Ph.D. DISSERTATION

Path Planning for Efficient Deployment and Collection of a Marsupial Robot Team

Marsupial 로봇 팀의 효율적인 배치 및 회수를 위한 경로 계획에 관한 연구

BY

HUNSUE LEE

AUGUST 2017

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Path Planning for Efficient Deployment and Collection of a Marsupial Robot Team

Marsupial 로봇 팀의 효율적인 배치 및 회수를 위한 경로 계획에 관한 연구

BY

HUNSUE LEE

AUGUST 2017

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Path Planning for Efficient Deployment and Collection of a Marsupial Robot Team

Marsupial 로봇 팀의 효율적인 배치 및 회수를 위한 경로 계획에 관한 연구

지도교수 이 범 희

이 논문을 공학박사 학위논문으로 제출함

2017 년 6 월

서울대학교 대학원

전기정보 공학부

이 훈 수

이훈수의 공학박사 학위논문을 인준함

2017 년 6 월

위 원 장	조 동 일
부위원장	이 범 희
위 원	서 승 우
위 원	심 형 보
위 원	지 상 훈

Abstract

This dissertation presents time-efficient approaches to path planning for initial deployment and collection of a heterogeneous marsupial robot team consists of a large-scale carrier robot and multiple mobile robots. Although much research has been conducted about task allocation and path planning of multi-robot systems, the path planning problems for deployment and collection of a marsupial robot team have not been fully addressed. The objectives of the problems are to minimize the duration that mobile robots require to reach their assigned task locations and return from those locations. Taking the small mobile robot's energy constraint into account, a large-scale carrier robot, which is faster than a mobile robot, is considered for efficient deployment and collection. The carrier robot oversees transporting, deploying, and retrieving of mobile robots, whereas the mobile robots are responsible for carrying out given tasks such as reconnaissance and search and rescue. The path planning methods are introduced in both an open space without obstacles and a roadmap graph which avoids obstacles. For the both cases, determining optimal path requires enormous search space whose computational complexity is equivalent to solving a combinatorial optimization problem. To reduce the amount of computation, both task locations and mobile robots to be collected are divided into a number of clusters according to their geographical adjacency and their energies. Next, the cluster are sorted based on the location of the carrier robot. Then,

an efficient path for the carrier robot can be generated by traveling to each deploying and loading location relevant to each cluster. The feasibility of the proposed algorithms is demonstrated through several simulations in various environments including three-dimensional space and dynamic task environment. Finally, the performance of the proposed algorithms is also demonstrated by comparing with other simple methods.

Keywords: Multi-robot systems, multi-robot path planning, marsupial robot, deployment, collection, energy constraint

Student Number: 2012-30227

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and motivation

1.1.1 Multi-robot system

Robotics technologies are being adapted for widely different applications. The applications include manufacturing automation, military equipment, robot-assisted endoscopic surgery, home cleaning, mobility assistance, unmanned aerial vehicles (UAV), autonomous cars, and even entertainment. As the number and spatial distribution of robotic applications increases, the use of multi-robot systems (MRS) becomes inevitable. Over the past few decades, a great deal of effort has been invested in the field of MRS, which has become important in robotics research. As frequently explained in the literature [18, 87, 120], the potential advantages of MRS over a single robot include: 1) increased spatial coverage; 2) fault-tolerance and robustness by redundancy; 3) flexibility; and

4) cost-effectiveness of total systems. Furthermore, some tasks may be accomplished only by using an MRS when the size of the tasks is too substantial to be done by an individual robot. Therefore, MRS are advantageous in various areas such as object transportation and manipulation [23], exploration [17,84], reconnaissance and surveillance [1,99], search and rescue [69], and map-building [92].

However, there are a few problems that arise from using MRS in practice, for example, the problems of communication, task allocation, cooperation and coordination, and collision avoidance. Most desired applications of MRS require resolving more than one problem, depending on the type of application. Therefore, it is important to characterize the desired MRS and choose a suitable approach for each problem. Among those problems, cooperation with other robots is the one of the fundamental issues. Cooperation can be divided into two main categories: 1) tight cooperation that requires continuous synchronization and coordination, such as box-pushing [72] and formation keeping [6,93]; and 2) loose cooperation, such as exploration [18], surveillance [94], mapping [30, 84], and tracking [9, 88].

For efficient cooperation, either centralized or distributed control architecture should be used in a rover. In a centralized system, almost all tasks are processed in a central controller, whereas each robot only performs sensing and actuating. [1] However, as the system increases in size, processing ability and communication bandwidth are limited, and fault tolerance might become more significant [48]. On the other hand, in a distributed (decentralized) system, each robot plans and solves its given tasks independently. This system

[1] In this study, the mobile robot includes UAV.

generally seems more flexible and robust than centralized systems. However, each robot cannot tell whether the entire mission is completed, and the amount of communication might increase. Therefore a mixed form of two systems, in which the central controller coordinates each rover's behavior, can be used. The central coordinator of an MRS is often concerned with task allocation, which is known to be a non-deterministic polynomial-time (NP) hard problem, meaning that optimal solutions cannot be found quickly for large problems [34]. Therefore, solutions to this problem are typically approximations that are acceptable in practice [5]. Allocating roles to each of the team robots is also one of the primary issues in MRS. It allows robots to specialize in behaviors for which they are responsible [19]. The overall efficiency of an MRS is expected to increase when the robots are specialized. The heterogeneity often enables each robot to perform its roles explicitly.

1.1.2 Marsupial robot team

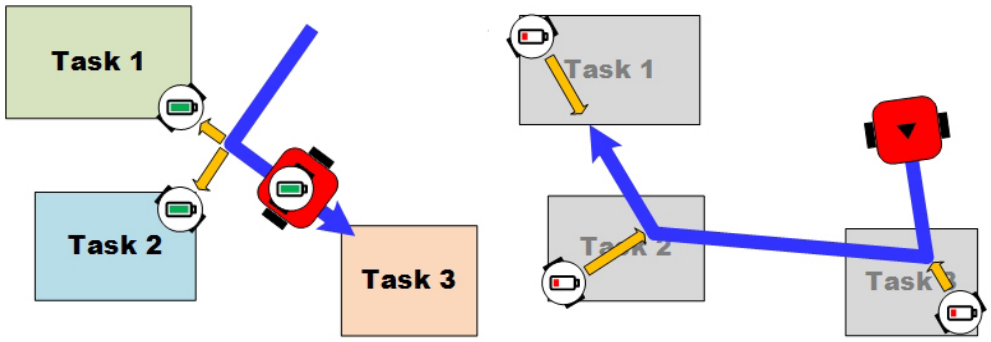
The marsupial robot team [26,28,56,73,80,99,101,118] has a special configuration with both characteristics: tight cooperation and loose cooperation. In other words, when one robot carries another robot, they work in tight cooperation. But once the robot releases the other robots, they work in loose cooperation. In this study, the heterogeneous robots composing this team are referred to as a *carrier* and a *rover*, respectively. The heterogeneity of the carrier and the rover allows each robot to perform its explicit roles, complementing each other's limits and increasing the efficiency of the entire system [19]. Advantages to rovers from carrier are: 1) rapid and energy efficient transportation to target area; 2)

Figure 1.1: Conceptual picture of the descent of NASA's Curiosity rover to the Martian surface by sky crane for Mars mission launching in 2020 [29].

protection during transport; 3) shelter from environmental conditions; 4) power recharge and swapping; 5) processing station; 6) proxy processing; and 7) communication relay. Therefore, this type of marsupial robot team can be used in various scenarios such as search and rescue [80], cooperative SLAM [118], planetary exploration [29], surveillance [99, 101], and drone delivery [73].

Figure 1.1 shows an example of using a marsupial robot team for the exploration of Mars [29]. The rover may not be able to overcome some terrain to go directly to its destination, or there may be energy or time constraints to travel a long distance. Therefore, the rover is maneuvered and deployed using the sky crane of an UAV, which acts as the carrier robot. Once the rover successfully lands, the actual exploration mission will be performed by the rover. If the UAV has a sufficient loading capacity or if the rover is light and small enough, it will be possible to transport and deploy more than one rover.

Likewise, in this paper, a fleet of rovers is transported by a carrier as depicted in Figure 1.2. By using this combination, total fuel cost can be econo-



(a) Deployment (b) Collection

Figure 1.2: Examples of (a) deployment and (b) collection of rovers with energy constraint by using transportation of a carrier. The red rounded rectangle is the carrier.

mized, and the maximum traveling distance of each rover can be overcome. The carrier, which moves faster than the rovers, is in charge of transporting, deploying, and collecting rovers. In addition, the carrier can be either a node that is connected to a central system or a central system itself that communicates with the rovers. Meanwhile, the energy of a rover being transported by the carrier is not consumed. Once a rover is unloaded at the deployment location, it begins moving to its initial task location. On the other hand, the rovers that have completed their given tasks should be collected, unless the cost of collecting the rovers is more expensive than the cost of the rovers. Assume that there is a series of optimal deployment points for the given set of tasks. Then, a series of optimal collection points can be expected to be the inverted series of optimal deployment points. However, this is applicable only when both the locations and the energies of all the robots at the beginning of collection are the same as those at the end of deployment. For example, if the location of the carrier is

changed from the last location in deployment, the optimal collecting path cannot be the same as the previous optimal deployment path. Furthermore, rovers are expected to be fully charged before deployment. Thus, we can assume that all the energies of the rovers are the same, while the remaining energies of the rovers after completing their tasks may be different from one another.

Path planning is a fundamental problem for a mobile robot to perform its mission without colliding with obstacles in the environment. Path planning problems for a robot [14, 33, 47] or MRS [12, 16] have been studied extensively. However, to the best of our knowledge, only a few studies [56, 62, 73, 75, 76, 118] have addressed the path planning problem of the marsupial robot team. In this study, our goal is to minimize the time rovers take to reach their assigned task location by using a carrier robot. (For example, in search and rescue mission, minimizing the time should be the most important criterion.) To implement such a system, we need to consider the following.

