. However, several of these foraging systems have congestion around the central depot when a large number of foraging robots return to the depot to unload the acquired resources [26]. When congestion arises, the central depot serves as the system’s bottleneck. To alleviate the congestion issue, Lu et al. [23, 26, 117] proposed a foraging system known as *multiple-place foraging systems*, which employs helper robots known as *dynamic depots* to collect resources from foraging robots and transport them to a central depot. Even though dynamic depots may lower the number of robots near the central depot, high traffic still causes congestion. Previously, I considered replacing dynamic depots with *dynamic robot chains*, which are sequences of robots that can transmit resources across a distance. [29]. As illustrated in Fig. 24, a proposed implementation of dynamic robot chains is based on mobile conveyors [33]. A foraging robot is able to place supplies on the last robot in a robot chain and have the chain deliver the resources to the central depot. This strategy may significantly alleviate traffic congestion close to the central depot [29].

Dynamic robot chains, however, cannot eliminate congestion since congestion might still exist near the end of robot chains. When there are many resources near the end of a robot chain, many foraging robots have to deposit the collected resources into the robot chain. Thus, the end of robot chains becomes the new bottleneck. In this section, I propose *dynamic robot chain networks*, which extends the idea of dynamic robot chains by having a network of robot chains that connect to multiple resource clusters. By allowing robot chains to have branches, the traffic near the end of the robot chains can be distributed to several branches, and congestion can be reduced. More importantly, the dynamic robot chain networks can achieve higher throughput by having multiple endpoints connecting to the different

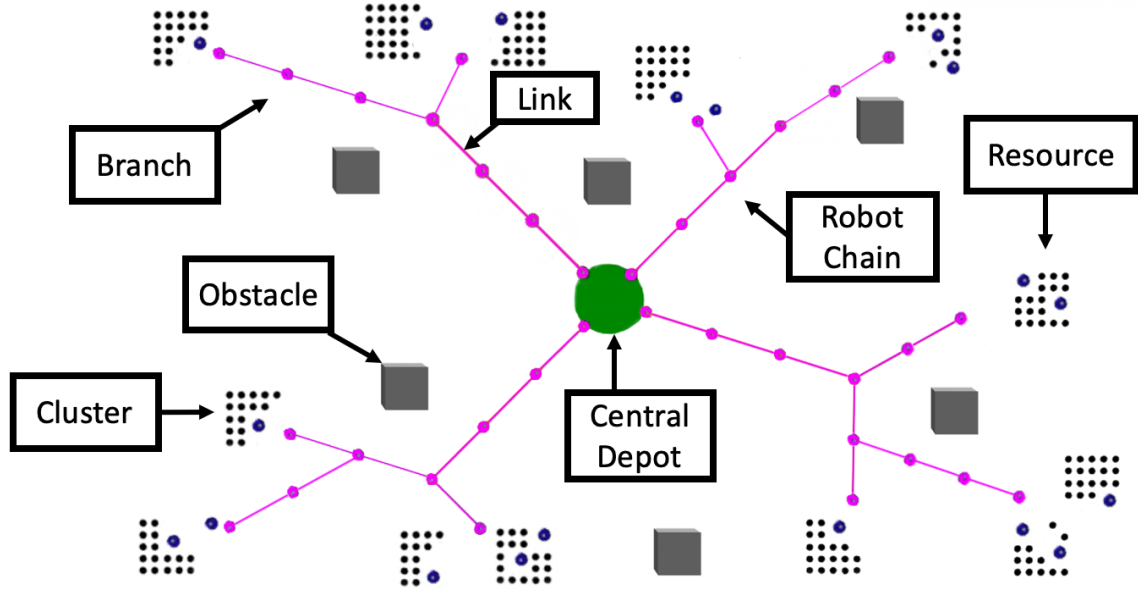


Figure 32: The example of four robot chains networks in the arena. The magenta dots are robots of the robot-chain, whereas the blue dots are foraging robots.

resource clusters using a few robots. I optimized the number of robots on a robot chain network by computing a *Euclidean Steiner tree with obstacle avoidance* that is the minimum spanning tree for connecting to multiple endpoints with the addition of Steiner points. I describe a high-level controller which decides when I should add new branches to an existing network and when I should relocate a subtree of a network to the locations on the Steiner tree. The experimental results present in Section. 6.7 demonstrate that MPFA with robot chain networks are more effective than both MPFA with dynamic depots and robot chains with no branch.

## 6.2 Foraging with Robot Chain Network

In a foraging task, multiple robots cooperates to gather resources in *an area*. The arena shown in Fig. 32 is one in which resources are placed in many *resource clusters*. In addition, there are obstacles that impede the robot's sight and mobility. Initially, the robots are unaware of the locations of the resource clusters and obstacles and must conduct reconnaissance to find them. The objective of the robot squad is to recover as many resources as possible and transport them to the central depot.

A robot is an omnidirectional mobile robot with a link forming component, a lidar sensor, a gripper, a wireless communication device, a resource detection device, and a resource-holder with limited storage space. A robot uses its lidar to identify the edges of the obstacles and other robots within the lidar's sensing range (see Fig. 26). The resource detection device can detect the exact locations of all resources within sensing range of the detection device. When the distance between two robots is less than  $d_{\max}$ , the two robots can use their link forming components to form a *link* between them such that one robot can send resources to the other robot via the link. The *capacity* of a link is the maximum number of resources that can transfer on the same link simultaneously.



A *robot chain* is a sequence of robots in which every pair of adjacent robots are linked together. A *robot chain network* is a set of robot chains that join together in a tree-like structure. To form a robot chain network, robots must be capable of establishing links to three adjacent robots to form Y-junctions. Three mobile conveyors can form a one-way Y-junction by overlapping the endpoints of their belts such that objects from two incoming belts can drop on the outgoing belt. The problem of finding the minimum size robot chain networks can be formulated as a *obstacle-avoiding Euclidean Steiner tree problem*: given  $N$  points in a 2D plane with obstacles, find a tree structure in the empty space that connects all points by lines of minimum total length with the help of newly-created points called Steiner points. In these cases, the  $N$  points are the locations of resource clusters, the lines are robot chains, and the Steiner points are Y-junctions. One nice theoretical result is that every vertex in a minimum Steiner tree must have a degree of two or three. Hence, it is sufficient to consider Y-junctions only when forming a robot chain network of minimum size.

At any time, a robot is in one of the four states: 1) *the mobile state*, 2) *the resource-collecting state*, 3) *the resource-dumping state*, and 4) *the robot-chain state*.

Robots can freely move in the mobile state. A robot can enter to the resource-collecting state and use own gripper to put a resource in own resource-holder when the robot arrives at a resource. After gathering the resource, the robot returns to the mobile state. When the robot arrives at an endpoint of a robot chain network or the central depot, it can enter the resource-dumping state to unload the resource.

Once the resource has been unloaded, the robot returns to its mobile state. When the robot is instructed to establish a link with other robots, the robot enters the robot-chain state and starts forming links with other robots. When a branch in a robot chain network is disbanded, the links are retracted, and the robot enters the mobile state. At any time, a robot plays one of the following *roles*: 1) *exploring robots*, 2) *foraging robots* and 3) *robot-chain robots*. An exploring robot can only enter the mobile state.

A foraging robot can enter the resource-dumping state, the resource-collecting state, and the mobile state. A robot-chain robot can enter the robot-chain state and the mobile state. A robot can change its role only when it is in a mobile state.

An *endpoint* is the last robot of a branch that connects to a resource cluster. Each foraging robot has a *preferred* endpoint, which is typically the nearest endpoint at the current position. A foraging robot can put its collected resources to the preferred endpoint, and then the resource will be transferred to the central depot via the network. If the resource-holder of an endpoint is full, or another foraging robot is using the endpoint, and the foraging robot has to wait until the endpoint becomes available. The resource-holder of an endpoint is full when *congestion* occurs on the network, causing the resources to pile up all the way to the endpoint. Congestion is mainly caused by the limited capacity of the link on the *main branch* that connects to the central depot. But this congestion can be alleviated if resources can be transferred quickly to the main branch. In experiments, there are four initial robot chain networks in the four quadrants of an arena in the beginning.

### 6.3 Foraging Robots Behavior

A foraging robot saves the location of the previous resource cluster it visited and collects resources from the cluster until there is no more resource in the cluster. When all resources in a cluster have been collected, the foraging robot will go to another cluster to collect resources or become an exploring robot and explore the area to discover other resource clusters and obstacles based on the exploration strategy in [26].

Robots share their locations wirelessly to the nearby robot, especially the adjacent robots on a robot chain. They also share the locations of the resources and the obstacles they know. The central depot acts as an information hub that redirects information via robot chain networks. Unless a robot is exploring a location that is far away from any robot chain network, it can receive the shared information from the central depot instantly. Robots that are not close to any network will have to get closer to a network from time to time to exchange information with other robots.

When a robot discovers an obstacle, they will integrate the information of the obstacle collected by their lidars (Fig. 26) by updating a shared data structure called the *obstacle map* (Fig. 27), which partitions the arena map into three kinds of regions: 1) obstacle regions, 2) empty regions and 3) unknown regions. In Fig. 27b, they are the magenta regions, the white regions, and the black regions, respectively. Foraging robots use the obstacle map to navigate in the arena. To visit a location, a foraging robot computes a *visibility graph* in the obstacle map (Fig. 27c) and removes all edges that are not in the empty regions (Fig. 27d). The foraging robot walks along the shortest route to the destination as determined by the graph of visibility. When traveling toward the goal, the foraging robot will update the visibility graph and recalculate the shortest route based on new knowledge about undiscovered places. If there is no way to get to the target from the empty area, the robot that is looking for one will randomly explore the area until it finds one.

### 6.4 Modification of Robot Chain Network

A robot chain network supports five modification operations: (1) adding a branch to an existing network, (2) deleting a branch from an existing network, (3) relocating a subtree in an existing network, (4) establishing a new network, and (5) disbanding a network. A *high-level controller* in Sec 6.5 decides when to apply these operations. Typically, when a new resource cluster is discovered, the high-level controller will consider either adding a branch connecting an existing network to the new resource cluster or relocating a subtree in an existing network to include the new resource cluster as a new endpoint in the network. When a resource cluster has no more resources, the branch to the resource cluster will be deleted. If all resource clusters at the endpoints of a network are empty, the entire network will be disbanded. If some resource clusters are not close to any existing networks, a new network to these clusters will be established.

The relocation of subtrees relies on the solution to the obstacle-avoiding Euclidean Steiner tree problem. For each resource cluster in a subtree  $\pi$  including the new resource cluster, I pick a point within a distance  $R_{end}$  from the center of the cluster but outside the cluster as an endpoint. The endpoint

should be between the center of the cluster and the network. Given the coordinates of a set of endpoints and the obstacle map, I first compute a nearly optimal Steiner tree that connects the central depot to the endpoints while avoiding the obstacles. I modified the Euclidean Steiner tree solver in [38] to make it works with obstacle avoidance using some heuristics in [35] and [37]. Second, I compute the *target locations* of the robots on the edges of the Steiner tree. The target locations should satisfy all physical constraints, such as the maximum link distance and the Y-junction configurations. After computing the target locations, the high-level controller will ask the endpoints in  $\pi$  to stop receiving resources from foraging robots and wait until all existing resources on the chain leave  $\pi$ . Note that other parts of the network can continue to function while relocating  $\pi$ . After that, the robot-chain robots in  $\pi$  are disbanded and turn them into foraging robots. Then a set of foraging robots that are close to the target locations are selected and moved to the target locations along the shortest path in the visibility graphs as described in Sec. 5.3.

Due to obstacles in unknown regions, the Steiner tree has to be recomputed from time to time during relocation. To encourage robots to explore the regions around the new subtree, the robots are given a certain amount of time, namely  $t_{\text{explore}}$ , to freely explore the region near the subtree while moving to the target locations. The primary concept of the exploration method is to retain the current best configuration of the subtree during the exploration and to ensure that robots have sufficient time to return to the previous best configuration before the allocated exploration time  $t_{\text{explore}}$  expires. When robots collect new information about the obstacles during exploration, the high-level controller will recompute the Steiner tree and the target locations and ask the robots to move to the new target locations.

The operation of establishing a new network is the same as the relocation of a subtree, except no existing subtree will be disbanded, and the subtree's root is the central depot. Adding a new branch is the same as the addition of new robot chains using the visibility graph in Fig. 27 as discussed in [29]

## 6.5 The High-level Controller

The foraging robots and the dynamics of robot chain networks are managed by a high-level controller at the central depot. The controller divides all robots into three groups based on their roles as discussed in Sec 6.2. An exploring robot will move to the nearest unknown regions in the obstacle maps as shown in Fig. 27, but visit the nearest endpoints of any network for every period of time  $t_{\text{info}}$  to report the collected information. The foraging robots will be evenly distributed to the endpoints of the networks and repeatedly collect resources from nearby resource clusters and then dump the resources to the endpoints. The robot-chain robots act reactively upon receiving resources from the upstream and push the resources to downstream.

Robots can switch roles only when a modification of robot chain networks occurs. The high-level controller considers modifying any networks only at a fixed time interval  $t_{\text{protect}}$ . No new modification can be made before the end of a time interval in order to prevent modifications from happening too frequently. At the end of the time interval, the controller checks whether some branches or networks should be removed and whether there are resource clusters that have not been reached by any network.

To reach these resource clusters, the high-level controller has three options: adding new branches to an existing network, relocating a subtree in an existing network, or establishing a new network. The last option is selected only when the resource clusters are too far away from existing networks.