- **Communication:** MRS requires communication for cooperation and coordination. This dissertation assumes a fully connected network ensuring connectivity is not the main objective. The number of connections rapidly increases each time a robot is added because topology requires $n(n-1)/2$ connections for n robots. If so, the use of a star network centered on a carrier may be considered. Note that, in an environment without obstacles, Bluetooth 4.0 and WiFi direct can communicate up to 100m (in most cases 20-30m) [15] and 200m [102], respectively. An additional wireless sensor network (WSN) deployment may be considered in case a robot

travels outside these ranges.

- **Obstacle Avoidance:** Obstacle avoidance, an essential factor for the continued use and reliability of a robot, can be approached from a global and local perspective. Usually global planning considers maps or static obstacles in the environment and local planning avoids unmapped or moving obstacles. In this study, we first consider an environment without obstacle. Then we consider static obstacles with the assumption that moving obstacles can be avoided through conventional reactive planning [55, 77]. Static obstacles can be avoided by creating roadmaps in open spaces. We use a probabilistic roadmap method (PRM), one of the most widely used sampling-based map building algorithms [57], to create the roadmap.

- Energy: There are three methods for path planning optimization: time, path-length, and energy optimization. As we focus on minimizing the time, we assume that the energy of a carrier, the larger robot, is infinite, and the battery capacity of a rover, the smaller robot, is limited and would be used for tasks such as image acquisition in addition to movement. For time-efficient path planning, the maximum possible travel distance with a given energy is important. Therefore, it is assumed that the amount of available energy and the maximum travel distance are linearly proportional.

- Dynamic environment: Due to the dynamics of the environment, new tasks may appear and an ordinary tasks may disappear during the operation. To cope with this, it is necessary to be able to quickly formulate a plan

for not only the initial plan but also the environment that changed during the mission.

Considering these facts, the problems are solved in open space first. Then, they are also solved in a graph which is generated by the PRM that considers static obstacles. Solving the optimal path to these problems is computationally expensive. Assume all rovers are deployed and collected at their exact task locations, and the amount of each task is the same. Then, these problems become equivalent to the traveling salesman problem (TSP), known to be an NP hard problem [25]. However, these problems have much higher computational complexity than the TSP because each candidate location for either deployment or collection can be either anywhere in the two-dimensional (2D) plane or a node in a graph. To solve those problems efficiently, each problem is iteratively divided into two sub-problems based on the distance between either tasks or rovers. (The distance here refers to the Euclidean distance in the open space and the distance calculated by A* algorithm [40] in the graph.) For each dividing step, optimal or nearly optimal solutions for all sub-problems can be found. Then the entire path is generated by merging and adjusting the solutions, and the duration is also calculated. Finally, by comparing the durations, the path with the minimum duration is chosen as for the deployment/collection path. This approach enables near real-time updating of deployment/collection paths in a dynamic environment. However, the method in the graph does not guarantee real-time updating.

1.2 Contributions of the thesis

The objective of this thesis is to develop an algorithmic approach to the path planning of marsupial robot team required to operate in environments. It is important to emphasize that the proposed collaborative algorithms in this thesis can not be analyzed by using the conventional techniques. This thesis is divided into two parts. The first part, which includes chapter 4 and 5 deals with the deployment and collection problems with the marsupial robot team in open space without any obstacle. The second part includes chapter 6, and deals with the path planning of marsupial robot team in an environment with static obstacles. The contribution of each chapter can be summarized as follows.

- **Chapter 2: Related Work.** In this chapter we review previous studies in multi-robot path planning and the other relevant fields. Then, the novelty of this thesis is presented.

- **Chapter 3: Preliminaries.** In this chapter we first introduce some notation and assumption. Then, we review some basic results in PRM and computational geometry on which we will rely throughout the thesis.

- **Chapter 4: Deployment of a Marsupial Robot Team.** In this chapter we study robot deployment problems where rovers have energy constraints. Surprisingly, little is studied about marsupial robot team versions, despite their practical relevance. To fill this gap, we study the following problem: m tasks exist in a bounded environment, and n rovers and a carrier are operated. The carrier transports all rovers initially and

deploys each rover for each task. The aim is to find a carrier's path which makes each rover successfully arrives a given task location, and minimizes the arrival time of all rovers. Our contribution is that we carefully formulate this problem and develop an efficient algorithm for deployment of rovers, considering the rover's energy constraint.

- **Chapter 5: Collection of a Marsupial Robot Team.** In this chapter we study robot collection problems where rovers are scattered after completing their tasks. The goal is to find a carrier's path for all rovers that can be retrieved from their location, minimizing the time. Our contribution is twofolds. First, we formulate this problem. Second, we develop an algorithm for collection which generates efficient collecting path.

- **Chapter 6: Deployment of a Marsupial Robot Team using a Graph.** In this chapter we study robot deployment problems in environment with static obstacles. The aim is to find a carrier's efficient path which avoids obstacles in the environment. This problem has important applications in areas such as surveillance and exploration, and search and rescue. For this problem, we can not use the existing path planning method because we use the marsupial robot team. To this end, first, we derive a roadmap by using PRM taking obstacles into account. Second, we make clusters based on the distances to tasks from the carrier's location which is calculated by using A* algorithm. Finally, we present an algorithm to find a series of deployment locations. We also perform extensive simulations.

- **Chapter 7: Conclusion.** In this final chapter we draw our conclusions, and present some ideas for future research.

Chapter 2

Related Work

The marsupial robot team is a category of MRS. To operate an MRS, several preceding techniques are needed. This entails recognizing the environment, distributing work, and planning the path of each robot. Therefore, multi-robot path planning issue is studied in section 2.1, multi-robot exploration is covered in section 2.2, and multi-robot task allocation is dealt in section 2.3. In addition, simultaneous localization and mapping (SLAM) is briefly introduced in section 2.4. On the other hand, motion planning of collective swarm, which is the other one of two broad categories in multiple mobile robot systems, is covered in section 2.5. In section 2.6, the related work using marsupial-like robot team is described, and the related work for collection is introduced in section 2.7. Then, roadmap generating methods is described in section 6.3. Finally, the novelty of this thesis is again highlighted in section 2.9.

2.1 Multi-robot path planning

The purpose of multi-robot path planning is for given multiple robots in workspace with starting and goal pose to determine path each robot should take reach its goal, while avoiding collisions other robots and obstacles. The optimization criteria of the problem can be minimizing: 1) total path length; 2) time to reach goals; and 3) energy to reach goals. Typically, this is abstracted as the problem of computing a set of non-colliding paths on a graph for multiple robots [89,112]. The related subject is also introduced in section 6.3.

2.2 Multi-robot exploration

Considerable researches for MRS cooperation have been conducted. For example, Wurm *et al.* [119] use exploration strategy which divides space into segments and allocates each robot to the individual segment by using Hungarian method. Based on dynamic programming principle, LaValle *et al.* [58] introduce an algorithm that enables each robot to achieve the goal, minimizing *loss functional* the authors defined. Gomez *et al.* [84] also formalize exploration and map-building solution of heterogeneous robots by using dynamic programming. The robots incrementally explore the environment with reduced search space. Listmann *et al.* [67] investigate frontier-based approach using Voronoi partition for multi-robot coverage problem. For path planning of multiple air vehicles, Zheng *et al.* [121] demonstrate a novel co-evolving and cooperating path planner (CCPP). These path planning approaches for MRS are described in a productive task allocation perspective that minimize travel time or fuel

expenditure [10].

2.3 Multi-robot task allocation

For multi-robot exploration, dynamic task allocation in unknown environment is essential [65]. Berhault *et al.* [13] adapt combinatorial auctions for allocating exploration targets. Rossi *et al.* [98] propose negotiation module of target points for heterogeneous robot team. Their team also develop simultaneous task subdivision and allocation method [97]. Delle *et al.* [27] present decentralized coordination and task allocation algorithm of unmanned aerial vehicles. Nestinger and Cheng [82] study a reconfigurable cooperative system framework which includes task allocation function. As explicit negotiations between many robots often require a considerable amount of communication, Regele and Levi [91] use heuristic priority adjustment after traditional path planning. Grocholsky *et al.* [39] introduce a cooperative surveillance that consists of unmanned ground vehicle (UGV) and UAV.

2.4 Simultaneous localization and mapping

SLAM was originally developed by Leonard *et al.* [64]. Montemerlo *et al.* [79] suggest FastSLAM which is an algorithm that recursively estimates the full posterior distribution over robot pose and landmark locations, yet scales logarithmically with the number of landmarks in the map. In recent years, cooperative strategy is applied to SLAM. Howard [45] suggests multi-robot SLAM using particle filters. Gil *et al.* [36] use Rao-Blackwellized particle filter for visual SLAM. Carpin [20] uses a map merging method for an MRS. Oh *et al.* [85] present

loop-closure detection method using bag-of-words for multi-robot SLAM. However, those studies generally assume the robots are initially scattered and do not consider deployment problem.

2.5 Motion planning of collective swarm

In MRS, intentionally cooperative systems have knowledge of the presence of other robots in the environment. However, collective swarm systems are those in which robots execute their own tasks with only minimal need for knowledge about other robot team members [106]. Collective swarm research is inspired from biological systems such as insect colonies, flocks of birds, and schools of fish [7]. Beni and Wang [11] first use the term *swarm intelligence* in their study. Rigatos [95] suggest two distributed stochastic search algorithm for motion planning of MRS: 1) distributed gradient; and 2) swarm intelligence theory. The proposed algorithms show that the location of each robot and the mean position of robots converge to the destination. The algorithms are based on system-theoretic approach, and stability is proven by Lyapunov analysis. However, heuristic tuning is required for particle swarm optimization (PSO). Jatmiki *et al.* [49] use PSO-based robot for odor source localization in obstacles environment. For the odor source localization, chemotaxis model and anemotaxis model are combined to solve dynamic Advection-Diffusion problems. To resolve local minima problem in conventional PSO, the *neutral-charge* concept is adopted to recognize environmental changes. In addition, the more efficient convergence is implemented by using wind utilization concept. The both proposed models can solve the problems, however, they can not accurately address

the real life scenarios. Rimon and Koditschek [96] address a method to create an artificial potential field in a fixed environment with all the information about the robot model and geographic information. The specific goal is to find a smooth bounded-torque feedback controller until it arrives at the destination and stops, avoiding obstacles. The authors assess the method as *Error detection and recovery*. Masoud and Masoud [71] propose nonlinear, anisotropic, harmonic potential fields (NAHPF) which is a new class of motion planners that can mark a constrained trajectory to a target zone in an environment that need not necessarily be a priori known. To overcome the weakness of hybrid partial differential equation-ordinary differential equation controller (HPC), evolutionary HPC is used in the study. Gazi and Passino [32] discuss the continuous time model of a swarm with M-objects in n-dimensional space. The motion of each object is determined by three factors: 1) attraction by distant objects; 2) repulsion by other objects in close proximity; and 3) attracted by the preferred region. The authors analyze stability and suggest conditions for convergence. Sepulchre *et al.* [105] develop a method to stabilize the relative equilibrium state. The issue of collective stabilization is how to connect them together to reach the desired level rather than how to control the dynamics of each entity. The authors assume that the particles are all identical, moving at a unit velocity, and are all-to-all coupled. The equilibrium corresponds to a parallel motion or a circular motion. The stabilization feedback derived from the Lyapunov function shows exponential stabilization and global convergence. Particle swarm has shown good performance, however, it has not been fully explained how it works, and existing methods have problems such as having unintentional dynamics or

limiting speed to follow trajectory. To cope with these, Clerc and Kennedy [24] propose a method to find the optimum by analyzing discrete time trajectory of particle and expanding it to continuous time. On the other hand, Masehian and Sedighizdeg [70] suggest a heuristic technique for multi robot motion planning which is based on PSO. Local planning uses the PRM. Compared to other PSOs, they are successful in diversity. In other words, there is a strong tendency to search for new places.

2.6 Marsupial robot team

2.6.1 Multi-robot deployment

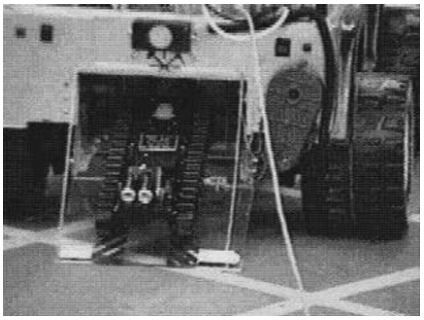
The deployment problem is fundamental to MRS. Wang *et al.* [113] introduce a distributed deployment algorithm for mobile agents, assuming fully connected communication topology. The algorithm initially calculates Voronoi partition. Then robots move to final locations until pre-defined inequality is satisfied. Carpin *et al.* [21] analyze how many robots are required for deployment when there is a chance to fail. Satoh *et al.* [103] propose a framework for dynamic deploying of agents. Although the framework provides coordination, it does not consider dividing and allocating robots for geometrically scattered tasks. By using stochastic extremum seeking method, Ghods *et al.* [35] design a control for deployment of N agents in a planar signal field. However, energy constraint of the agents is not considered in the study. Wang *et al.* [114] investigate a multi-robot deployment method by simple rules, sensors, and precondition. In the study, PSO without *nostalgia* is used as a distributed approach. Kloetzer and Belta [54] demonstrate a deployment method for visiting regions of inter-

est. However, as most of computation should be performed offline before deployment, dynamic adjustment of deployment cannot be achieved. For a spatial-temporal coverage, Baroudi *et al.* [8] discuss GPS-free robots deployment which use landmarks in the deployment area, and is verified in network simulation tool. The method estimates distance by using receive signal strength indicator (RSSI). Several studies of deployment have been conducted for WSN [46, 115], which is strongly related to the coverage problem. Stergiopoulos and Tzes [108] suggest a coordination algorithm for autonomous optimal deployment of the nodes in a WSN. The algorithm relies on suitable partitioning of the sensed space, based on certain Helly-type theorems, guaranteeing distributed information flow. By using relay robots and Voronoi diagram, Uchimura *et al.* [111] use a deployment algorithm that enables leader robot and base station can communicate each other.

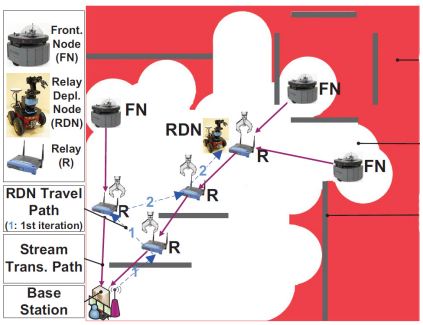
2.6.2 Marsupial robot

Heterogeneous robot team consists of robots with different abilities. In this team, using a larger carrier for transporting and deploying small rovers is advantageous, as the carrier may traverse longer distances than a rover. There are not many studies on the marsupial robot team, most of them [44] focused on novel architecture, and only a few of them focus on the path planning problem of this team.

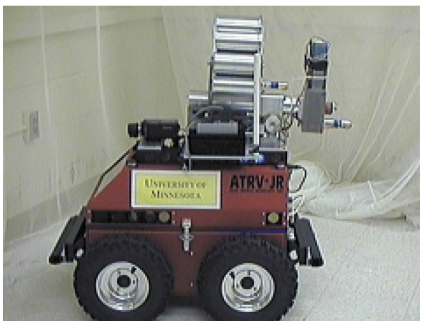
Figure 2.1 shows the example of the cooperative strategy by using marsupial-like robot team. First, one of the earliest studies on architecture implementation is proposed by Murphy *et al.* [80, 81]. (Figure 2.1a.) Wang *et al.* [116] demon-



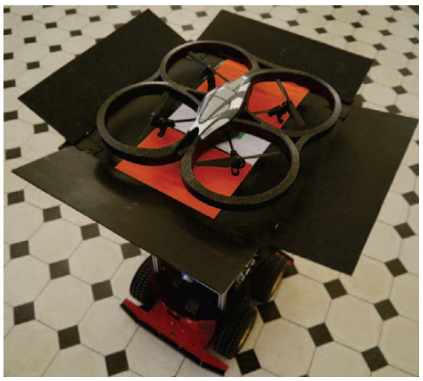
(a) Micro-rover is deployed from the car-like Silver Bullet [81]



(b) Carrier-based sensor deployment [90]



(c) Ranger carries scouts for reconnaissance and surveillance [99]



(d) Ground vehicle carries UAV for indoor surveillance [101]

(e) TraxBot carries several scouts [26]

(f) AVERT robotic system for indoor parking [2]

Figure 2.1: Examples of cooperative strategy for MRS that a robot carries another robots or agents.

strate carrier-based sensor deployment for a WSN. As they assume finite load capacity of the carrier, the carrier has to repeatedly move back to the position where sensors are stored. Pei and Mutka [90]. also introduce a relay deployment method for the front node to explore an unknown environment. However, each sensor itself in those studies cannot move from the deployed location. (Figure 2.1b.) Rybski *et al.* [99] use rangers and scouts for reconnaissance and surveillance. (Figure 2.1c.) The scout combines rolling and jumping locomotion to reach a given area, however path finding technique is not presented. Kadioglu and Papanikolopoulos [51] presented a physical method for transporting scouts by using Pioneer robot. Saska *et al.* [101] develop a heterogeneous unmanned ground vehicle (UGV)-UAV system for cooperative indoor surveillance. (Figure 2.1d.) However, there is only one UAV, and efficient path generation is not considered. Couceiro *et al.* [26] demonstrate a ranger called TraxBot which deploys the scouts, maintaining a maximum the full connectivity of a WSN. (Figure 2.1e.) However they assume that the mission is started only after all scouts reached their goal positions. Drenner *et al.* [28] propose a marsupial system where a multi-level hierarchy allows carriers to transport large number of rovers. The study introduces electrical design and vision system of docking station, and the actual experiment shows that the exploration robot is moved to a fixed position on the station. Marek *et al.* [74] introduce Marsubot and Motherbot, and deal with energy allocation and distribution for long term autonomy. Minten *et al.* [78] describe a docking method based on vision information. In all the studies mentioned above, no consideration is given to path planning or only very simple algorithms are presented. Amanatiadis *et al.* [2] introduce an

autonomous vehicle lifting and transportation system what is called *AVERT*. (Figure 2.1f.) However, the efficient deployment of *Bogie* components is not addressed. Tran *et al.* [110] present canine assisted robot deployment for search and rescue.

Some studies deal with the path planning problem of marsupial robot team. Min *et al.* [76] develop a vision-based algorithm for effective deployment of small multiple robots. The algorithm makes marsupial robot to drive into the center of the task area and unload small robots. Mei *et al.* [75] propose a deployment algorithm with energy and timing constraints for coverage problem. The study aims to consume robot's energy efficiently, not exceeding the timing constraint. It means that a slower plan could be made within a given time limit to reduce energy consumption. Therefore, it may be difficult to use the method to missions such as search and rescue where minimizing time is the most important. Furthermore, the study assumes carrier's travel time is negligible since it travels at a much higher speed than robots. However, the dynamics of the robot must be considered for path planning. Wurm *et al.* [118] suggest a way to integrate a traditional cost-based planner with temporal planner. When a carrier encounters a place where it cannot go by itself, it moves to the nearest meeting location and deploys the rover. Compared with the heuristic method of Stachniss [107], the study shows better performance. However, the duration for symbolic actions, such as deploying a rover, is not considered in the planning. Las *et al.* [56] deal with the path planning of marsupial vehicles consisting of a carrier and a passenger for multi-agent surveillance and reconnaissance. The method is useful for reconnaissance that minimizes the exposure by an

opponent, but there is a limit that one carrier can deploy only one passenger. Mathew *et al.* [73] address the path planning of a truck and a quadrotor for package delivery in urban area. They formulate *heterogeneous delivery problem* (HDP) on a graph and propose a feasible solution. However, to efficiently traverse the non-urban environment, it is necessary to consider creating a more efficient node in the open space. Lee *et al.* [60, 62] describe efficient techniques for deployment problem that minimizes time for robots to reach all the task locations, considering energy constraint and robots'' motion capabilities.