I opt for building large robot chain networks with as many endpoints as possible after setting aside a certain number of robots as exploring robots. Hence, the high-level controller prefers adding new branches instead of relocating a subtree. When adding a new branch, some foraging robots have to convert into robot-chain robots. As long as the average number of foraging robots for each endpoint of a robot network is not less than a given number  $N_{\text{forage}}$ , the addition of new branches should be allowed; otherwise, the high-level controller should relocate a subtree. In relocation of a subtree, the controller calculates the number of robots that are available to establish the new subtree. If the number of robots is large enough for filling out the minimum Steiner tree, the relocation proceeds; otherwise, the relocation is put on hold until some other branches are disbanded, and more robots become available. Since the new Steiner tree can be smaller than the subtree before relocation, some robot-chain robots can become foraging robots after relocation.

## 6.6 Experimental Configurations

To evaluate the robot chain network  $RC_{\text{network}}$ , I conducted four experiments on ARGoS. In first two experiments, I compared a robot chain network algorithm  $RC_{\text{network}}$  to the other two robot chain algorithms in [29], ( $RC_{\text{no}}$  and  $RC_{\text{one}}$ ). In  $RC_{\text{no}}$ , robot chains can relocate to close to multiple resource clusters, but the length of robot chains does not change. In  $RC_{\text{one}}$ , the robot chains can be extended to the center of multiple resource clusters. In the first experiment, the arena size is  $20m \times 20m$ .

The 1000 resources are distributed as 40 clusters, and the shape of the clusters is  $5 \times 5$ . The number of robots is 40, 50, 60, and 70. Initially, four robot chains are distributed uniformly, and four robots in each chain. The simulated foraging time is 30 minutes. The exploration time of robots in robot chains is 2 minutes, and the frequency of checking relocation is 6 minutes. For testing the flexibility of the algorithm, I conducted the second experiment where the arena, resources, and robot swarm size are larger. The values of the parameters are summarized in Table 6.

In the experiment, I measured how the frequency of network modification and the frequency of robot chains' relocation  $t_{\text{protect}}$  affects the foraging performance in the  $20 \times 20$  arena with obstacles. The experimental setup is summarized in Table 7. In the last experiment, I measured how the exploration time  $t_{\text{explore}}$  affects the modification of the networks or robot chains in the  $20m \times 20m$  arena with obstacles. Three groups of experiments were conducted based on the number of robots. The experimental setup is summarized in Table 8.

Table 6: The configuration of experiments 3-1 & 3-2

|  |                                  |                                  |
|--|----------------------------------|----------------------------------|
| Arena size ( $m \times m$ )                  | $20 \times 20$                   | $30 \times 30$                   |
| Number of resource                           | 1000: ( $40 \times 5 \times 5$ ) | 2250: ( $90 \times 5 \times 5$ ) |
| Number of robots                             | 40, 50, 60, 70                   | 60, 80, 100, 120                 |
| Number of robots in each initial robot chain | 4                                | 9                                |
| Foraging time (minute)                       | 30                               | 30                               |
| $t_{\text{explore}}$ (minute)                | 2                                | 2                                |
| $t_{\text{protect}}$ (minute)                | 6                                | 6                                |

Table 7: The configuration of experiments 4-1

|  |                  |
|--|------------------|
| Arena size ( $m \times m$ )                      | $20 \times 20$   |
| Number of resource                               | 1000             |
| Number of robots                                 | 40, 60, 80       |
| Foraging time (minute)                           | 30               |
| Number of obstacles                              | 4, 8, 16, 32     |
| Percentage of robots in the initial robot chains | 30%              |
| $t_{\text{explore}}$ (minute)                    | 2                |
| $t_{\text{protect}}$ (minute)                    | 3, 6, 12, 24, 30 |

Table 8: The Configuration of experiment 4-2

|                                      |                  |
|--------------------------------------|------------------|
| Arena size ( $m \times m$ )          | $20 \times 20$   |
| Number of obstacles                  | 2, 4, 6, 8       |
| Number of robots in each robot chain | 4, 5, 6          |
| $t_{\text{explore}}$ (s)             | 30, 60, 120, 240 |

## 6.7 Experimental Results

The performance of foraging is measured by the quantity of resources gathered and transported to the central gathering zone. The collision time is the time taken for robots to avoid colliding with the boundary of the arena or other robots. We assessed the findings statistically to see if the foraging performance changed predictably with various setups. Each data set shown in the graphs is an average of 30 runs, and each error bar represents 95% confidence intervals.

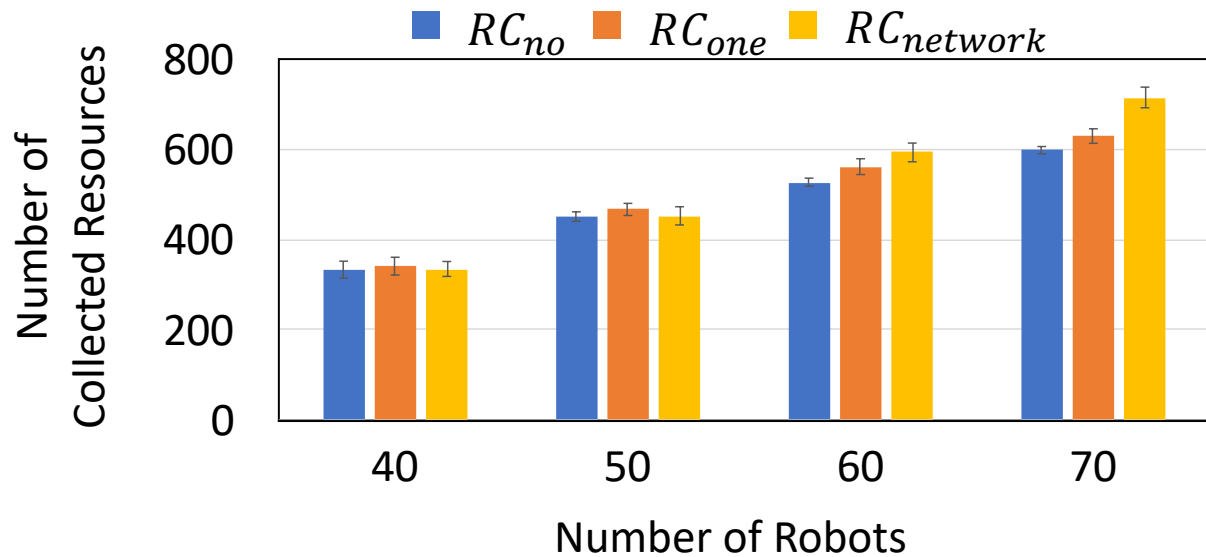


Figure 33: Foraging performance in Experiment 3-1.

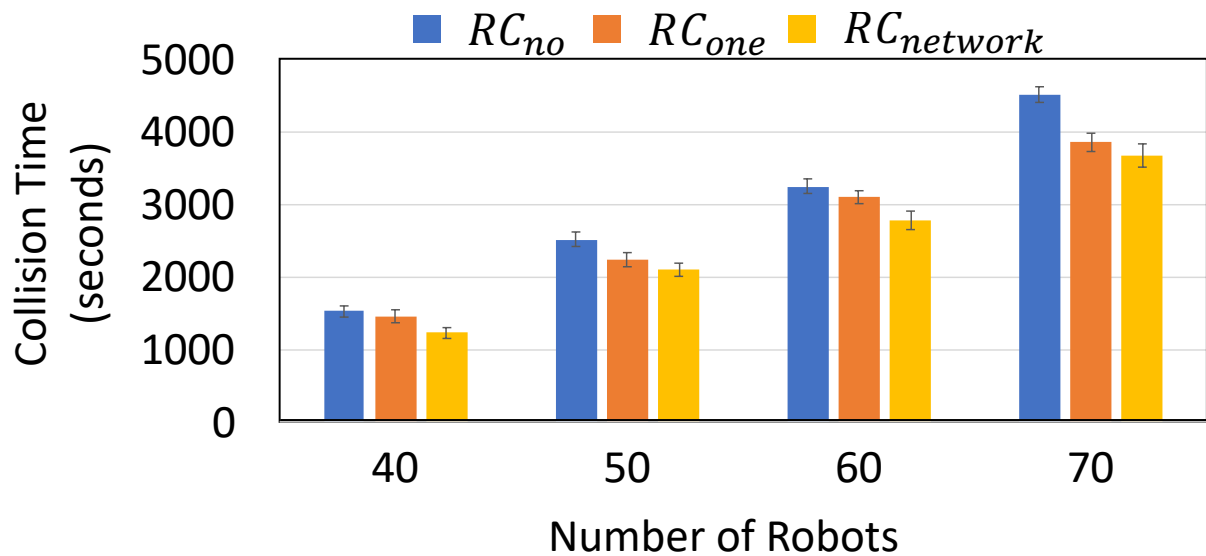


Figure 34: The collision time of each swarm in Experiment 3-1.

Fig. 33 demonstrates the foraging performance of all three algorithms in Experiment 3-1. The foraging performance increases as the number of robots increases. When the number of robots is 40, all performances are low and similar. When the number of robots is 60, the performance of  $RC_{network}$  outperforms other algorithms. When the number of robots is 70, the performance of  $RC_{network}$  is 13.6%

higher than  $RC_{one}$  and 19.5% higher than  $RC_{no}$ .

Fig. 34 compares the collision time of all three algorithms in Experiment 3-1. The collision time increases as the number of robots increases. The difference is higher when the number of robots is 70, the collision time in the  $RC_{network}$  is 4.7% shorter than the time in  $RC_{one}$  and 18.6% shorter than  $RC_{no}$ .

Compared to the performance in Experiment 3-1, Fig. 35 demonstrates the same trend of the foraging performance in Experiment 3-2. When the number of robots is 120, the performance of  $RC_{network}$  outperforms other algorithms. When the number of robots is 80, the performance of  $RC_{network}$  is 14.9% higher than  $RC_{one}$  and 21.0% higher than  $RC_{no}$ . Fig. 36 also demonstrates the same trend of the collision time in Experiment 3-2. When the number of robots is 120, the collision time in the  $RC_{network}$  is 4.5% shorter than the time in  $RC_{one}$  and 17.1% shorter than  $RC_{no}$ .

Fig. 37 demonstrates the foraging performance in Experiment 4-1. The foraging performance of the robot chains or networks decreases as the number of obstacles increases. When the number of robots is 40, the trend of the performance is not obvious. When the number of robots is 60 and 80, the performance with  $t_{protect} = 3$  is significantly lower than others. In all cases, the performance with  $t_{protect} = 6$  is slightly better than the performance with  $t_{protect} = 12$ , but always better than others.

Fig. 38 demonstrates the performance of the modification of networks and the relocation of robot chains in Experiment 4-2. The performance has the same trend in all three groups of experiments. The distance  $d$  increases as the number of obstacles increases. The shorter exploration time results in a longer distance  $d$ . The robot chain algorithm with the known obstacle map always has the best performance. When the number of robots in the robot chain is 6, and the number of obstacles is 8, the distance using the robot chain algorithm with the obstacle map is 35.8%, 52.6%, 181.2%, and 274.6% shorter than the distance using the robot chain algorithm with the exploration time, 240, 120, 60, and 30 seconds.



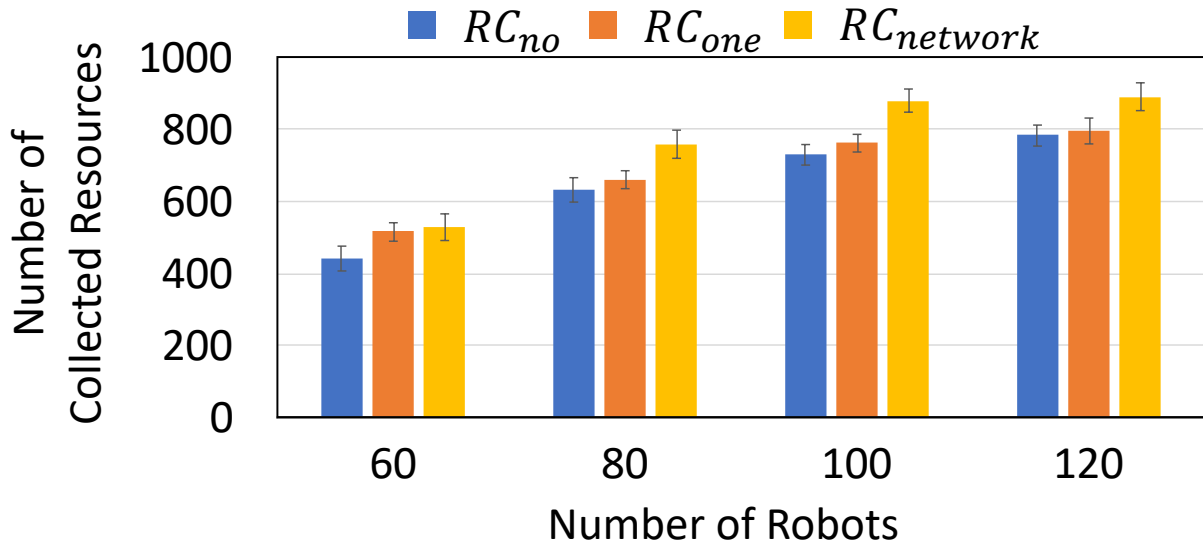


Figure 35: Foraging performance in Experiment 3-2.