2.7 Robot collection

To the best of our knowledge, an analysis of robot collection is not been presented, except for our previous study [61]. Although Sahin and Zergeroglu [100] use a computationally efficient path planning method for a collection of mobile robots, the planning in the study is a sort of formation control. The other studies focus on not robots collection but data collection. Goerner *et al.* [37] investigate constructing a path for mobile data collecting robot in heterogeneous sensor network. The total cost of data collection is defined as the sum of transmission energy of the sensor and movement energy of robot. Hollinger *et al.* [43] introduce path planning for data collection in underwater environment. In the study, a covering set of probabilistic neighborhoods is found. Then the optimal path for the TSP of the covering set is acquired using *Concorde* [4]. Finally, an autonomous underwater vehicle (AUV) tours the path. Chen *et al.* [22] present a strategy that robot can collect the sensing data from WSN. In addition, deployment and collection problems are similar to the transshipment problem in

logistics optimization which aims for minimum-durational transportation, and supply chain management (SCM). Typically the candidate locations in logistics or SCM are finite where optimal or near-optimal solution can be found on a graph.

2.8 Roadmap generation

Motion planning methods to produce a roadmap can be divided into three categories according to approaches. These three methods are based on 1) geometric information; 2) artificial potential fields; and 3) sampling. Since the approaches based on artificial potential fields are introduced in section 2.5, we briefly introduce geography-based methods and sampling based methods in this section.

2.8.1 Geometric algorithms

Visibility graph, which is first suggested by Lozano *et al.* [68], constructs a path as a polygonal line connecting start position and goal position through vertices of Cobs. This method is conceptually simple and produces shortest paths. However, this graph is only suitable for 2D. The best algorithm for visibility graph is $O(n^2 \log n)$. Two common cell decomposition approaches use exact cell and approximate cell. Cell decomposition method is numerically stable and simple to implement. Voronoi diagrams [109] are well studied for reactive mobile robot path planning. This method maximizes clearance which is good for an uncertain robot. However, this results in unnatural attraction to open space, and suboptimal paths.

2.8.2 Sampling-based algorithms

Sampling-based algorithm samples *milestones* in free-configuration space. Then a roadmap is constructed by connecting two milestones. This approach works well for high-dimensional configuration spaces. Two popular methods using sampling-based approach are rapidly-exploring random tree (RRT) and PRM. Sampling-based planners are more efficient in most practical problems and probabilistically complete. However, they offer weaker guarantees.

Rapidly-exploring random tree

RRT, developed by Lavalle and Kuffner [59], is a single-query search algorithm which randomly builds a space-filling tree. The tree is incrementally constructed. RRT is easy to implement. However, the convergence rate is unknown and metric sensitivity should be reduced. There are a lot variants of RRT such as RRT* and bi-directional-RRT (BiRRT).

Probabilistic roadmap method

PRM is multi-query algorithm which is proposed by Kavraki *et al.* [52]. The idea is to take random samples from configuration, declare them as vertices in free-configuration space, and try to connect nearby vertices with local planner. PRM can solve some of previously unsolved problems and support fast queries with enough preprocessing. However, it is neither optimal nor complete. (PRM is probabilistic complete.) Furthermore, the method is unlikely to sample nodes in narrow passages. To navigate narrow passages, obstacle-based PRM (OBPRM) is also proposed [3] as a variant of PRM.

2.9 Novelty of the thesis

This research is different from previous studies in the following ways:

1) This dissertation develops a combined loose and tight cooperation strategy, achieved by carrying one another, for efficient deployment and collection of rovers;

2) The proposed algorithms can reduce both the deployment and collection time of rovers, considering both the energy constraint and the dynamics of robots;

3) The proposed algorithms can reduce the number of computations so that they are applicable for near real-time rovers in a dynamic task environment.

Chapter 3

Preliminaries

In this chapter, we first introduce some notation and assumption. Then, we review some basic results in combinatorics on which we will rely throughout this thesis.

3.1 Notation

To approach the problem, we define space and motion domain first. Given a three-dimensional environment, $\mathcal{W} \in \mathbb{R}^3$ denotes the workspace, and $\mathcal{WO}_i$ denotes a workspace obstacle. Let $\mathcal{QO}_i$ denote a configuration space obstacle such that $\mathcal{QO}_i = \{q \in \mathcal{Q} | R(\mathbf{q}) \cap \mathcal{WO}_i \neq \emptyset\}$, where $\mathbf{q}$ denotes a point in configuration space $\mathcal{Q}$. $\mathcal{Q}_{free} = \mathcal{Q} \setminus (\bigcup \mathcal{QO}_i)$ denote the free configuration space. Initial configurations of a carrier and m holonomic rovers are respectively denoted by $\mathbf{q}_{c_1} = (x_{c_1}, y_{c_1}, 0) \in \mathcal{Q}_{free}$ and $\mathbf{q}^i_{r_1} = (x^i_{r_1}, y^i_{r_1}, z^i_{r_1}) \in \mathcal{Q}_{free}$ for $1 \leq i \leq m$. The orientation of carrier is denoted by θ_c. A goal configuration of the rover

is denoted by $\mathbf{q}_{r_G}^i = (x_{r_G}^i, y_{r_G}^i, z_{r_G}^i) \in \mathcal{Q}_{free}$, for $1 \leq i \leq m$, which is a task location. An undirected graph in $\mathcal{Q}_{free}$, which is initially empty, is denoted by $\mathcal{G} = (N, E)$, where N and E are respectively a set of nodes and edges. In the graph, the configuration $\mathbf{q}_{c_1}$ of the carrier corresponds to node $n_1 \in N$, and the goal configuration $\mathbf{q}_{r_G}^i$ corresponds to $n_g^i \in N$.

On the other hand, the linear velocities of the carrier and the rover are respectively denoted by $v_c^t \leq v_c^{max}$ and $v_r^t \leq v_r^{max}$, and the constant accelerations of the carrier and the rover are respectively denoted by a_c and a_r. The carrier's constant angular velocity is w_c. t_u denotes the duration for unloading rovers and t_l denotes the duration for loading rovers. The initial energy of the carrier and the rover is respectively denoted by ϵ_c and ϵ_{r_i}. The required energy for one task is denoted by ϵ_{task}. A maximum travel distance of a rover for deployment and collection is respectively denoted by d_{deploy} and $d_{collect}^i$.

A set of deployment points to be obtained is denoted by $\Omega = \{\mathbf{q}_{\omega_1} = (x_{\omega_1}, y_{\omega_1}, 0), \mathbf{q}_{\omega_2} = (x_{\omega_2}, y_{\omega_2}, 0), \ldots, \mathbf{q}_{\omega_\alpha} = (x_{\omega_\alpha}, y_{\omega_\alpha}, 0)\}$ where $1 \leq \alpha \leq m$. In the same way, a set of collection points to be obtained is denoted by $\Gamma = \{\mathbf{q}_{\gamma_0} = (x_{\gamma_1}, y_{\gamma_1}, 0), \mathbf{q}_{\gamma_2} = (x_{\gamma_2}, y_{\gamma_2}, 0), \ldots, \mathbf{q}_{\gamma_\beta} = (x_{\gamma_\beta}, y_{\gamma_\beta}, 0)\}$ where $1 \leq \beta \leq n$. A group of rovers which are unloaded at the same deployment location $\mathbf{q}_{\omega_\alpha}$ is denoted by *cluster* $\mathbf{P}_{\omega_\alpha}$, which is a subset of rovers. In the same way, a group of rovers which are collected at the same collection location $\mathbf{q}_{\gamma_\beta}$ is denoted by *cluster* $\mathbf{P}_{\gamma_\beta}$, which is a subset of rovers. Clearly, arbitrary two clusters are mutually exclusive, and $\bigcup_{k=1}^{\alpha} P_{\omega_k} = \bigcup_{k=1}^{\beta} P_{\gamma_k}$.

3.2 Assumptions

To simplify the problems, additional assumptions are given as follows.

- There is no collision between robots.
- Moving obstacles can be avoided by using existing local planning methods.
- The rover is a holonomic robot.
- The carrier moves faster than or equal to a rover, *i.e.*, $v_c^{max} \geq v_r^{max}$.
- Each rover can communicate with the carrier via a wireless network. However, the energy for communication is not considered.
- Each robot can be located by itself or by other positioning systems.
- Each task requires exactly one rover to execute it, and each rover is capable of executing at most one task at any given time.
- The number of rovers m is always bigger than the number of tasks.
- The initial energy of the carrier ϵ_c is sufficient for deploying, moving, and collecting all rovers. Therefore, in the latter part of this study, only the energy of a rover is considered.
- All rovers initially have the same amount of energy. After subtracting ϵ_{task} for a task, all the remaining energy $\epsilon_{r_i} - \epsilon_{task}$ is used for round-trip of a rover from the deployment position. Therefore, the maximum traveling distance of a rover d_{deploy} for approaching task location is determined by half of the remaining energy. By assuming there is a linear correlation

ρ between the energy and d_{deploy}, then $d_{deploy} = \rho(\epsilon_{r_i} - \epsilon_{task})/2$ for all rovers.

- Each rover can estimate its remaining energy after finishing its given task.
- The carrier can deploy and collect rovers only when it is not moving.
- Assuming dynamic environment, existing tasks can disappear and new tasks can be created during the mission. The rovers to be collected also can disappear.
- The generated path is well followed by the carrier and rovers

3.3 Clustering algorithm

Clustering methodology is crucial for minimizing the duration. Typical clustering algorithms [31] are center-based methods such as *k-means* [42], hierarchical methods [50], density-based methods [41], and grid-based methods [104]. However, guaranteeing optimality of clustering considering energy constraint is not easy. Therefore, for efficient computing, a cluster is divided into two clusters iteratively by the following procedure:

1) Find two farthest tasks in a cluster and assign them as two clusters;
2) For each task, calculate the distances to two farthest tasks, and assign the task to the cluster which includes the closer one.

Note that if there are more than one pair of the farthest two points, only one set having the nearest task to the carrier is selected. This cluster dividing method is described in Algorithm 3.1.

Algorithm 3.1: Cluster divider

Input: Cluster data: $I_{cluster}$,
Index of the cluster to be divided: Idx
Output: Divided cluster data: $I_{cluster}$

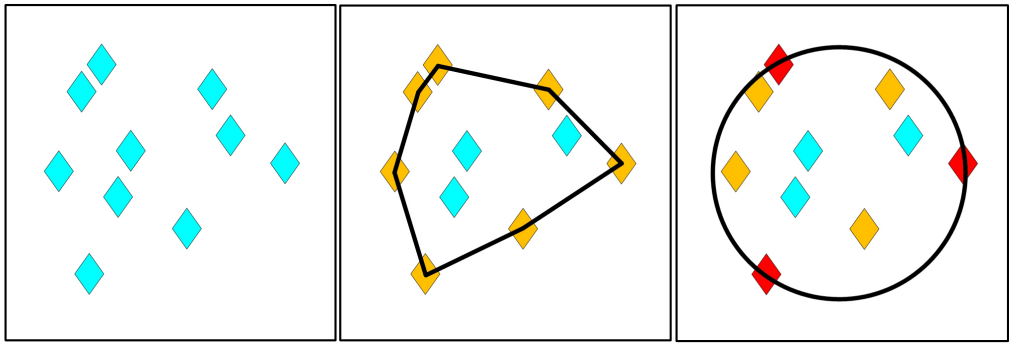
```
/* Collect tasks with respect to the index                    */
```
$size \leftarrow$ GET_SIZE($I_{cluster}$)
for $i \leftarrow 1$ **to** $size$ **do**
  if $I_{cluster}[i].index = Idx$ **then**
    $Dividing\_Targets \leftarrow i$

```
/* Check the number of targets                                */
```
$size \leftarrow$ GET_SIZE($Dividing\_Targets$)
if $size = 1$ **then**
  return

```
/* Choose an arbitrary target and find the farthest point     */
```
$Max\_Dist1 \leftarrow 0$
$Max\_Index1 \leftarrow 0$
for $i \leftarrow 1$ **to** $size$ **do**
  $d \leftarrow$ GET_DIST($Dividing\_Targets[i].position -$
   $Dividing\_Targets[1].position$)
  if $d > Max\_Dist1$ **then**
    $Max\_Dist1 \leftarrow d$
    $Max\_Index1 \leftarrow i$

```
/* Find another farthest point from the preivous point        */
```
$Max\_Dist2 \leftarrow 0$
$Max\_Index2 \leftarrow 0$
for $i \leftarrow 1$ **to** $size$ **do**
  $d \leftarrow$ GET_DIST($Dividing\_Targets[i].position -$
   $Dividing\_Targets[Max\_Index1].position$)
  if $d > Max\_Dist2$ **then**
    $Max\_Dist2 \leftarrow d$
    $Max\_Index2 \leftarrow i$

$[Idx\_Near, Idx\_Far] \leftarrow$
 GET_ORDER($prev\_point, Max\_Index1, Max\_Index2$)
UPDATE_INDEX($I_{cluster}, Idx$)
return $I_{cluster}$



(a) (b) (c)

Figure 3.1: Procedure to find minimum bounded circle (a) A cluster including 10 tasks (b) Convex hull of the cluster (c) Minimum bounded circle.

3.4 Minimum bounded circle and sphere of a cluster

Minimum bounded circle and sphere are used in deployment of a marsupial robot team. Once a cluster of tasks is determined, either the minimum bounded circle in 2D space or the minimum bounded sphere in 3D space is estimated. By using the estimated minimum circle/sphere, the center and the radius of the circle/sphere can be known. They are used for calculating the deployment locations. For α-th cluster, finding the center of the cluster e_α and radius π_α of the circle/sphere is represented as the minimization problem:

$$\begin{aligned} &\text{minimize} && \pi_\alpha \\ &\text{subject to} && \|\mathbf{q}_G^i - e_\alpha\| \leq \pi_\alpha, \end{aligned} \tag{3.1}$$

where $\mathbf{q}_G^i \in \mathbf{P}_{\omega_\alpha}$. Problem solving procedure is described in Figure 3.1. We first find convex hull [38], as shown in Figure 3.1b, so that only outer points are considered for finding the circle/sphere. Among those outer points, arbitrary three points are chosen and the circle/sphere is found. If there exists a point

which is located outside of the circle/sphere, then find another circle/sphere which includes the new point until there is no point outside of the circle/sphere. Next, in case of finding the circle, the center of the bounded circle $(x_{e_\alpha}, y_{e_\alpha})$ and its radius π_α is computed by finding three points (x_1, y_1), (x_2, y_2), (x_3, y_3) which satisfy as follows:

$$(x_1 - x_{e_\alpha})^2 + (y_1 - y_{e_\alpha})^2 = \pi_\alpha^2, \tag{3.2}$$

$$(x_2 - x_{e_\alpha})^2 + (y_2 - y_{e_\alpha})^2 = \pi_\alpha^2, \tag{3.3}$$

$$(x_3 - x_{e_\alpha})^2 + (y_3 - y_{e_\alpha})^2 = \pi_\alpha^2. \tag{3.4}$$

To eliminate the quadratic terms, subtracting pairs of the equations yield:

$$2(x_1 - x_2)x_{e_\alpha} + 2(y_1 - y_2)y_{e_\alpha} = x_1^2 - x_2^2 + y_1^2 - y_2^2, \tag{3.5}$$

$$2(x_1 - x_3)x_{e_\alpha} + 2(y_1 - y_3)y_{e_\alpha} = x_1^2 - x_3^2 + y_1^2 - y_3^2. \tag{3.6}$$

Similarly, in case of finding the sphere, the center of the bounded sphere $(x_{e_\alpha}, y_{e_\alpha}, z_{e_\alpha})$ and its radius π_α is computed by finding three points (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) which satisfy as follows:

$$(x_1 - x_{e_\alpha})^2 + (y_1 - y_{e_\alpha})^2 + (z_1 - z_{e_\alpha})^2 = \pi_\alpha^2, \tag{3.7}$$

$$(x_2 - x_{e_\alpha})^2 + (y_2 - y_{e_\alpha})^2 + (z_2 - z_{e_\alpha})^2 = \pi_\alpha^2, \tag{3.8}$$

$$(x_3 - x_{e_\alpha})^2 + (y_3 - y_{e_\alpha})^2 + (z_3 - z_{e_\alpha})^2 = \pi_\alpha^2. \tag{3.9}$$

To eliminate the quadratic terms, subtracting pairs of the equations yield:

$$2\Big((x_1-x_2)x_{e_\alpha}+(y_1-y_2)y_{e_\alpha}+(z_1-z_2)y_{e_\alpha}\Big) = x_1^2-x_2^2+y_1^2-y_2^2+z_1^2-z_2^2, \tag{3.10}$$

$$2\Big((x_1-x_3)x_{e_\alpha}+(y_1-y_3)y_{e_\alpha}+(z_1-z_3)y_{e_\alpha}\Big) = x_1^2-x_3^2+y_1^2-y_3^2+z_1^2-z_3^2. \quad (3.11)$$

As special cases, if the number of outer points is exactly three, then they make a circle/sphere. If the number is two, then the center is the midpoint of two points, and the radius is the distance from the center to any of two points. If the number is one, then the point itself is the center, and the radius is zero.

Chapter 4

Deployment of a Marsupial Robot Team

The purpose of this chapter is to find a feasible solution to the path planning of a marsupial robot team for initial deployment. We first draw the objective, and analyze the computational complexity of the problems for optimal solutions. To efficiently investigate the problem, we first consider the simple case with two tasks in 2D space where the optimal solution can be obtained easily. Then we extend the case to three-dimensional (3D) case, two-cluster case, and general case. The work in this chapter is based on the conference paper [60,63] and the journal article [62].

4.1 Problem definition

In this section we set up the problem we study in this chapter. The objective is for each rover to reach its relevant task location in minimum time. This

objective is formulated as minimizing the maximum duration as follows:

$$\begin{aligned} &\text{minimize} \quad \max_{i=1,\dots,m} (\|\mathbf{q}_{r_1}^i - \mathbf{q}_{r_G}^i\|) \\ &\text{subject to} \quad \|\mathbf{q}_{r_1}^i - \mathbf{q}_{r_G}^i\| \leq d_{deploy} \quad (i = 1, \dots, m) \end{aligned} \tag{4.1}$$

However, in the setting that the carrier transports multiple rovers, each duration for a task T_i is calculated as a sum of the transportation time from the initial location, the unloading time, and the approaching time for a rover. Let $f(\mathbf{v}, \mathbf{w})$ be the duration of the carrier from one deployment location $\mathbf{v}$ to the next deployment location $\mathbf{w}$. Then the duration can be represented as a sum of moving time and rotating time as the following equation:

$$f(\mathbf{v}, \mathbf{w}) = \begin{cases} S/v_c^{max} + v_c^{max}/a_c + rt(\theta_{c_\mathbf{v}}), & \text{if } S \geq \frac{v_c^{max\,2}}{a_c} \\ 2\sqrt{S/a_c} + rt(\theta_{c_\mathbf{v}}), & \text{otherwise,} \end{cases} \tag{4.2}$$

where $S = \|\mathbf{w} - \mathbf{v}\|$, $\theta_{c_\mathbf{v}}$ is the heading angle of the carrier at $\mathbf{v}$, and $rt(\theta_{c_\mathbf{v}})$ is the computing function of rotating time for the carrier at $\mathbf{v}$. Note that if $S < v_c^{max2}/a_c$ in (2), then v_c^t cannot reach v_c^{max}. If a rover is deployed at α-th deployment point $\mathbf{q}_{\omega_\alpha}$, the duration T_i is represented as the following equation:

$$T_i = \sum_{k=1}^{\alpha} \Big(f(\mathbf{q}_{\omega_{k-1}}, \mathbf{q}_{\omega_k}) + t_u \Big) + \frac{\|\mathbf{q}_{\omega_k} - \mathbf{q}_G^i\|}{v_r^{max}}, \tag{4.3}$$

where $\mathbf{q}_{\omega_{k-1}} = \mathbf{q}_{c_1}$. Then the objective can be formulated as finding the set of optimal deployment points as follows:

$$\Omega^\star = \arg\min_{\Omega \in \mathbb{R}^3} \Big(\max(T_1, T_2, \dots, T_m) \Big). \tag{4.4}$$