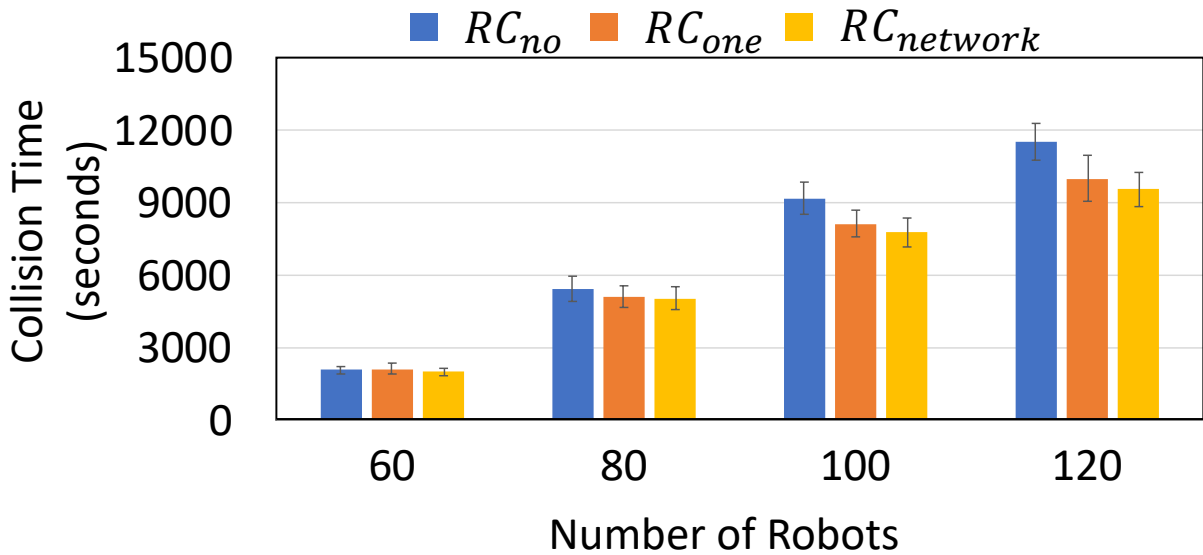


Figure 36: The collision time of each swarm in Experiment 3-2.

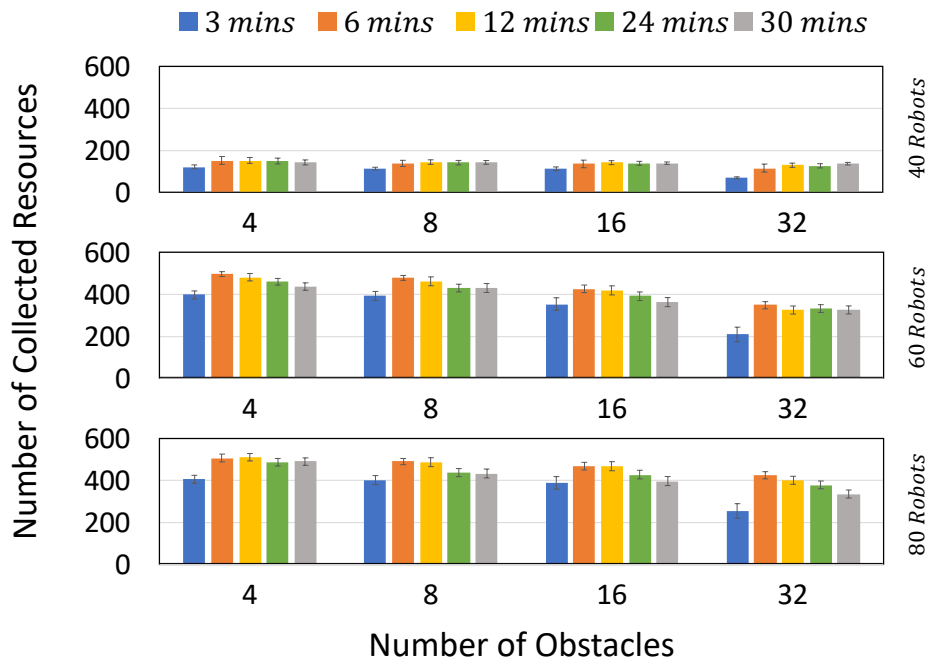


Figure 37: Foraging performance using different  $t_{\text{protect}}$  in Experiment 4-1.

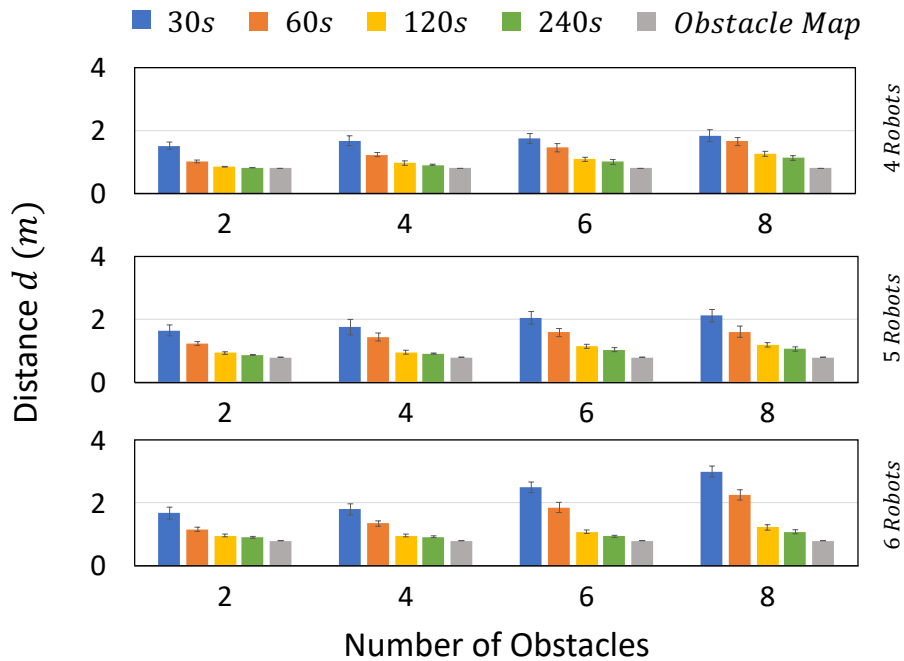


Figure 38: Relocating performance using a different  $t_{\text{explore}}$  in Experiment 4-2.

## 6.8 Summary

In section V, I have already demonstrated that the dynamic robot chain algorithm outperforms the MPFA with dynamic depots. In this section, I designed tree-like robot chain networks for foraging robot swarms based on obstacle-avoiding Euclidean Steiner trees. The collected objects can be transported on the robot chain from their discovered locations to the central collection zone. Compared to existing multi-places dynamic depot foraging systems, the network formation can overcome two major limitations of the central place foraging algorithm (CPFA) further: congestion in transportation and the long travel distance between the location of one depot and the locations of multiple clusters. I presented the formation and disbandment of the robot chain network algorithm  $RC_{network}$  and described the modification procedure of subtrees. The experiments (Fig. 33 and Fig. 35) show that  $RC_{network}$  outperforms other two robot chain algorithms with no or one single branch,  $RC_{no}$  and  $RC_{one}$ , respectively.

The reason for the superior performance of robot chain networks is as follows. When there are multiple discovered clusters, one robot chain is insufficient to serve the foraging in multiple clusters. When there are no extended branches, robots need to travel between the multiple clusters and the location of the last robot in the robot chain. Robots collide with each other as in the MPFA with dynamic depots. When there are multiple branches to the clusters, many robots are in the robot chains and branches, and a smaller number of foraging robots travel in the arena. Therefore, the collision time is higher than the collision time in  $RC_{network}$  (Fig. 34 and Fig. 36). Real-time adaptive strategy is a key component of  $RC_{network}$  since it can create multiple branches to connect the multiple clusters. It reduces the travel time of foraging robots and therefore increases the foraging performance.

The foraging performance in the third experiment (Fig. 37) indicates that the frequency of network modification or robot chain's relocation should be selected carefully. The robot chain, either with a high relocation frequency or without relocation, has a low performance. The relocation is efficient, but there is a cost of relocation. There are tradeoffs between the relocation and the cost. The results in the last experiment indicate the exploration time is also critical in the modification. The longer exploration time results in better robot chain networks, but the rate of improvement decreases.

Using dynamic robot chain networks that adapt to the distribution of resources,  $RC_{network}$  is an effective method that reduces bottlenecks in robot chains and increases foraging performance. This study demonstrates that dynamic robot chain networks may significantly enhance the transportation performance of foraging swarm robotics systems, hence enhancing the performance of foraging. This unique system has the following advantages: 1) it offers more efficient transportation than current no branch or single branch robot chain algorithms; 2) it reduces robot collision time; and 3) it is adaptable to local circumstances.

## VII Mobile Workstations

Many existing mobile robots can only accomplish their specified tasks while they are immobile. The productivity of these robots may be enhanced if they are able to accomplish certain activities while moving. In this section, I present mobile workstations, which integrate movable platforms with manufacturing gear to boost efficiency by overlapping production and delivery time. I offer a model of mobile workstations and their duties and discuss the algorithm for task planning for a group of mobile workstations. The temporal planning challenge for mobile workstations combines the characteristics of traveling salesman problems (TSP) and job shop scheduling problems (JSP), although there is little research that concurrently addresses JSP and TSP.

### 7.1 Introduction

Mobility is vital for many of the duties developed for service robots. Attaching a robot to a mobile platform is the most frequent method of giving robots movement at now. Some mobile platforms enable users to quickly connect any device. A smartphone platform called Patin by Flower Robotics<sup>2</sup>, for instance, enables users to install domestic gadgets such as lighting and air purifiers. However, few researchers have studied adding more advanced gadgets that can conduct production activities while the mobile platform is in motion. For instance, robots in Robocup@Home can only do certain duties when they stop at areas with the relevant appliances. Similarly, Robocup@Work robots are limited to operating machinery at stationary workstations.



Figure 39: A mobile microwave robot.

<sup>2</sup><http://www.flower-robotics.com/patin>



Figure 40: A mobile printer robot.

In this section, I investigate the ramifications of linking production equipment to a mobile platform and the service robot tasks' scheduling . These robots are referred to as "mobile workstations" because they are able to produce or process objects while transferring them. I argue that this mode of operation gives a fresh method for saving considerable time. Consider the scenario shown in Fig. 41, in which you want to get a cup of coffee. If the coffee machine is located far from your office, you may save time by ordering a cup of coffee to be brewed and then having a robot deliver it to you. However, you may save even more time if the robot is outfitted with a coffee maker that can simultaneously brew and deliver a cup of coffee to your office. By overlapping delivery and production time, mobile workstations may save a substantial amount of time and serve more customers.

Mobile workstations extend the notion of mobile coffee-making robots to include any robotic systems with production equipment. For instance, Fig. 39 and Fig. 40 depict the hardware implementation of two distinct mobile workstations, one for printing and the other for microwave cooking. A model of mobile workstations and job planning algorithms are sufficiently inclusive to include all of these devices. The success of these mobile workstations is contingent on the overlap of their production and mobility times to the greatest extent feasible. I investigate the temporal planning issue of *multiple* mobile



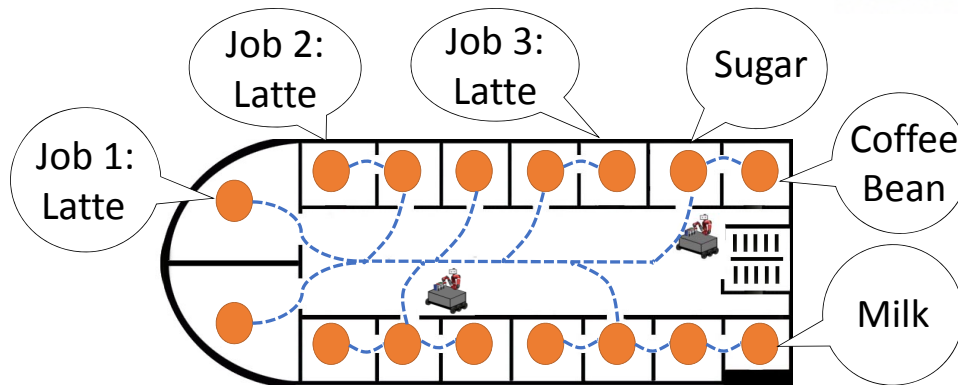


Figure 41: A scenario involving a mobile coffee making robot.

workstations: given  $M$  tasks and  $N$  mobile workstations, how can I allocate the jobs and schedule the activities for the workstations such that the whole group may complete the jobs in the shortest period of time? Minimizing the makespan makes sense, particularly when many mobile workstations may operate in tandem. This system will execute the planning algorithm whenever a task is added to or deleted from the work queue to ensure that optimality is maintained.

This planning problem is NP-hard because it encompasses two challenging NP-hard problems: the traveling salesman problem (TSP) and the job shop scheduling problem (JSP). The TSP and the JSP are well-known problems for which a considerable amount of literature exists. However, the planning and scheduling literature has few studies that mainly address JSP and TSP simultaneously inside a single framework.

I solve this problem by developing a novel algorithm that searches the space of task graphs whose nodes represent movement and production actions. After locating the optimal task graph with the shortest critical path, the method converts the task graph into a temporal plan. As with many temporal planning problems, a thorough approach for constructing an optimal plan would run in exponential time (unless  $P = NP$ ) and cannot reasonably address situations requiring more than a dozen activities. The second method does a local search in the space of task graphs in order to fast generate a suboptimal temporal plan for a large number of tasks. Experiments were undertaken to compare suggested algorithms with SGPlan 5 [41, 126], one of the finest domain-independent temporal planners that employs Planning Domain Definition Language (PDDL) [127], which is expressive enough to specify the issue. There is no domain-dependent planner capable of solving the issue, however I compared suggested algorithm with the Google Optimization Tools's JSP solver<sup>3</sup> using a collection of test cases that the JSP solver can solve. The experimental findings demonstrate that the suggested local search method beats both Google's JSP solver and SGPlan 5. Moreover, when the problem size is small, the algorithm may construct temporal plans with makespans that are comparable to those of optimum solutions.

<sup>3</sup>[https://developers.google.com/optimization/scheduling/job\\_shop](https://developers.google.com/optimization/scheduling/job_shop)

## 7.2 Mobile Microwave Robots

To illustrate the concept of mobile workstations, I built a mobile microwave robot, as shown in Fig. 39. I used Clearpath’s Husky vehicle as to the mobile platform and put a 1100W microwave oven on top of it. The oven is powered by an uninterruptible power supply (UPS) unit hidden in the compartment of the vehicle. I installed a Kinova JACO<sup>2</sup> 6DOF robot arm with three fingers on the vehicle, which was programmed to open the door of the oven, put an object in the oven, press the on/of buttons of the oven, and take an object out of the oven. Due to the size and the structure of the robot arm, the robot can only handle tall containers such as coffee mugs. The robot used a Hagisonic StarGazer Robot Localization sensor for localization in an indoor environment. To facilitate the movement of the mobile platform and avoid collisions, I installed an ASUS Xtion PRO LIVE RGB and depth sensor on it. A webcam is installed for object manipulations. One limitation of the current implementation of the robot is that the robot arm can only grab things from certain predefined locations. In the future, I intend to program the robot arm to grip things directly from a refrigerator.

The robot is operated by two Intel NUCs running ROS under Ubuntu Linux operating systems. Upon receiving high-command commands such as going to a room to grab a cup, the robot will carry out the actions autonomously. The scheduling algorithm is run on a desktop computer which can remotely send commands to the robot to control it. I implemented a simple user interface for users to check the current temporal plan and monitor the current state of the robot. Through the user interface, a user can submit a job to the system, and then the scheduling algorithm will regenerate the temporal plan.

## 7.3 The Scheduling Problem for a Team of Mobile Workstations

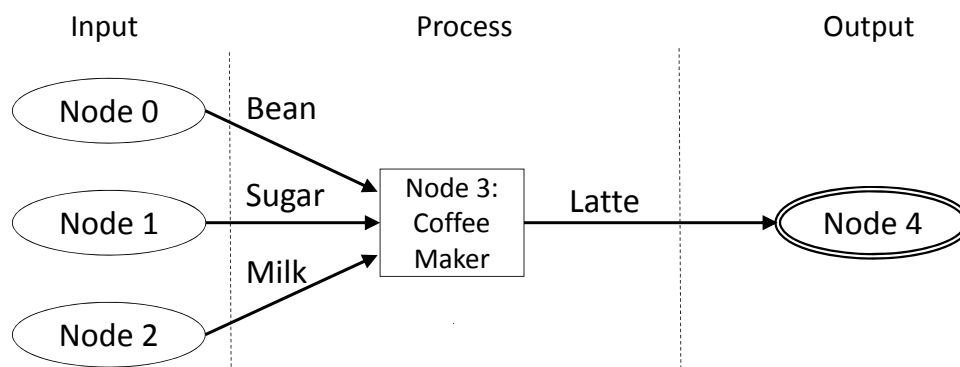


Figure 42: A model of the mobile coffee making robot.

To generalize the concept of mobile microwave robots, I define a model of mobile workstations as follows. A model of a mobile workstation as a directed tree that consists of  $n_{in}$  input nodes, one process node, and one output node, where  $n_{in} \geq 0$ . Hence, the total number of nodes is  $n = n_{in} + 2$ . The input nodes represent the hardware for gathering raw materials, the process node represents the machinery for production, and the output node represents the hardware for delivery of the final product. Fig. 42 shows the model of the mobile coffee making robot. In this model, there are three input nodes, which present the hardware for fetching the raw materials for coffee making. These input nodes connect to a process



node which represents a coffee maker. Likewise, the process node connects to the output node, which represents a coffee maker. This model captures the essential hardware components on a mobile coffee making robot. The edges in Fig. 42 represent the possible flow of objects inside a workstation. The incoming edges of the process node represent *prerequisites* for the production; the process node must acquire all raw materials from the input nodes before starting the production of the product.

A *job* is a message describing 1) the duration of each task, 2) where to fetch the raw materials, and 3) where to deliver the final product at each node in the model. Fig. 43 shows three jobs for a mobile coffee making robot. Each node in a job represents a task that a workstation needs to perform at the corresponding node in the model of the workstation. The number in a node of a job represents the *duration* of the task represented by the node, whereas the location in a node of a job represents the location at which the workstation must carry out the task. For instance, in Job 1 the mobile coffee making robot spends 30s at Room 6 to get some coffee bean, 40s at Room 7 to get some sugar, and 20s at Room 15 to get some milk. When the robot has three materials, it takes 250 seconds to make a latte. The robot took 30s to finish the task of delivering a latte at Room 0. As can be seen, the time of same nodes in various tasks might vary; this variance helps us to express the distinction between jobs. For instance, Job3 may have demanded a big latte, requiring the robot to spend additional time preparing coffee. Notice that when a workstation grabs an object for an input node or deliver the final product for an output node, the workstation must stay at the location specified by the node until the task is finished.

A dedicated server processes all job requests. After receiving a task, the server will place it in a *job queue*. When the *job queue* is modified by the addition or removal of jobs, a scheduling algorithm will be executed to allocate the jobs to various workstations and update the existing temporal plans for all mobile workstations. Consider a scenario in which there are  $N$  mobile workstations and  $M$  jobs in the *job queue*. The scheduling method creates a temporal plan consisting of a number of *task actions* and *move actions*. There are three kinds of task actions:

- $\text{Fetch}(w, o, l, d)$ —the workstation  $w$  gets an object  $o$  at a location  $l$ ;
- $\text{Process}(w, o, o_1, o_2, \dots, o_n, d)$ —the workstation  $w$  makes an object  $o$  from the object  $o_1, o_2, \dots, o_n$ ;
- $\text{Deliver}(w, o, l, d)$ —the workstation  $w$  delivers an object  $o$  at a location  $l$ ; and

But there is only one kind of move actions:

- $\text{Move}(w, l, d)$ —the workstation  $w$  moves to location  $l$ .

Formally,  $\pi$  is a list of  $\langle t, a \rangle$  where  $t$  is the time to execute the action  $a$ . A *valid* temporal plan must satisfy the following constraints:

- There is one task action for each node for each job;
- There is one move action for each input or output node for each job;
- All actions assigned to a job must also be assigned to the same workstation;

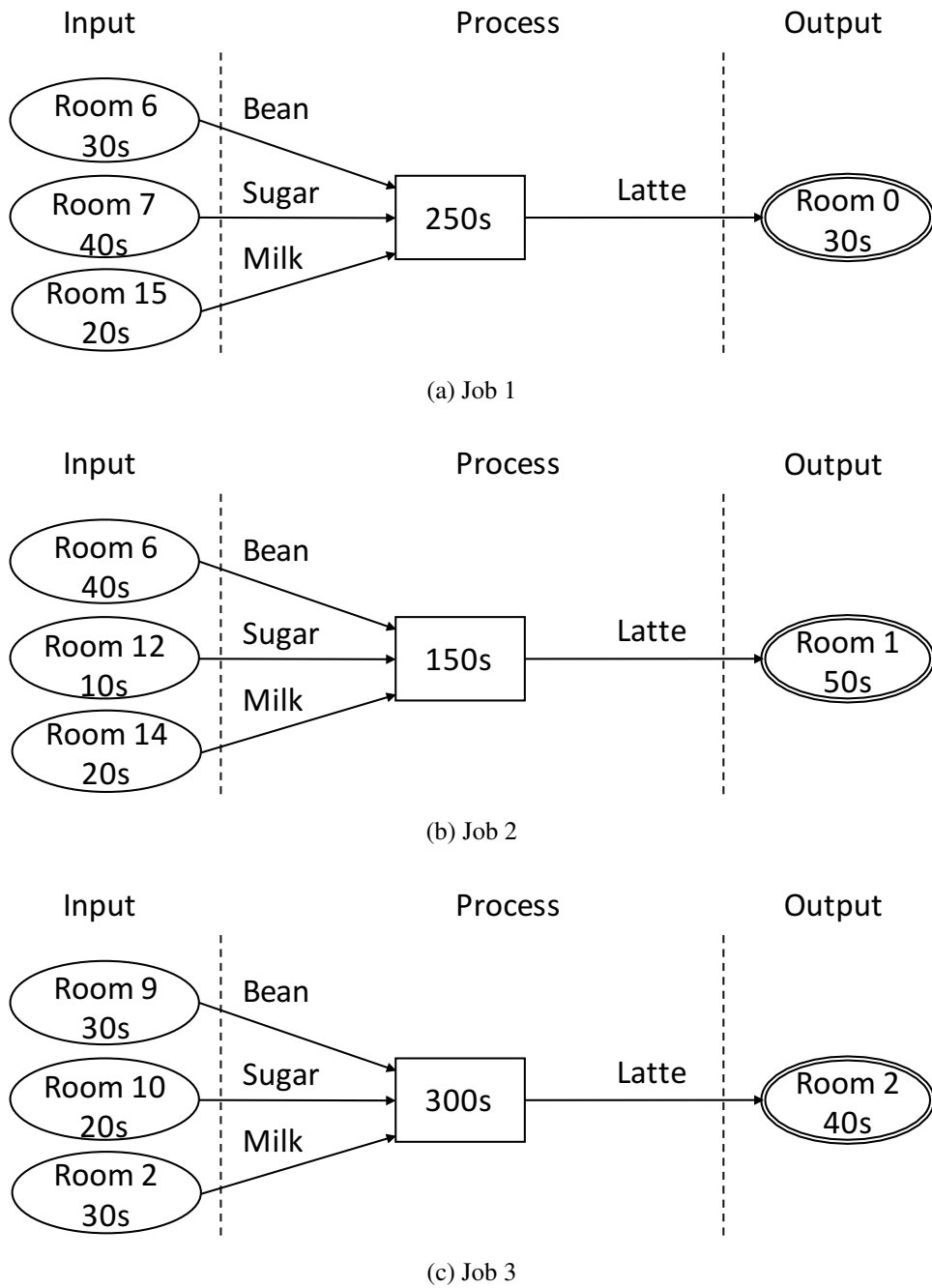


Figure 43: The example of three mobile coffee making robot's jobs.

- The starting time of a task action  $a_1$  must not be earlier than the finish time of the task action  $a_2$  that is a predecessor of  $a_1$  in the job;
- There should be no move action whose time interval overlaps with the interval for any input and output nodes of the same workstation;
- For each task action for an input or output node, there should be a move action whose location and workstation are the same as the workstation and location of the task action. Moreover, the finish time of the move action should occur before the start time of the task action, and there is no other move action of the same workstation between the finish time and the start time; and
- The time intervals for the task actions for the same node and the same workstation cannot overlap each other.

In each of these actions,  $d$  represents the maximum duration for performing the actions to complete the task.

The schedule of these actions is a temporal plan  $\pi$  of mobile workstations.  $\pi$  is formally a list of pairs, each of which is  $(t, a)$  where  $a$  represents the execution of the action at the start of  $t$ . When there are two mobile workstations, the plan for jobs in Figure 43 is shown in Figure 45. Some parameters of this plan's activities are excluded because they may be deduced from Fig. 43. The makespan of this plan for these jobs is 610s. This temporal plan is *optimal*, since there is no alternative temporal plan with a shorter makespan than 610s.

## 7.4 Planning Algorithms

Given  $M$  jobs and  $N$  workstations I compute a temporal plan as a minimum the makespan plan. Even with a modest number of tasks, a forward exhaustive search method is not efficient enough to identify a plan. Consequently, I develop a novel graph search method based on a task graph  $G = (V, E)$ , where each vertex in  $V$  represents a job and each edge in  $E$  indicates the temporal ordering of two tasks. A  $(v_1, v_2) \in E$  edge indicates that the job at  $v_1$  must complete before the task at  $v_2$ . When there is no edge connecting two vertices, the tasks may be performed in parallel since there is no indirect reliance between them. Fig. 44 depicts the task graph for the three tasks seen in Fig. 43, whose output nodes are linked to a new diamond-shaped vertex. A *supertree* is a subtree represented in Fig. 44 by the black arrows.

From a supertree, a task graph may be created by adding (1) exclusive-task edges and (2) movement vertices and movement edges. Since there are several methods to add these edges and vertices (see below), a supertree may be used to create numerous task graphs. A task graph's critical path is the longest path from the root to a vertex, where the length of a path is the total of the maximum duration of all vertices along the path. Then, I schedule a set of project tasks using the critical path approach (CPM) [128]: Find the task graph with the shortest critical route among all feasible graphs formed from a supertree. This idea focuses on efficiently enumerating all potential task graphs.