Let $g(\mathbf{q}_{\omega_i})$, where $1 \leq i \leq \alpha$, be the maximum duration that a rover travels for its task in $\mathbf{P}_{\omega_i}$ from the relevant deployment location $\mathbf{q}_{\omega_i}$. Then,

$$g(\mathbf{q}_{\omega_i}) = \frac{\max\left(\|\mathbf{q}_{\omega_i} - \mathbf{q}^1_{r_G}\|, \ldots, \|\mathbf{q}_{\omega_i} - \mathbf{q}^b_{r_G}\|\right)}{v_r^{max}}, \tag{4.5}$$

where b is the number of rovers belong to the deployment location $\mathbf{q}_{\omega_i}$, and $\mathbf{q}^b_{r_G}$ is the goal location of b-th rover in $\mathbf{P}_{\omega_i}$. Then the maximum duration τ_i that a rover travels in $\mathbf{P}_{\omega_i}$ from the initial location of the carrier is defined as follows:

$$\tau_i = \sum_{k=1}^{i} \Big(f(\mathbf{q}_{\omega_{k-1}}, \mathbf{q}_{\omega_k}) + g(\mathbf{q}_{\omega_i}). \tag{4.6}$$

Consequently, the equivalent equation of (4.4) can be defined by using (4.6) as follows:

$$\Omega^\star = \arg\min_{\Omega \in \mathbb{R}^3} \Big(\max(\tau_1, \tau_2, \ldots, \tau_\alpha) \Big). \tag{4.7}$$

4.2 Complexity analysis

To compute computational complexity of the problem above, assume the 2D plane is divided into g grids. As the worst case, the number of all possible routes for the carrier is as follows:

$$\begin{aligned} & g + g(g-1) + g(g-1)(g-2) + \ldots \\ & + g(g-1)\ldots(g-m+2)(g-m+1) \\ & = \textstyle\sum_{k=0}^{m-1} \Big(\prod_{j=0}^{k} (g-j) \Big). \end{aligned} \tag{4.8}$$

This is bounded above by g^m. Let $h(\alpha)$ be the number of all cases that each task is assigned among α deployment points. Then $h(\alpha)$ is expressed in a such

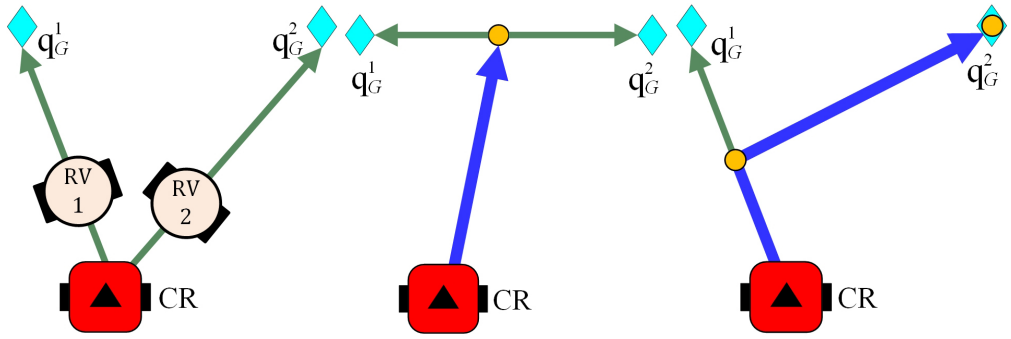
way that each task will select its deployment point with condition that every deployment point should have at least one task as follows:

$$h(\alpha) = \alpha^m - \sum_{k=1}^{\alpha-1}(\binom{\alpha}{k} h(\alpha - k)). \tag{4.9}$$

For the same area, as the grid size decreases, the number of the grids g increases. Therefore, in most cases, $\alpha \ll g$. Compared with g^m then, $h(\alpha)$ can be considered as a constant. Consequently, the overall computational complexity of the problem using a carrier becomes $O(g^m)$. Here, the number of grids g increases as the area increases, and increases when a precise grid is needed. As a result, the complexity increases exponentially, and this brute-force methods for optimality require a tremendous computation. Therefore, heuristic but efficient algorithms are needed.

4.3 Optimal deployment path planning for two tasks

Assume two task locations, $\mathbf{q}_G^1$ and $\mathbf{q}_G^2$, as depicted in Figure 4.1. Then, there are three cases that two rovers are deployed. First, the deployment time to be compared is initially calculated by letting each rover approach each task location without any transportation by a carrier, unless the energy of the rover is not enough to reach the goal, as in Figure 4.1a. Second, the carrier moves to the midpoint of two tasks and deploys two rovers, as in Figure 4.1b. In the most cases, the initially calculated deployment time is longer than the time calculated by using a carrier. If the duration in Figure 4.1b is faster than the duration in Figure 4.1a, the solution in Figure 4.1b is chosen. In the same way, Figure 4.1c can be chosen if the duration is faster than those two previous solutions.



(a) Case I (b) Case II (c) Case III

Figure 4.1: Three cases of deployment for two tasks. (a) Case I: Two rovers directly approach (b) Case II: Carrier goes the midpoint of two tasks and deploys two rovers (c) Case III: Carrier goes first deployment location and approaches to second task location.

Therefore, the first deployment location in Figure 4.1c should be calculated for the optimal solution.

4.3.1 Deployment for two tasks in 2D space

Consider two tasks $\mathbf{q}_G^1$ and $\mathbf{q}_G^2$ again, as presented in Figure 4.2. In the figure, the carrier is closer to $\mathbf{q}_G^1$ than $\mathbf{q}_G^2$. Therefore, the carrier should approach $\mathbf{q}_G^1$ first to reduce the overall deployment time. In this case, generally, the optimal duration can be achieved when two rovers respectively arrive two goal locations at the same time. [1] If there is no delay on the carrier resulted from rotation, acceleration/deceleration, and unloading at $\mathbf{q}_{\omega_2}$, then the optimal deployment location is the intersection of line segment $\overline{\mathbf{q}_{c_1}\mathbf{q}_G^1}$:

$$y = \frac{y_{c_1} - y_{G_1}}{x_{c_1} - x_{G_1}}(x - x_{c_1}) + y_{c_1}, \tag{4.10}$$

[1] This may not be the optimal duration, if $\mathbf{q}_G^1$ is too close or $\mathbf{q}_G^2$ is too far from the carrier.

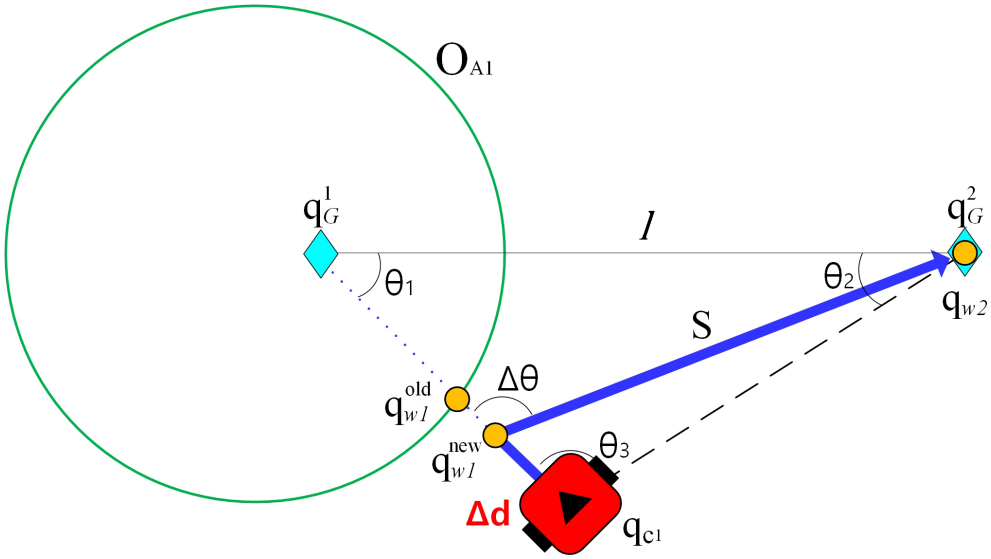


Figure 4.2: Path planning for two tasks, $\mathbf{q}_G^1$ and $\mathbf{q}_G^2$

where $\min(x_{c_1}, x_{G_1}) \leq x \leq \max(x_{c_1}, x_{G_1})$, and the *circle of Apollonius* O_{A1}. O_{A1} has a given ratio of distances $|v_c^{max}|/|v_r^{max}|$ to two given points $\mathbf{q}_G^1$ and $\mathbf{q}_G^2$. By using internal and external division of line segment $\overline{\mathbf{q}_G^1\mathbf{q}_G^2}$ as follows:

$$\text{internal-division: } \left(\frac{v_r^{max}x_{G_2} + v_c^{max}x_{G_1}}{v_r^{max} + v_c^{max}}, \frac{v_r^{max}y_{G_2} + v_c^{max}y_{G_1}}{v_r^{max} + v_c^{max}}\right), \tag{4.11}$$

$$\text{external-division: } \left(\frac{v_r^{max}x_{G_2} - v_c^{max}x_{G_1}}{v_r^{max} - v_c^{max}}, \frac{v_r^{max}y_{G_2} - v_c^{max}y_{G_1}}{v_r^{max} - v_c^{max}}\right), \tag{4.12}$$

the equation of O_{A1} is formulated as the following equation:

$$\left(x - \frac{(v_c^{max})^2 x_{G_1} - (v_r^{max})^2 x_{G_2}}{(v_c^{max})^2 - (v_r^{max})^2}\right)^2 + \left(y - \frac{(v_c^{max})^2 y_{G_1} - (v_r^{max})^2 y_{G_2}}{(v_c^{max})^2 - (v_r^{max})^2}\right)^2$$
$$= (v_c^{max})^2 (v_r^{max})^2 \frac{(x_{G_1} - x_{G_2})^2 + (y_{G_1} - y_{G_2})^2}{\left(v_c^{max2} - v_r^{max2}\right)^2}. \tag{4.13}$$