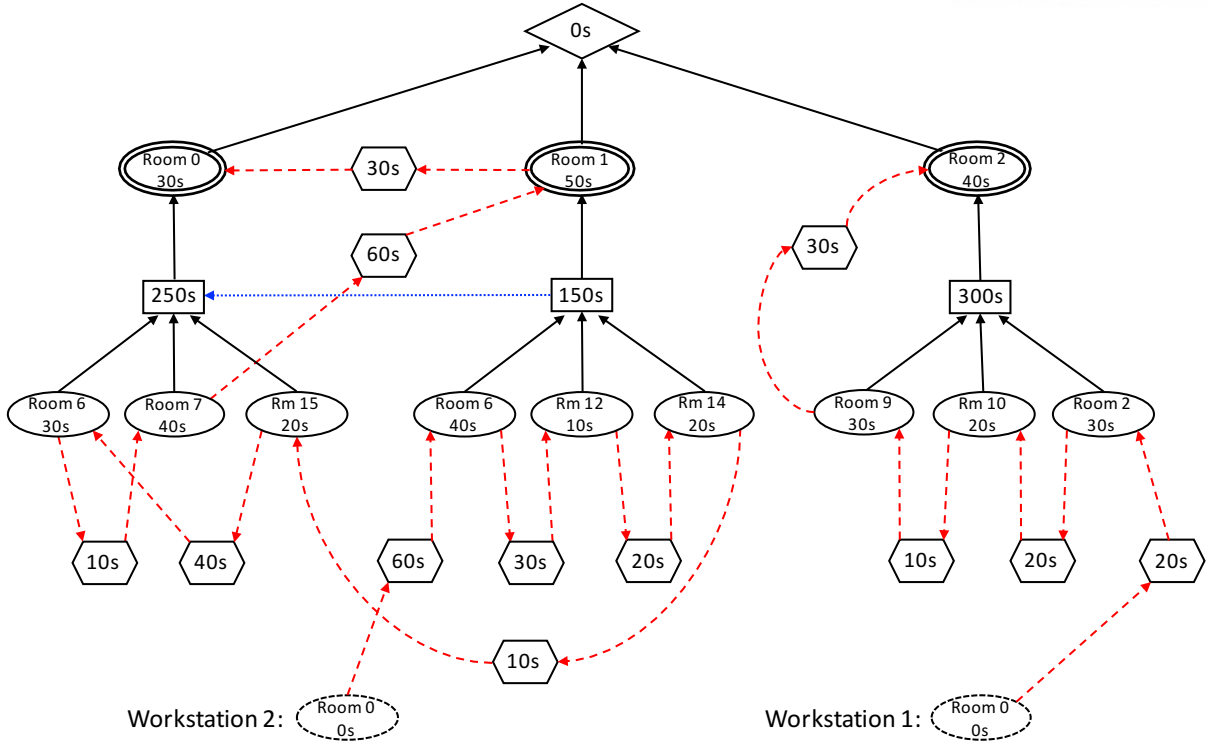


Figure 44: A task graph for the jobs in Figure 43

**Exclusive-Task Edges:** Consider a supertree  $T$  constructed by combining a set of  $M$  jobs:  $J_1, J_2, \dots, J_M$ . the algorithm identified the vertex of a node  $i$  in a job  $J_j$  by  $(i, j)$ . First, the algorithm divide the set of jobs into  $N$  partitions:  $\rho^0 = \{\mathbb{J}_1, \mathbb{J}_2, \dots, \mathbb{J}_N\}$ , such that all jobs in  $\mathbb{J}_k$  are allocated to the workstation  $k$ . In each partition  $\mathbb{J}_k$ , I add a set of edges to  $T$  to indicate the constraints that a process node of the workstation  $k$  cannot concurrently complete two distinct tasks. These mutually exclusive requirements may be expressed using a collection of directed edges linking the vertices  $(i, j)$  for each process node  $i$  and for each  $J_j \in \mathbb{J}_k$ . For each process node  $i$ , let  $\rho_{(i,k)}^1 = (j_1, j_2, \dots, j_{|\mathbb{J}_k|})$  be a *permutation* of the indexes of the jobs in  $\mathbb{J}_k$ . Then the algorithm add an *exclusive-task* edge between  $(i, j_x)$  and  $(i, j_{x+1})$ , for  $1 \leq x < |\mathbb{J}_k|$ . In Figure 44, for instance, the blue line represents an exclusive-task edge  $((3,2), (3,1))$  that enforces the ordering that the vertex  $(3,1)$  must be executed after  $(3,2)$ .

**Movement Vertices and Movement Edges:** The algorithm add a set of *movement edges* and *movement vertices* to  $T$  to represent the route of the mobile workstation  $k$ . In  $T$ , all output and the input node vertices for the partition  $\mathbb{J}_k$  are connected with the places the workstation  $k$  should visit. In addition, the workstation must stay at these locations for a given duration to deliver or fetch objects. Let  $\{v_1, v_2, \dots, v_r\}$  represent the set of all output and input node vertices for the jobs in  $\mathbb{J}_k$ ,  $l_x$  is the location of  $v_x$ , if  $1 \leq x \leq r$ , and  $l_0$  is the location of the workstation  $k$  initially. Given a permutation  $\rho_k^2 = (s_1, s_2, \dots, s_r)$  of the first  $r$  positive integers, I add a new vertex  $v_{(l_{s_i}, l_{s_{i+1}})}^{\text{move}}$  between  $v_{s_i}$  and  $v_{s_{i+1}}$ , for  $1 \leq s < r$ , and vertex  $v_{(l_0, l_{s_1}}^{\text{move}}$  and an edge  $(v_{(l_0, l_{s_1}}^{\text{move}}, v_{s_1})$  as well as the edges  $(v_{s_i}, v_{(l_{s_i}, l_{s_{i+1}})}^{\text{move}})$  and  $(v_{(l_{s_i}, l_{s_{i+1}})}^{\text{move}}, v_{s_{i+1}})$  to  $T$ . These movement vertices and movement edges are shown in Fig. 44 by the

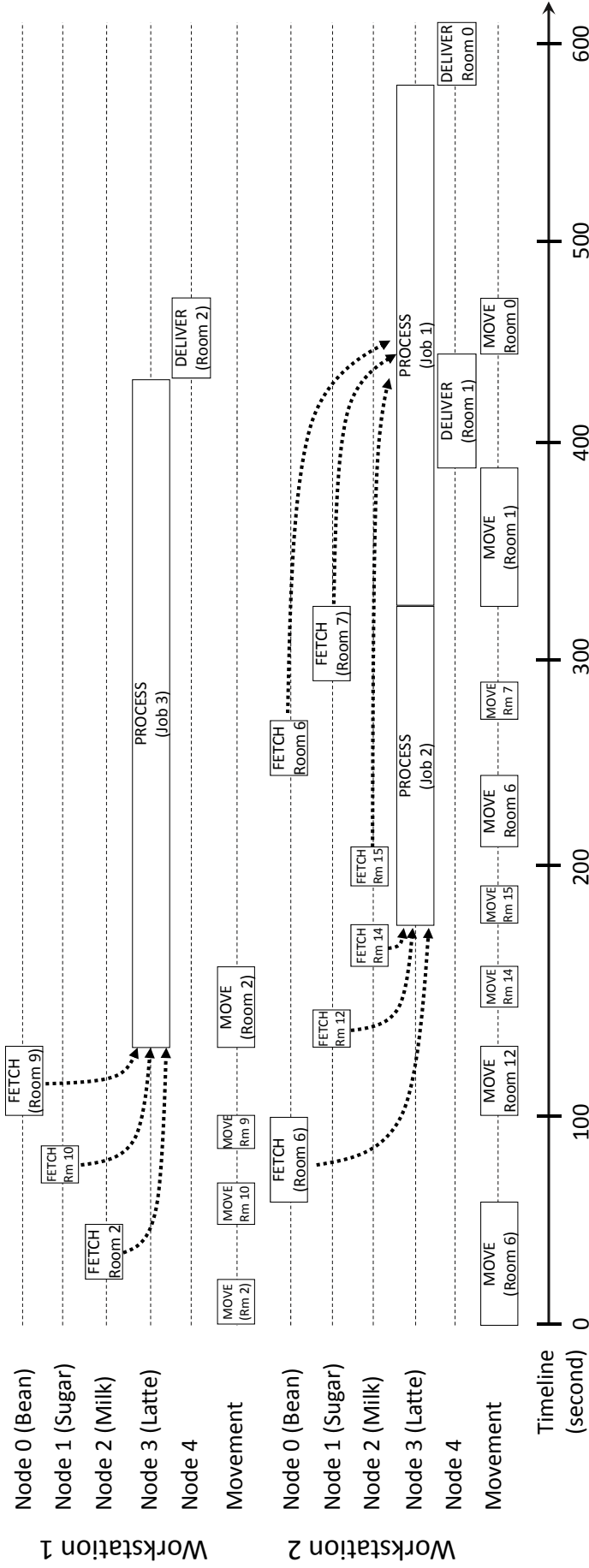


Figure 45: An optimal temporal plan for the three jobs in Figure 43.

hexagons and the red edges, respectively. The maximum duration of a movement vertex  $v_{(l_{s_i}, l_{s_{i+1}})}^{\text{move}}$  is the amount of time it takes for the robot to go from  $l_{s_i}$  to  $l_{s_{i+1}}$ .

---

**Algorithm 2** The Multiagent Scheduling Algorithm.

---

```

1: procedure COMPLETESEARCH( $J_1, J_2, \dots, J_M$ )
2:   Construct a superjob  $T$  from  $J_1, J_2, \dots, J_M$ .
3:   Let  $D_{\min}$  be a large number.
4:   Let  $S_{\min} = \{\}$ 
5:   for each possible job assignment  $u$  to the workstations do
6:     Let  $\mathbb{J}_w$  be the jobs assigned to  $w$  according to  $u$ .
7:     Let  $V_{\text{IO}}^w$  be the vertices for all input and output
       nodes of the jobs in  $\mathbb{J}_w$ .
8:     Let  $H_w$  be the set of process nodes in the jobs in  $\mathbb{J}_w$ 
9:     for each permutation  $\rho_w^1$  of  $H_w$  do
10:      Add new edges to  $T$  according to  $\rho_w^1$ .
11:      for each topological ordering  $\rho_w^2$  in  $V_{\text{IO}}^w$  do
12:        Add vertices and edges to  $T$  according to  $\rho_w^2$ .
13:        if all  $(\rho_w^1, \rho_w^2)$  are assigned for  $1 \leq w \leq N$  then
14:           $S := \{ \{ \rho_w^1, \rho_w^2 \} \}_{1 \leq w \leq N}$ 
15:          Compute the length  $D$  of the critical path
16:          If  $D < D_{\min}$  then  $D_{\min} := D$ ;  $S_{\min} = S$ 
17:        end if
18:        Remove the vertices and edges created for  $\rho_w^2$ .
19:      end for
20:      Remove the edges created for  $\rho_w^1$ .
21:    end for
22:  end for
23:  Add vertices and edges to  $T$  according to  $S_{\min}$ .
24:  return the temporal plan  $\pi$  extracted from  $T$ .
25: end procedure

```

---

A complete search algorithm computes the shortest critical path by enumerating all feasible tuples of the form  $(\rho^0, \{(\rho_{(i,k)}^1, \rho_k^2)\}_{1 \leq k \leq N})$ , where  $T$  is the set of all new vertices and edges. The algorithm use a depth-first search to determine the length of the critical path for each tuple. Then, the algorithm determine which pair provides the shortest critical route. Note that  $\rho_k^2$  must be a *topological ordering* ordering in order to preserve the temporal dependencies between input and output nodes. This enumeration algorithm has a running time of  $O(M^N \times M! \times r! \times (|V| + |E|))$ , where  $N$  is the number of workstations,  $M$  is the number of jobs, and  $r = |V_{\text{IO}}|$  is the total number of output and input in the supertree.

Due to its exponential running time, this complete search method is impractical despite the fact that it guaranteed to produce the optimum result. In fact, similar to TSPs, these problems are NP-hard, meaning that it is unlikely that any algorithm will return an optimal solution to any large problem in a reasonable amount of time. Consequently, I design a local search method that can tackle huge problems rapidly but suboptimally. The algorithm's pseudo-code is presented in in Algorithm 3. Instead of exhaustively

enumerating all cases of movement edges, movement vertices, exclusive-task edges, and job partitions, the local search randomly choose a tuple  $(\rho^0, \{(\rho_{(i,k)}^1, \rho_k^2)\}_{1 \leq k \leq N})$  and modifies it by randomly swapping the vertices in the permutations, in an attempt to improve the solution (Lines 14 and 15).

This random swapping is comparable to the min-conflict heuristics for local search employed to solve the well-known N-Queens problem [129]. Multiple sampling seeks to enhance the answer by repeating the inner loop  $N_{repeat}$  times with various swapping actions, while random restart reruns the local search from  $N_{restart}$  distinct beginning solutions.  $S_{min}$  contains the best solution currently available. Both  $N_{restart}$  and  $N_{repeat}$  are fewer than 5 in the experiments. The algorithm has a running time of  $O(|V| + |E|)$ , therefore it can run extremely quickly and handle several tasks. By delivering a temporal plan based on  $S_{min}$  at any point in time, the algorithm may be converted into an anytime algorithm.

---

**Algorithm 3** The local-search algorithm

---

```

1: procedure LOCALSEARCH( $J_1, J_2, \dots, J_m$ )
2:   Construct a supertree  $T$  from  $J_1, J_2, \dots, J_m$ .
3:   Let  $D_{min}$  be a very large number and let  $S_{min}$  be  $\{\}$ 
4:   Repeat the following  $N_{restart}$  times
5:     Randomly generate a permutation  $\rho^0$  of the jobs.
6:     Repeat the following  $N_{repeat}$  times
7:       Randomly generate permutations  $\rho_{(i,k)}^1$  and  $\rho_k^2$ 
8:        $S := (\rho^0, \{(\rho_{(i,k)}^1, \rho_k^2)\}_{1 \leq k \leq N})$ 
9:       Add exclusive-task edges to  $T$  according to  $\rho_{(i,k)}^1$ 
10:      Add movement vertices and edges to  $T$  w.r.t.  $\rho_k^2$ .
11:      Compute length  $D$  of critical path in  $T$ 
12:      If  $D < D_{min}$  then  $D_{min} := D; S_{min} := S$ 
13:      Remove the exclusive-task edges and the
14:      movement vertices/edges from  $T$ 
15:      Randomly modify  $\rho_k^2$  by swapping without
16:      violating the topological order.
17:      Randomly modify  $\rho_{(i,k)}^1$  by swapping.
18:      Add vertices and edges to  $T$  according to  $S_{min}$ .
19:   return the temporal plan  $\pi$  extracted from  $T$ .
20: end procedure

```

---

## 7.5 Experimental Evaluation

As demonstrated in Fig. 39 and Fig. 40, I developed two robots that function as mobile workstations. In addition, I created a web-based user interface for task submission and mobile workstation monitoring. These robots perform as expected on one level of the university building. The system also addresses several practical concerns, such as rerouting when a human user frees a robot that has been stranded.