Considering all the delays, the carrier should deploy first rover at the optimal location $\mathbf{q}_{\omega_1}^{new}$ in Figure 4.2, which is closer to the carrier than $\mathbf{q}_{\omega_1}^{old}$. $\mathbf{q}_{\omega_1}^{new}$ can be computed by finding Δd in the equation below:

$$\frac{\xi - \Delta d}{v_r^{max}} = f(\mathbf{q}_{\omega_1}^{new}, \mathbf{q}_{\omega_2}) + t_u, \tag{4.14}$$

where $\xi = \|\mathbf{q}_{c_1} - \mathbf{q}_{G_1}\|$, which means the distance between the carrier and the first task, and:

$$S = \sqrt{l^2 + (\xi - \Delta d)^2 - 2l(\xi - \Delta d)\cos\theta_1}, \tag{4.15}$$

$$rt(\theta_{\omega 1}^{new}) = \Big(\theta_3 + \frac{\Delta d}{\xi} \cdot \theta_2\Big)/w_c \tag{4.16}$$

in the function f in (4.14). Once Δd is calculated, the distance from $\mathbf{q}_{\omega_1}^{new}$ to $\mathbf{q}_G^1$, $\|\mathbf{q}_{\omega_1}^{new} - \mathbf{q}_G^1\|$ is determined. Here, the distance should not exceed the maximum traveling distance of a rover, *i.e.*, $\|\mathbf{q}_{\omega_1}^{new} - \mathbf{q}_G^1\| \leq d_{deploy}$. If not, the deployment location $\mathbf{q}_{\omega_1}^{new}$ should be moved until $\|\mathbf{q}_{\omega_1}^{new} - \mathbf{q}_G^1\| = d_{deploy}$, where the energy constraint is satisfied. Therefore, the optimal deployment solution for two tasks in 2D space can be acquired at all times by using this method.

4.3.2 Deployment for two tasks in 3D space

Figure 4.3 describes how the optimal deployment locations can be obtained for two tasks in 3D space. The Δd for $\mathbf{q}_{\omega_1}^{new}$ can be found by equalizing the durations that first rover moves from $\mathbf{q}_{\omega_1}^{new}$ to $\mathbf{q}_G^1$, and the carrier moves from $\mathbf{q}_{\omega_1}^{new}$ to $\mathbf{q}_{\omega_2}$ plus second rover moves from $\mathbf{q}_{\omega_2}$ to $\mathbf{q}_G^2$ as follows:

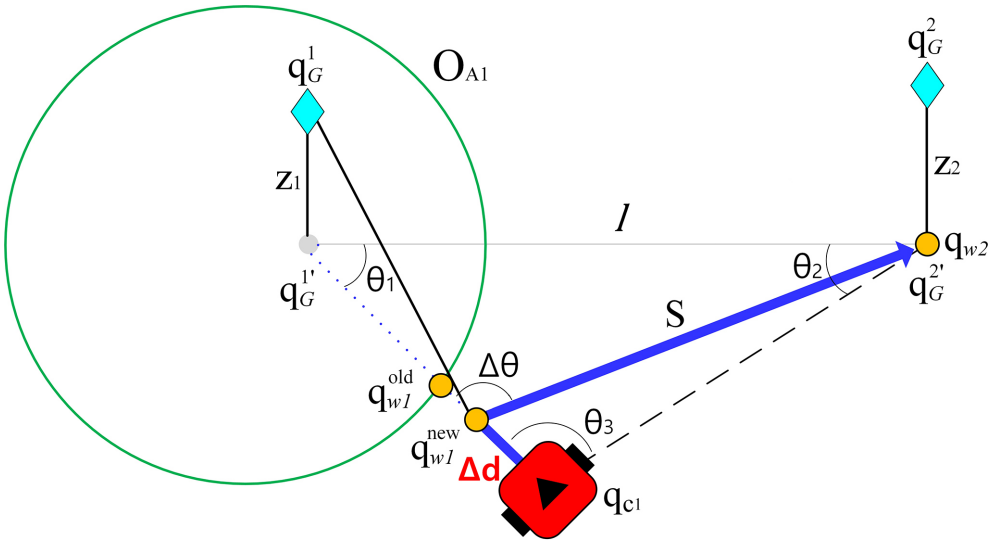


Figure 4.3: Finding the optimal deployment location $\mathbf{q}_{\omega_1}^{new}$ for two given tasks, $\mathbf{q}_G^1$ and $\mathbf{q}_G^2$ in 3D space

$$\frac{\sqrt{(\xi - \Delta d)^2 + |z_{G_1}|^2}}{v_r^{max}} = f(\mathbf{q}_{\omega_1}^{new}, \mathbf{q}_{\omega_2}) + t_u + \frac{|z_{q_1}|}{v_r^{max}}, \tag{4.17}$$

where $0 \leq \Delta d \leq \xi$. If any height of the task location z_{G_i} is the same as d_{deploy}, the carrier should arrive $(x_{G_i}, y_{G_i}, 0)$ at any cost. However, the location cannot be reached, if the height is bigger than d_{deploy}.

4.4 Path planning algorithm of a marsupial robot team for deployment

Expanding on the deployment method for two tasks, assume two clusters of tasks, $\mathbf{P}_{\omega_1}$ and $\mathbf{P}_{\omega_2}$, and their imaginary task locations $\mathbf{e}_1$ and $\mathbf{e}_2$ which are located at the center of the clusters as depicted in Figure 4.4. Then two deployment points $\mathbf{q}_{\omega_1}$ and $\mathbf{q}_{\omega_2}$ can be found for $\mathbf{e}_1$ and $\mathbf{e}_2$ by the previous method in section 4.3. The rover for $\mathbf{q}_G^1$ moves π_1 from $\mathbf{e}_1$, and the rover for q_2 moves

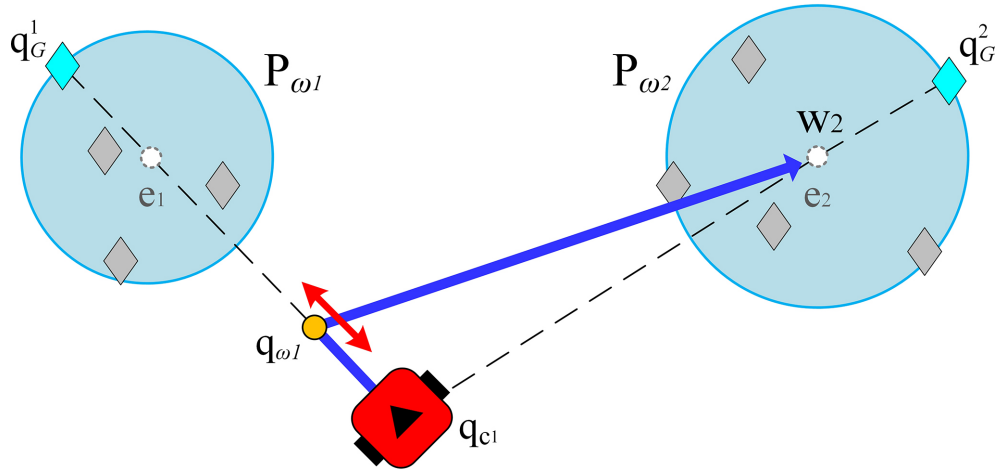


Figure 4.4: Path planning of the carrier for two clusters, $\mathbf{P}_{\omega_1}$ and $\mathbf{P}_{\omega_2}$, having multiple tasks.

π_2 from $\mathbf{e}_2$. Therefore, this time difference also should be compensated. The equation is formulated by using (4.14):

$$\frac{\xi - \Delta d + \pi_1}{v_r^{max}} = f(\mathbf{q}_{\omega_1}, \mathbf{q}_{\omega_2}) + t_u + \frac{\pi_2}{v_r^{max}}, \tag{4.20}$$

where $\xi = \|\mathbf{q}_{c_1} - \mathbf{e}_1\|$. As the number of clusters increases, the earlier deployment location is likely to move toward the carrier. Finally, as shown in Figure 4.5, the carrier should instead deploy rovers at their maximum range d_{deploy} unless the next cluster is the last. In any case, the rovers would have reached the task locations by the time the carrier arrives latter location.

In the same way, the carrier deploy rovers at their maximum range d_{deploy} in 3D space as demonstrated in Figure 4.6. The carrier should stop near $\mathbf{P}_{\omega_\alpha}$ first, then go to $\mathbf{P}_{\omega_{\alpha+1}}$. Let the center of $\mathbf{P}_{\omega_\alpha}$, $\mathbf{P}_{\omega_{\alpha+1}}$, and the current location of

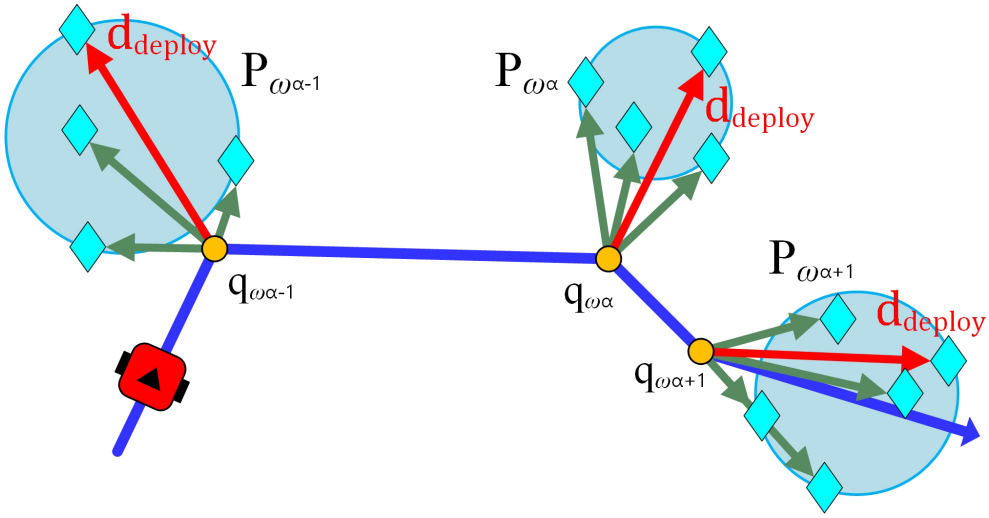


Figure 4.5: In earlier cluster, the carrier should deploy rovers at maximum distance from the farthest task in the cluster

the carrier be $(x_\alpha, y_\alpha, z_\alpha)$, $(x_{\alpha+1}, y_{\alpha+1}, z_{\alpha+1})$, and $(x_{c_i}, y_{c_i}, 0)$ respectively. First, the line segment between the carrier and $(x_{\alpha+1}, y_{\alpha+1}, 0)$ which is the projected point of $(x_{\alpha+1}, y_{\alpha+1}, z_{\alpha+1})$ is found as follows:

$$y = \frac{y_{\alpha+1} - y_{c_i}}{x_{\alpha+1} - x_{c_i}}(x - x_{c_i}) + y_{c_i}, \tag{4.21}$$

where $min(x_{c_i}, x_{\alpha+1}) \leq x \leq max(x_{c_i}, x_{\alpha+1})$. Next, the another line segment which is perpendicular to (4.21) and crosses $(x_\alpha, y_\alpha, 0)$ is found as follows:

$$y = \frac{x_{c_i} - x_{\alpha+1}}{y_{\alpha+1} - y_{c_i}}(x - x_\alpha) + y_\alpha. \tag{4.22}$$