The assessment of suggested algorithms is mostly focused on simulation since it enables the testing of several robots operating concurrently in a much bigger region. I compared suggested algorithms to SGPlan 5, one of the most effective temporal planners [41, 126]. In addition, I compared suggested



methods with (1) a forward search algorithm that constructs a temporal plan by exhaustively assigning tasks to nodes in increasing time and (2) a full graph search algorithm that enumerates all potential task graphs, as described in Section 7.4.

I developed a Java simulator and executed the trials on a Linux cluster with 20 nodes, 120 cores, and 4GHz Intel Core i7 processors. I randomly produced six maps based on the floor layouts that are typical in office and hotel contexts. The time required to go between two neighboring points on a map is proportional to the distance between those sites plus a Gaussian error factor, which indicates the variation in timing along various edges. I suppose the robots are sufficiently tiny that they will not collide and get stranded on the same route.

Initially, I did an experiment using a mobile workstation with up to forty randomly generated nodes (including input, process, and output nodes). I produced 120 issues with varying amounts of tasks at random. The duration of a node is a random number ranging from 1s to 100s. I categorized issue instances based on the total number of nodes and assessed the running time and plan length of the algorithms' solutions. The outcome is shown in Figures in Fig. 46 and Fig. 47.. The error bars in these statistics represent confidence ranges of 95 percent. Each data point in the graph represents the mean of 120 readings.

I set the time limit of 30 minutes so that if an algorithm cannot deliver a solution within the time limit, I declare that it fails to solve the instance of the issue. As seen in Fig. 46 and Fig. 47, the forward search algorithm and the graph search algorithm cannot solve any issue when the number of nodes exceeds 18, however the local search algorithm and SGPlan 5 can solve all problems in a few seconds. Nonetheless, the quality of plans created by the proposed local search algorithm is much superior to that of SGPlan (see Fig. 47). When the number of nodes is fewer than or equal to 14, the local search algorithm may create almost identically long plans to those generated by the forward search algorithm and the graph search algorithm. As the plan length develops slowly until the number of nodes reaches 40 (see Fig. 47), I conclude that the local search algorithm's plans are close to optimum when the number of nodes exceeds 14. However, I am uncertain as to whether the solutions will be near to optimum for much bigger issues.

In conclusion, I ran an experiment to compare the suggested method to a job shop scheduling algorithm. However, there is no current JSP solver capable of handling movement activities as do the suggested algorithm. I designed a set of test problems in which all movement actions have the same duration if they travel to the same area in order to facilitate fair comparisons. This enables us to add the time of movement to the duration of each task in a job so that a JSP solver can automatically account for the cost of movement. Moreover, if two consecutive actions in a plan created by a JSP solver occur at the same place, I modify the plan to eliminate the duplication of movement time. This experiment's parameters were identical as those for SGPlan 5, but I utilized the JSP solver in Google Optimization Tools. The results are shown in Fig. 48 and Fig. 49. Compared to Fig. 46 and Fig. 47, the JSP solver outperformed SGPlan 5 due to the fact that the JSP solver did not need to solve the TSP issue encoded in the problem, but SGPlan 5 did. However, the JSP solver did not perform as well as the suggested local search algorithm in terms of both plan length and execution time.

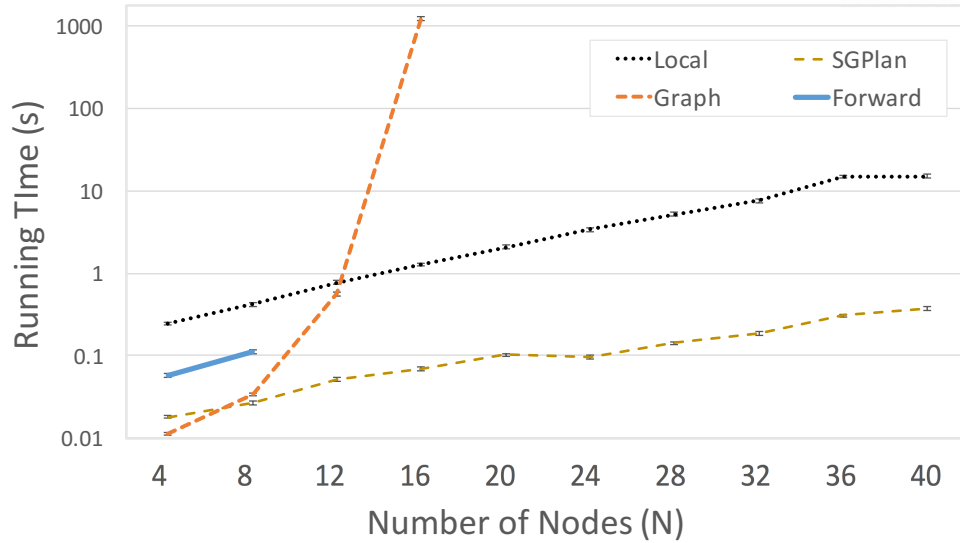


Figure 46: Running time according to different numbers of nodes

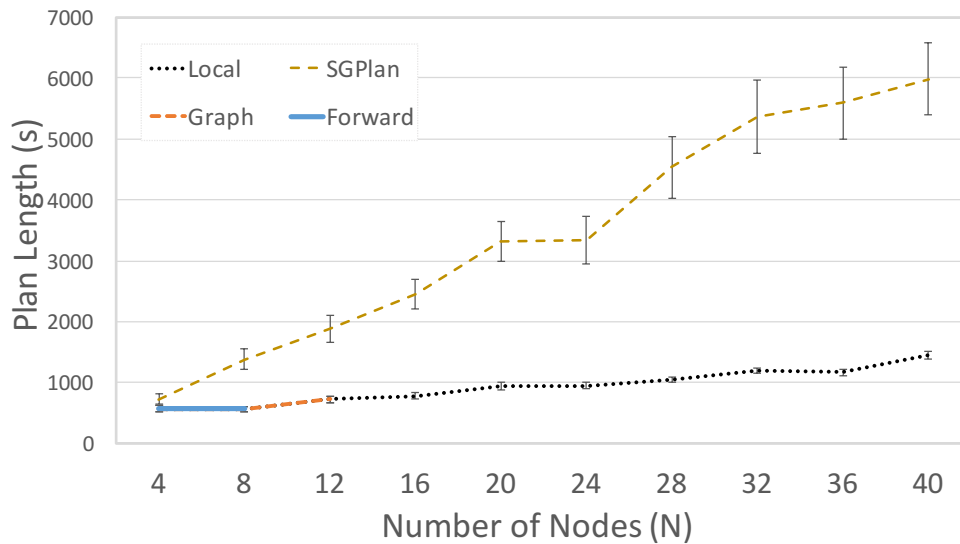


Figure 47: Plan length according to different numbers of nodes

## 7.6 Summary

Combining a mobile platform with production machinery, a mobile workstation increases productivity by overlapping production time and delivery time. In this section, I provided a comprehensive, optimum, temporal planning algorithm for mobile workstations and outlined a realistic, but suboptimal, local-search method that enables a small number of mobile workstations to execute a large number of tasks. Experiments in Section 7.5 demonstrate that the suggested local search algorithm surpasses Google's JSP solver and one of the finest temporal planners, SGPlan5, in terms of solution quality and execution time. Currently, the algorithm simply accounts for temporal variations in low-level motion control by planning for the maximum length of actions and motions. In the future, I hope to include motion planning in the proposed method in order to better optimize the route by considering the exact time of low-level control.

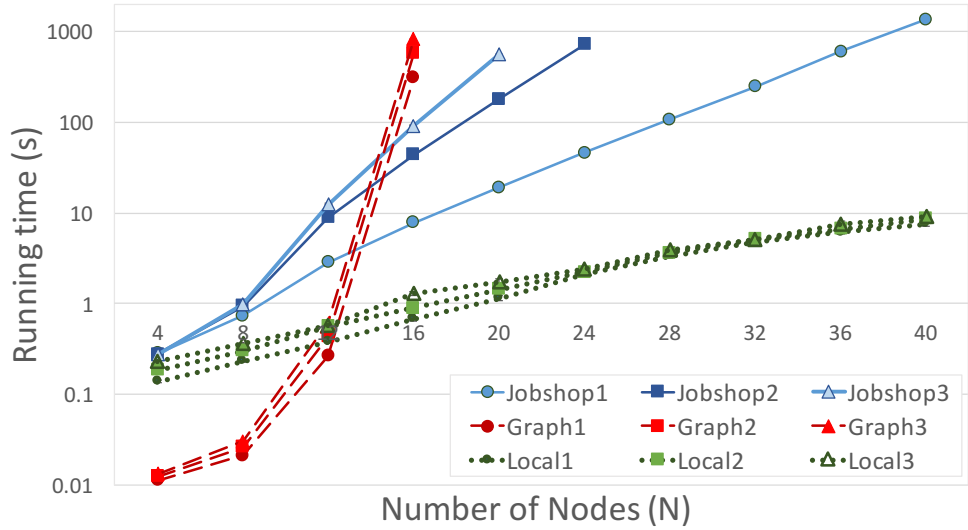


Figure 48: Running times according to different numbers of nodes

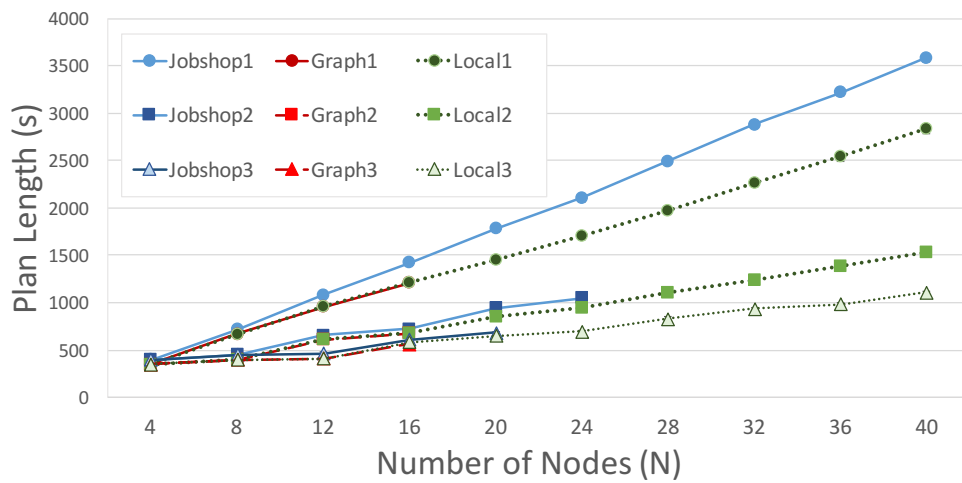


Figure 49: Plan length according to different numbers of nodes

## VIII Conclusion and Future work

This thesis presented an improving robot team's performance by passing objects between robots, including 1) mobile conveyor belts, 2) dynamic robot chains and 3) mobile workstations. First, a new conveyor system called mobile conveyor lines can organize itself independently to transport goods to a specified location. In Section IV, I discussed the incorporation of mobility into conveyor belts and how to link several mobile conveyor belts to construct a conveyor line that reaches a predetermined destination. Key findings include a comprehensive set of equations describing the reachable set of a movable conveyor belt on a flat surface, which leads to an effective probabilistic strategy for autonomous setup. I provided a design for mobile conveyor belts and examined the mobile conveyor belts' accessibility. The algorithm for the configuration determines if a particular destination is reachable and a method for generating a configuration for connecting conveyor belts to reach the destination. The experimental findings shown in Section 4.7 demonstrate that suggested configuration algorithms, in conjunction with a heuristic that biases the search towards the target, may rapidly develop conveyor belt designs for challenges. Conveyor belts are good at moving a lot of things at once, so they could be a big part of any robotic system used for logistics or rescue operations.

Second, the goal of foraging tasks in robotics is to quickly find for materials and bring them to a central place for collecting. For the previous multiple-place foraging algorithm with dynamic depots, foraging effectiveness diminishes as search regions and swarm numbers increase: depots must travel long distances to transport supplies to the center, and more robots result in increased congestion on their excursions. I adapted the robot chain to a foraging task in which a team of robots collects materials in collaboration. The key is the construction of robot networks that can overcome the two fundamental constraints of current dynamic depot foraging systems: congestion around the central collecting zone and large delivery distances. The robots have the capacity to exchange resources at a distance. Each robot chain links a place for foraging to the central collecting zone. Instead of distributing resources through a single robot, supplies are transferred via robot chains from foraging places straight to the center, thereby reducing congestion in the vicinity of the central collecting zone. In Section 5.4, I discuss a robot-controlling algorithm and the relocation method of dynamic robot chains in order to get closer to the resources while avoiding obstacles.

In addition, I introduce robot chain networks, an extension to dynamic robot chains that extends robot chains with branches that each reach distinct resource clusters. I formulate the challenge of locating the shortest robot chain networks as the Euclidean Steiner tree problem and describe how Steiner trees may be used to enhance the efficiency of foraging operations. Using the robot simulator ARGoS, I simulated robot swarms. Experiments in Section 5.6 demonstrate that robots utilizing the MPFA with dynamic chains outperform those using dynamic depots and experience less congestion. Experiments in 6.7 demonstrate that robots using robot chain networks can avoid obstacles and gather more resources than the original robot chain design.

Third, many present mobile service robots can only carry out their specific duties while they are immobile. The productivity of these robots may be enhanced if they are able to accomplish certain ac-

tivities while moving. I presented the notion of mobile workstations, which integrate mobile platforms with manufacturing equipment to boost productivity by overlapping delivery and production time. I offer a model of mobile workstations and their duties and discuss the algorithm for task planning for a mobile workstation team. The temporal planning challenge for mobile workstations combines the characteristics of traveling salesman problems (TSP) and job shop scheduling problems (JSP), although there is little research that concurrently addresses TSP and JSP. In Section 7.4, I presented three algorithms as complete, optimal, temporal planning algorithms for mobile workstations, and described a practical but sub-optimal local-search algorithm that allows a small number of mobile workstations to handle a lot of jobs. A mobile conveyor belt and a mobile workstation demonstrate how to improve the performance in applications involving passing objects.

Based on the results so far, I would like to do some more research in the future. Regarding the mobile conveyor belt, currently, I presented how to form a conveyor line with mobile conveyor belts. But the work done so far didn't think about moving mobile conveyor belts into the right place to make a conveyor line. The shape of the robot changes depending on the pitch angle of the conveyor parts. The mobile conveyor belts need to link each other. Therefore, only a few multiple agent motion plans are workable for this problem. However, I expect many mobile conveyor belts will have a similar path or overlapping paths. In the future, we can use these features to make a better algorithm for mobile conveyor belts to plan their movements. In addition, current mobile conveyor belts are adapted for constructing conveyor lines on flat ground. However, in the real world, the floor is not always flat. I want to study connecting two desired points via a conveyor line using a moving conveyor belt in a non-flat environment, such as a rescue mission. The algorithm would be more complex and would have to consider more scenarios for connecting the mobile conveyor belts.

The algorithm also checks whether the place to dispatch mobile conveyor belts can arrive and whether a linked mobile conveyor can safely move objects to the next mobile conveyor belt without falling. These studies will extend current research in the direction of how to use the existing mobile conveyor belt in the real world. Seconds is an extension of research on the foraging problem. I applied a concept of mobile conveyor belts to the foraging problem to establish a robot-chain that has higher performance than existing algorithms. One of the key achievements is reducing congestion around the central depot by distributing the robots moving to the center depot across a chain of robots. Nowadays, in swarm robotics, many researchers want to build systems based on robots with finite energy (battery). Each robot has a limited amount of power and needs to be charged before it can go to the charging spot.

It means that the charging location will also be congested just like a central depot. In these cases, if the robot chain could transfer the energy to the other foraging robots, it would reduce the congestion and the energy consumption for visiting the charging location. In addition, by using a helper robot that can transmit energy to other robots, the system prevents the possibility that the robot's energy will become zero like a dead robot. In operating the helper robot, the key points will be how to schedule requests and manage energy according to requests. The third step is to expand research into mobile workstations. Existing mobile workstations use a single robot to collect materials, process items, and deliver them. In the real world, if we want to make a mobile workstation, the size of the robot can

become very large depending on the task. Because mobile workstations need to carry all machines on the mobile platform. The task can be split into several sub-tasks based on the process machines. Each mobile workstation can handle sub-tasks, which are a part of the whole process. The system requires more mobile workstations to build and deliver a single product, but it will be able to perform tasks in parallel with multiple requests.

## References

- [1] A. Drogoul and J. Ferber, “From tom thumb to the dockers:some experiments with foraging robots,” *From Animals to Animats*, vol. 2, p. 451, 1993.
- [2] J. P. Hecker, J. C. Carmichael, and M. E. Moses, “Exploiting clusters for complete resource collection in biologically-inspired robot swarms,” in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 434–440.
- [3] L. Guangbu and L. Ruqiong, “Belt conveyor modeling and performance simulation based on amesim,” in *2009 Second International Conference on Information and Computing Science*, vol. 4. IEEE, 2009, pp. 304–307.
- [4] B. Peter, G. Anna, and R. Ivica, “3d mathematical model of conveyor belt subjected to a stress loading,” *Podzemni radovi*, vol. 13, pp. 179–188, 2006.
- [5] D. Pitcher, “Joining conveyor belting,” *Bulk Handling Today*, pp. 17–20, 2005.
- [6] A. J. G. Nuttall, “Design aspects of multiple driven belt conveyors,” 2007.
- [7] V. Donis, A. Rachkovskii, and V. Sin, “How the conveyor belt length affects belt weigher accuracy,” *Measurement techniques*, vol. 47, no. 2, pp. 163–167, 2004.
- [8] T. Collins and W.-M. Shen, “Paso: an integrated, scalable pso-based optimization framework for hyper-redundant manipulator path planning and inverse kinematics,” *Information Sciences Institute Technical Report*, 2016.
- [9] K. Mankge *et al.*, “A simulation approach to constraints management of an underground conveyor system,” Ph.D. dissertation, 2014.
- [10] R. L. MCNEARNY and Z. NIE, “Simulation of a conveyor belt network at an underground coal mine,” *Mineral Resources Engineering*, vol. 9, no. 03, pp. 343–355, 2000.
- [11] Y.-f. Hou and Q.-r. Meng, “Dynamic characteristics of conveyor belts,” *Journal of China University of Mining and Technology*, vol. 18, no. 4, pp. 629–633, 2008.
- [12] K. N. S. Ananth, V. Rakesh, and P. K. Visweswarao, “Design and selecting the proper conveyor-belt,” *International Journal of Advanced Engineering Technology*, vol. 4, no. 2, pp. 43–49, 2013.



- [13] B. KARolewSKI and P. Ligocki, “Modelling of long belt conveyors,” *Eksploatacja i Niezawodność*, vol. 16, no. 2, 2014.
- [14] S. Zhang and X. Xia, “Modeling and energy efficiency optimization of belt conveyors,” *Applied energy*, vol. 88, no. 9, pp. 3061–3071, 2011.
- [15] D. J. Fonseca, G. Uppal, and T. J. Greene, “A knowledge-based system for conveyor equipment selection,” *Expert systems with applications*, vol. 26, no. 4, pp. 615–623, 2004.
- [16] H. Lauhoff, “Speed control on belt conveyors-does it really save energy?; geschwindigkeit-regelung bei gurtfoerderern-spart es wirklich energie?” *Glueckauf*, vol. 142, 2006.
- [17] W. Liu, A. F. Winfield, J. Sa, J. Chen, and L. Dou, “Towards energy optimization: Emergent task allocation in a swarm of foraging robots,” *Adaptive behavior*, vol. 15, no. 3, pp. 289–305, 2007.
- [18] E. Castello, T. Yamamoto, F. Dalla Libera, W. Liu, A. F. Winfield, Y. Nakamura, and H. Ishiguro, “Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach,” *Swarm Intelligence*, vol. 10, no. 1, pp. 1–31, 2016.
- [19] T. P. Flanagan, K. Letendre, W. R. Burnside, G. M. Fricke, and M. E. Moses, “Quantifying the effect of colony size and food distribution on harvester ant foraging,” *PloS one*, vol. 7, no. 7, p. e39427, 2012.
- [20] G. M. Fricke, J. P. Hecker, A. D. Griego, L. T. Tran, and M. E. Moses, “A distributed deterministic spiral search algorithm for swarms,” *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2016.
- [21] Q. Lu, A. D. Griego, G. M. Fricke, and M. E. Moses, “Comparing physical and simulated performance of a deterministic and a bio-inspired stochastic foraging strategy for robot swarms,” in *Intl. Conf. on Robotics and Automation (ICRA)*, May 2019, pp. 9285–9291.
- [22] Q. Lu, M. E. Moses, and J. P. Hecker, “A scalable and adaptable multiple-place foraging algorithm for ant-inspired robot swarms.” Robotics Science and Systems (RSS) workshop on On-line decision-making in multi-robot coordination, 2016.
- [23] Q. Lu, J. P. Hecker, and M. E. Moses, “The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms,” *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2016.
- [24] T. P. Flanagan, N. M. Pinter-Wollman, M. E. Moses, and D. M. Gordon, “Fast and flexible: Argentine ants recruit from nearby trails,” *PLOS ONE*, vol. 8, no. 8, pp. 1–7, August 2013.
- [25] C. A. Chapman, L. J. Chapman, and R. McLaughlin, “Multiple central place foraging by spider monkeys: Travel consequences of using many sleeping sites,” *Oecologia*, vol. 79, no. 4, pp. 506–511, 1989. [Online]. Available: <http://www.jstor.org/stable/4218988>

- [26] Q. Lu, M. E. Moses, and J. P. Hecker, “Multiple-place swarm foraging with dynamic depots,” *Autonomous Robots*, vol. 42, no. 4, pp. 909–926, 2018.
- [27] E. Ferrante, A. E. Turgut, E. Duéñez-Guzmán, M. Dorigo, and T. Wenseleers, “Evolution of self-organized task specialization in robot swarms,” *PLoS Comput Biol*, vol. 11, no. 8, pp. 1–21, 2015.
- [28] G. Pini, A. Brutschy, A. Scheidler, M. Dorigo, and M. Birattari, “Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer,” *Artificial life*, vol. 20, no. 3, pp. 291–317, 2014.
- [29] D. Lee, Q. Lu, and T.-C. Au, “Multiple-place swarm foraging with dynamic robot chains,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 337–11 342.
- [30] S. Nouyan, A. Campo, and M. Dorigo, “Path formation in a robot swarm,” *Swarm Intelligence*, vol. 2, no. 1, pp. 1–23, 2008.
- [31] P. M. Maxim, W. M. Spears, and D. F. Spears, “Robotic chain formations,” *IFAC Proceedings Volumes*, vol. 42, no. 22, pp. 19–24, 2009.
- [32] F. Wang, P. Liu, S. Zhao, B. M. Chen, S. K. Phang, S. Lai, T. H. Lee, and C. Cai, “Guidance, navigation and control of an unmanned helicopter for automatic cargo transportation,” in *Proceedings of the 33rd chinese control conference*. IEEE, 2014, pp. 1013–1020.
- [33] Dohee Lee and Tsz-Chiu Au, “Automatic configuration of mobile conveyor lines,” *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3841–3846, 2016.
- [34] P. Crescenzi, V. Kann, M. Halldórsson, M. Karpinski, and G. Woeginger, “Minimum geometric steiner tree,” *A Compendium of NP Optimization Problems*, 2000.
- [35] V. Parque and T. Miyashita, “Obstacle-avoiding euclidean steiner trees by n-star bundles,” in *IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 315–319.
- [36] M. Zachariasen and P. Winter, “Obstacle-avoiding euclidean steiner trees in the plane: an exact algorithm,” in *Workshop on Algorithm Engineering and Experimentation*, 1999, pp. 286–299.
- [37] L. Garrote, L. Martins, U. J. Nunes, and M. Zachariasen, “Weighted euclidean steiner trees for disaster-aware network design,” in *International Conference on the Design of Reliable Communication Networks (DRCN)*, 2019, pp. 138–145.
- [38] W. D. Smith, “How to find steiner minimal trees in euclidean d-space,” *Algorithmica*, vol. 7, pp. 137–177, 1992.

- [39] H. Miyashita, T. Yamawaki, and M. Yashima, “Control for throwing manipulation by one joint robot,” in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1273–1278.
- [40] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.
- [41] Y. Chen, C.-W. Hsu, and B. W. Wah, “Sgplan: Subgoal partitioning and resolution in planning,” *Edelkamp et al. (Edelkamp, Hoffmann, Littman, & Younes, 2004)*, 2004.
- [42] Y. Chen, B. W. Wah, and C. Hsu, “Temporal planning using subgoal partitioning and resolution in sgplan,” *Journal of Artificial Intelligence Research*, vol. 26, pp. 323–369, 2006.
- [43] M. Fox and D. Long, “Pddl2. 1: An extension to pddl for expressing temporal planning domains,” *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.
- [44] J. C. Beck, P. Prosser, and E. Selensky, “Vehicle routing and job shop scheduling: What’s the difference?” in *ICAPS*, 2003, pp. 267–276.
- [45] P. Brucker and R. Schlie, “Job-shop scheduling with multi-purpose machines,” *Computing*, vol. 45, no. 4, pp. 369–375, 1990.
- [46] G. Dudek, M. R. Jenkin, E. Milios, and D. Wilkes, “A taxonomy for multi-agent robotics,” *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.
- [47] E. Şahin, “Swarm robotics: From sources of inspiration to domains of application,” in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.
- [48] G. S. S. M. J. M. R. T. Vaughan, K. Støy, “Whistling in the dark: cooperative trail following in uncertain localization space,” *Proceedings of the fourth international conference on Autonomous agents*, pp. 187–197, 2000.
- [49] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić, “Blazing a trail: insect-inspired resource transportation by a robot team,” in *Distributed autonomous robotic systems 4*. Springer, 2000, pp. 111–120.
- [50] S. A. Sadat and R. T. Vaughan, “So-lost-an ant-trail algorithm for multi-robot navigation with active interference reduction.” in *ALIFE*, 2010, pp. 687–693.
- [51] L. Steels, “Cooperation between distributed agents through self-organisation,” in *proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Cambridge, England*. Citeseer, 1990.
- [52] H. Hamann and H. Wörn, “An analytical and spatial model of foraging in a swarm of robots,” in *International workshop on swarm robotics*. Springer, 2006, pp. 43–55.

- [53] J. P. Hecker, K. Letendre, K. Stolleis, D. Washington, and M. E. Moses, “Formica ex machina: ant swarm foraging from physical to virtual and back again,” in *International Conference on Swarm Intelligence*. Springer, 2012, pp. 252–259.
- [54] S. Goss, J.-L. Deneubourg, P. Bourguine, and E. Varela, “Harvesting by a group of robots,” in *Proceedings of the First European Conference on Artificial Life*, 1992, pp. 195–204.
- [55] B. Werger and M. J. Mataric, “Robotic food chains: Externalization of state and program for minimal-agent foraging,” *From Animals to Animats*, vol. 4, pp. 625–634, 1996.
- [56] N. R. Hoff, A. Sagoff, R. J. Wood, and R. Nagpal, “Two foraging algorithms for robot swarms using only local communication,” in *2010 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2010, pp. 123–130.
- [57] N. Hoff, R. Wood, and R. Nagpal, “Distributed colony-level algorithm switching for robot swarm foraging,” in *Distributed autonomous robotic systems*. Springer, 2013, pp. 417–430.
- [58] E. H. Ostergaard, G. S. Sukhatme, and M. J. Matari, “Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks,” in *Proceedings of the fifth international conference on Autonomous agents*, 2001, pp. 29–30.
- [59] D. Goldberg and M. J. Mataric, “Robust behavior-based control for distributed multi-robot collection tasks,” University of Southern California Los Angeles United States, Tech. Rep., 2000.
- [60] P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, “A study of territoriality: The role of critical mass in adaptive task division,” 1996.
- [61] G. Pini, A. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari, “Autonomous task partitioning in robot foraging: an approach based on cost estimation,” *Adaptive behavior*, vol. 21, no. 2, pp. 118–136, 2013.
- [62] R. C. Arkin, T. Balch, and E. Nitz, “Communication of behavioral state in multi-agent retrieval tasks,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 588–594.
- [63] P. E. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini, “Communication strategies in multi-robot search and retrieval: Experiences with mindart,” in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 317–326.
- [64] M. J. Krieger and J.-B. Billeter, “The call of duty: Self-organised task allocation in a population of up to twelve mobile robots,” *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 65–84, 2000.
- [65] J. Timmis, L. Murray, and M. Neal, “A neural-endocrine architecture for foraging in swarm robotic systems,” in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Springer, 2010, pp. 319–330.

- [66] P. Rongier and A. Liégeois, “Analysis and prediction of the behavior of one class of multiple foraging robots with the help of stochastic petri nets,” in *IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 5. IEEE, 1999, pp. 143–148.
- [67] K. Lerman and A. Galstyan, “Mathematical model of foraging in a group of robots: Effect of interference,” *Autonomous robots*, vol. 13, no. 2, pp. 127–141, 2002.
- [68] A. Vardy, G. Vorobyev, and W. Banzhaf, “Cache consensus: rapid object sorting by a robotic swarm,” *Swarm Intelligence*, vol. 8, no. 1, pp. 61–87, 2014.
- [69] G. Vorobyev, A. Vardy, and W. Banzhaf, “Conformity and nonconformity in collective robotics: A case study,” in *ECAL 2013: The Twelfth European Conference on Artificial Life*. MIT Press, 2013, pp. 981–988.
- [70] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, “The dynamics of collective sorting robot-like ants and ant-like robots,” in *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior*, 1991, pp. 356–365.
- [71] T. Wang and H. Zhang, “Multi-robot collective sorting with local sensing,” in *Ieee intelligent automation conference (IAC)*. Citeseer, 2003.
- [72] V. Hartmann, “Evolving agent swarms for clustering and sorting,” in *Proceedings of the 7th Annual conference on Genetic and Evolutionary Computation*, 2005, pp. 217–224.
- [73] R. Beckers, O. E. Holland, and J.-L. Deneubourg, “Fom local actions to global tasks: stigmergy and collective robotics,” in *Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic, Volume 1, Volume 2 Prerational Intelligence: Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems, Volume 3*. Springer, 2000, pp. 1008–1022.
- [74] M. Maris and R. Boeckhorst, “Exploiting physical constraints: heap formation through behavioral error in a group of robots,” in *Proceedings of IEEE/RSJ international conference on intelligent robots and systems. IROS’96*, vol. 3. IEEE, 1996, pp. 1655–1660.
- [75] O. Holland and C. Melhuish, “Stigmergy, self-organization, and sorting in collective robotics,” *Artificial life*, vol. 5, no. 2, pp. 173–202, 1999.
- [76] M. Wilson, C. Melhuish, A. B. Sendova-Franks, and S. Scholes, “Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies,” *Autonomous Robots*, vol. 17, no. 2, pp. 115–136, 2004.
- [77] A. Vardy, “Accelerated patch sorting by a robotic swarm,” in *2012 Ninth Conference on Computer and Robot Vision*. IEEE, 2012, pp. 314–321.

- [78] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, “Clustering objects with robots that do not compute,” in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 2014, pp. 421–428.
- [79] A. Martinoli, A. J. Ijspeert, and L. M. Gambardella, “A probabilistic model for understanding and comparing collective aggregation mechanisms,” in *European Conference on Artificial Life*. Springer, 1999, pp. 575–584.
- [80] S. Kazadi, A. Abdul-Khaliq, and R. Goodman, “On the convergence of puck clustering systems,” *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 93–117, 2002.
- [81] W. W. Cohen, “Adaptive mapping and navigation by teams of simple robots,” *Robotics and autonomous systems*, vol. 18, no. 4, pp. 411–434, 1996.
- [82] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, “Pheromone robotics,” *Autonomous Robots*, vol. 11, no. 3, pp. 319–324, 2001.
- [83] A. Sgorbissa and R. C. Arkin, “Local navigation strategies for a team of robots,” *Robotica*, vol. 21, no. 5, pp. 461–473, 2003.
- [84] F. Ducatelle, G. A. D. Caro, and L. M. Gambardella, “Robot navigation in a networked swarm,” in *International Conference on Intelligent Robotics and Applications*. Springer, 2008, pp. 275–285.
- [85] A. Wurr and J. Anderson, “Multi-agent trail making for stigmergic navigation,” in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2004, pp. 422–428.
- [86] J. Mullins, B. Meyer, and A. P. Hu, “Collective robot navigation using diffusion limited aggregation,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2012, pp. 266–276.
- [87] V. Sperati, V. Trianni, and S. Nolfi, “Self-organised path formation in a swarm of robots,” *Swarm Intelligence*, vol. 5, no. 2, pp. 97–119, 2011.
- [88] M. Szymanski, T. Breitling, J. Seyfried, and H. Wörn, “Distributed shortest-path finding by a micro-robot swarm,” in *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer, 2006, pp. 404–411.
- [89] E. Castello, T. Yamamoto, Y. Nakamura, and H. Ishiguro, “Task allocation for a robotic swarm based on an adaptive response threshold model,” in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*. IEEE, 2013, pp. 259–266.
- [90] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, “Self-organized task allocation to sequentially interdependent tasks in swarm robotics,” *Autonomous agents and multi-agent systems*, vol. 28, no. 1, pp. 101–125, 2014.



- [91] L. E. Parker, “Alliance: An architecture for fault tolerant multirobot cooperation,” *IEEE transactions on robotics and automation*, vol. 14, no. 2, pp. 220–240, 1998.
- [92] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, “Self-organised task allocation in a group of robots,” in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 389–398.
- [93] A. Campo and M. Dorigo, “Efficient multi-foraging in swarm robotics,” in *European Conference on Artificial Life*. Springer, 2007, pp. 696–705.
- [94] W. Agassounon, A. Martinoli, and R. Goodman, “A scalable, distributed algorithm for allocating workers in embedded systems,” in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*, vol. 5. IEEE, 2001, pp. 3367–3373.
- [95] C. Jones and M. J. Mataric, “Adaptive division of labor in large-scale minimalist multi-robot systems,” in *Proceedings 2003 IEEE/RSJ international conference on intelligent robots and systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1969–1974.
- [96] W. Liu, A. F. Winfield, J. Sa, J. Chen, and L. Dou, “Towards energy optimization: Emergent task allocation in a swarm of foraging robots,” *Adaptive behavior*, vol. 15, no. 3, pp. 289–305, 2007.
- [97] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, “Autonomous self-assembly in swarm-bots,” *IEEE transactions on robotics*, vol. 22, no. 6, pp. 1115–1130, 2006.
- [98] T. P. Flanagan, K. Letendre, W. Burnside, G. M. Fricke, and M. Moses, “How ants turn information into food,” in *2011 IEEE symposium on artificial life (ALIFE)*. IEEE, 2011, pp. 178–185.
- [99] T. P. Flanagan, K. Letendre, W. R. Burnside, G. M. Fricke, and M. E. Moses, “Quantifying the effect of colony size and food distribution on harvester ant foraging,” *PloS one*, vol. 7, no. 7, p. e39427, 2012.
- [100] J. P. Hecker and M. E. Moses, “Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms,” *Swarm Intelligence*, vol. 9, no. 1, pp. 43–70, 2015.
- [101] G. M. Fricke, P. H. Joshua, D. G. Antonio, T. T. Linh, and E. M. Melanie, “A Distributed Deterministic Spiral Search Algorithm for Swarms,” in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. Copyright 2016, IEEE reprinted with permission from the authors., 2016, pp. 4430–4436.
- [102] C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, *et al.*, “Argos: a modular, parallel, multi-engine simulator for multi-robot systems,” *Swarm intelligence*, vol. 6, no. 4, pp. 271–295, 2012.
- [103] D. J. Sumpter and M. Beekman, “From nonlinearity to optimality: pheromone trail foraging by ants,” *Animal behaviour*, vol. 66, no. 2, pp. 273–280, 2003.



- [104] T. H. Collett, B. A. MacDonald, and B. P. Gerkey, “Player 2.0: Toward a practical robot programming framework,” in *Proceedings of the Australasian conference on robotics and automation (ACRA 2005)*. Citeseer Citeseer, 2005, p. 145.
- [105] R. Vaughan, “Massively multi-robot simulation in stage,” *Swarm intelligence*, vol. 2, no. 2, pp. 189–208, 2008.
- [106] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.
- [107] B. Browning and E. Tryzelaar, “Übersim: a multi-robot simulator for robot soccer,” in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 948–949.
- [108] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, *et al.*, “Swarmanoid: a novel concept for the study of heterogeneous robotic swarms,” *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 60–71, 2013.
- [109] R. L. Graham, “Bounds for certain multiprocessing anomalies,” *Bell system technical journal*, vol. 45, no. 9, pp. 1563–1581, 1966.
- [110] B. Roy and B. Sussmann, “Les problemes d’ordonnancement avec contraintes disjonctives,” *Note ds*, vol. 9, 1964.
- [111] J. Błażewicz, E. Pesch, and M. Sterna, “The disjunctive graph machine representation of the job shop scheduling problem,” *European Journal of Operational Research*, vol. 127, no. 2, pp. 317–331, 2000.
- [112] W. Liu, “Design and modelling of adaptive foraging in swarm robotic systems,” Ph.D. dissertation, Faculty of Environment and Technology, University of the West of England, Bristol, 2008.
- [113] W. Liu and A. F. T. Winfield, “Modeling and optimization of adaptive foraging in swarm robotic systems,” *International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, Dec. 2010.
- [114] Wenguo Liu, A. F. Winfield, Jin Sa, Jie Chen, and Lihua Dou, “Towards energy optimization: Emergent task allocation in a swarm of foraging robots,” *Adaptive Behavior*, 2007.
- [115] E. Castello, T. Yamamoto, F. D. Libera, W. Liu, A. F. Winfield, Y. Nakamura, and H. Ishiguro, “Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach,” *Swarm Intelligence*, 2016.
- [116] J. P. Hecker and M. E. Moses, “Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms,” *Swarm Intelligence*, vol. 9, no. 1, pp. 43–70, 2015.

- [117] Q. Lu, M. E. Moses, and J. P. Hecker, “A Scalable and Adaptable Multiple-Place Foraging Algorithm for Ant-Inspired Robot Swarms.” Workshop on On-line decision-making in multi-robot coordination, 2016 Robotics Science and Systems Conference, arXiv:1612.00480, 2016.
- [118] T. P. Flanagan, K. Letendre, W. R. Burnside, G. M. Fricke, and M. E. Moses, “Quantifying the effect of colony size and food distribution on harvester ant foraging,” *PLOS ONE*, vol. 7, pp. 1–9, 07 2012.
- [119] G. M. Fricke, P. H. Joshua, D. G. Antonio, T. T. Linh, and E. M. Melanie, “A Distributed Deterministic Spiral Search Algorithm for Swarms,” in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. Copyright 2016, IEEE reprinted with permission from the authors., 2016, pp. 4430–4436.
- [120] Q. Lu, A. D. Griego, G. M. Fricke, and M. E. Moses, “Comparing physical and simulated performance of a deterministic and a bio-inspired stochastic foraging strategy for robot swarms,” in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 9285–9291.
- [121] M. Tindo, M. Kenne, and A. Dejean, “Advantages of multiple foundress colonies in *belonogaster juncea juncea* l.: greater survival and increased productivity,” *Ecological Entomology*, vol. 33, no. 2, pp. 293–297, 2008.
- [122] C. A. Chapman, L. J. Chapman, and R. McLaughlin, “Multiple central place foraging by spider monkeys: Travel consequences of using many sleeping sites,” *Oecologia*, vol. 79, no. 4, pp. 506–511, 1989. [Online]. Available: <http://www.jstor.org/stable/4218988>
- [123] Q. Lu, “An efficient multiple-place foraging algorithm for scalable robot swarms,” 2019.
- [124] G. Pini, A. Brutschy, A. Scheidler, M. Dorigo, and M. Birattari, “Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer,” *Artificial Life*, vol. 20, no. 3, pp. 291–317, July 2014.
- [125] E. Ferrante, A. E. Turgut, E. Duñez-Guzmán, M. Dorigo, and T. Wenseleers, “Evolution of self-organized task specialization in robot swarms,” *PLOS Computational Biology*, vol. 11, no. 8, pp. 1–21, 08 2015.
- [126] Y. Chen, B. W. Wah, and C.-W. Hsu, “Temporal planning using subgoal partitioning and resolution in sgplan,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, pp. 323–369, 2006.
- [127] M. Fox and D. Long, “PDDL2.1: An extension to PDDL for expressing temporal planning domains,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, pp. 61–124, 2003.
- [128] J. E. Kelley and M. R. Walker, “Critical-path planning and scheduling,” in *Proceedings of The Eastern Joint Computer Conference*, 1959.
- [129] R. Sosič and J. Gu, “Efficient local search with conflict minimization: A case study of the *n*-queens problem,” *Knowledge and Data Engineering*, vol. 6, no. 5, pp. 661–668, 1994.