Then, the deployment location $\mathbf{q}_{\omega_\alpha}$ can be found as a dot on (4.22). To minimize the travel distance of the carrier, we find Δd which satisfies the following equation:

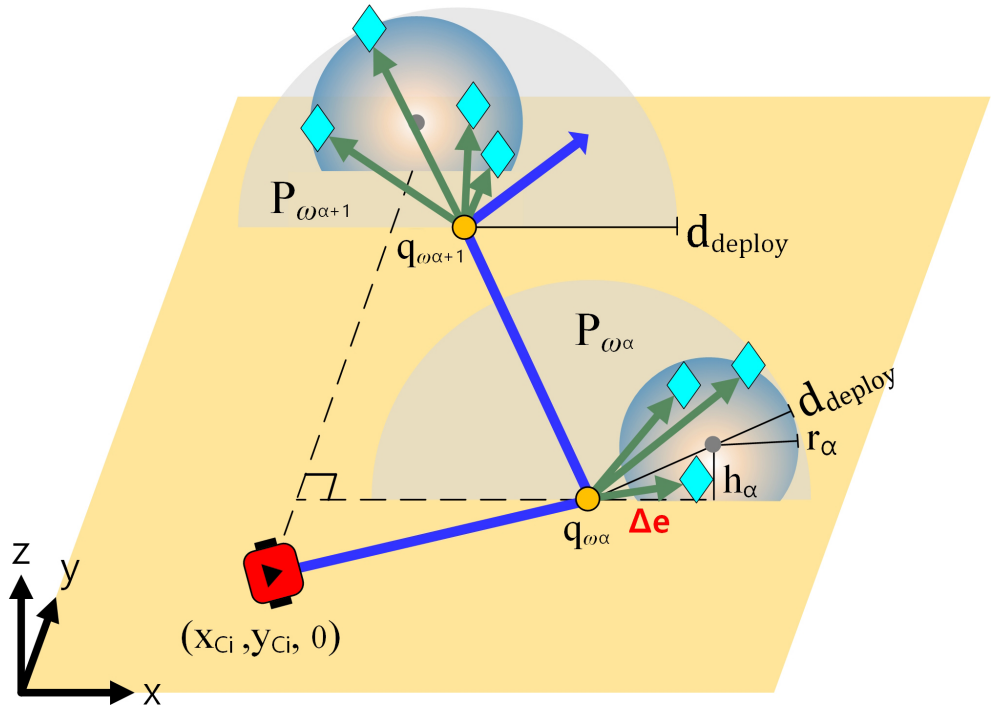


Figure 4.6: Calculation of deployment locations in 3D space. The locations are calculated by using the maximum traveling distance of the rover, the size of the cluster, and the direction to the next cluster.

$$(\Delta d)^2 + z_\alpha^2 = (d_{deploy} - \pi_\alpha)^2, \tag{4.23}$$

so that the distance from $\mathbf{q}_{\omega_\alpha}$ to the farthest point in $\mathbf{P}_{\omega_\alpha}$ is the same as the maximum traveling distance of the rover, d_{deploy}. If a diameter of a cluster is longer than d_{deploy}, Δd in (4.23) cannot be solved because $z_\alpha > (d_{deploy} - \pi_\alpha)$. Therefore, the deployment point cannot be acquired.

The overall path planning procedure using Algorithm 3.1 in section 3.3 is presented in Algorithm 4.1 and 4.2. First, an initial set of clusters is created so that radii of all clusters are shorter than d_{deploy}. For every step the clusters are

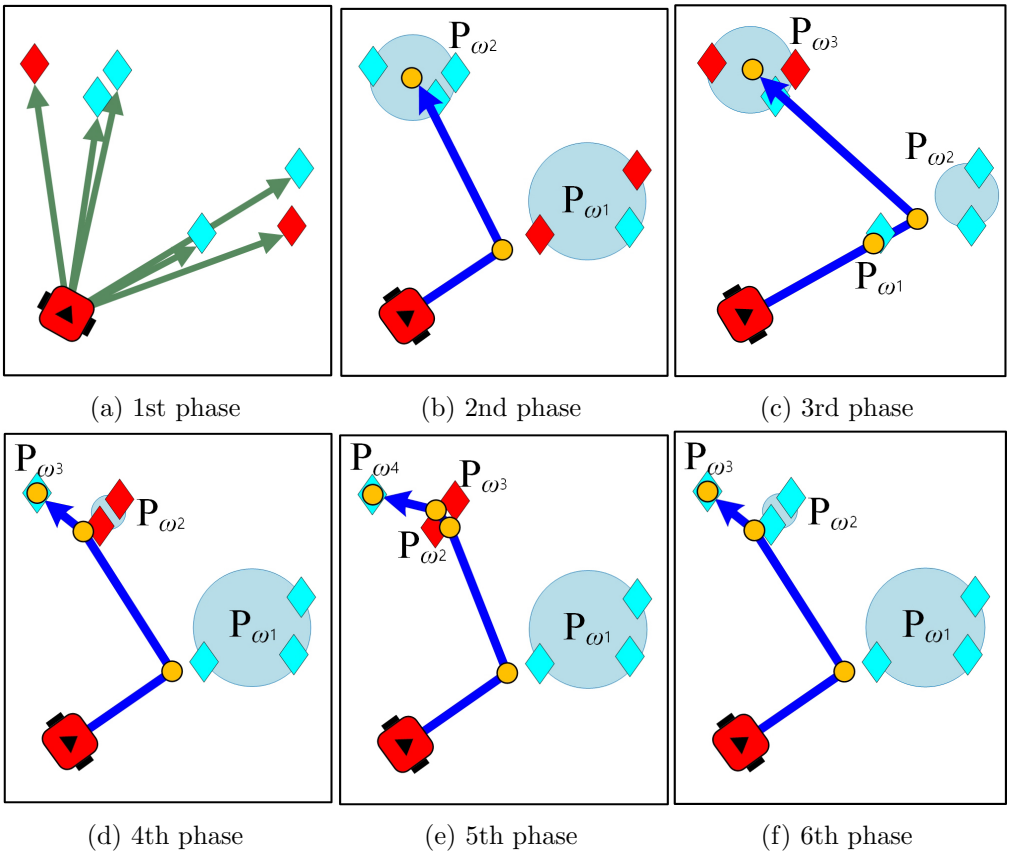
arranged, the deployment path and its time for deployment can be computed. However, the generated path often crosses over itself which result in the inefficiency of the path. Inspired by two-opt algorithm [66], two divided clusters are re-ordered to find the best path among them. Then the computed deployment time can be compared with the previous one so that the faster one is chosen. If the computed deployment time is faster than the previously computed time, one of the previously divided cluster is divided recursively. Otherwise, the previously divided two clusters are merged again. Therefore, all tasks are divided and clustered until the overall time for deployment can be no longer reduced. Figure 4.7 gives an example procedure of the proposed deployment algorithm.

Algorithm 4.1: Deployment path generator

Input: carrier's initial location: $(x_{c_1}, y_{c_1}, 0, \theta_c)$,
Task information: $\mathcal{D}$
Output: Deployment path: Ω, Elapsed Time: T_{min}
`/* Initialization */`
$T_{min} \leftarrow T_{max}$, $Idx \leftarrow 1$, $Num_{cluster} \leftarrow 1$
$I_{cluster} \leftarrow \mathcal{D}$
`/* Initial clustering */`
while $Idx \leq Num_{cluster}$ **do**
  $R_{cluster} \leftarrow$ GET_RADIUS(Idx)
  if $R_{cluster} > d_{deploy}$ **then**
    $I_{cluster} \leftarrow$ DIVIDE_CLUSTER($I_{cluster}, Idx$)
    $Num_{cluster} \leftarrow Num_{cluster} + 1$
  else
    $Idx \leftarrow Idx + 1$
$Idx \leftarrow 1$
`/* Initial path generation */`
$\Omega \leftarrow$ GET_DEPLOY_PATH($I_{cluster}$)
$T_{min} \leftarrow$ CALC_TIME(Ω)

Algorithm 4.2: Deployment path generator (continue)

```
   /* Compare and update of deployment path                          */
 1 while Idx ≤ Num_cluster do
       /* A cluster consists of only one task cannot be divided
          */
 2     Q_cnt ← COUNT_TASKS(I_cluster, Idx)
 3     if Q_cnt = 1 then
 4         Idx ← Idx + 1
 5         continue
 6     I_cluster ← DIVIDE_CLUSTER(I_cluster, Idx)
       /* Generate a path                                            */
 7     Num_cluster ← Num_cluster + 1
 8     Ω_cand ← GET_DEPLOY_PATH(I_cluster)
 9     T_cand ← CALC_TIME(Ω_cand)
       /* Generate alternative path from the swapped cluster         */
10     I_cluster_temp ← SWAP_CLUSTERS(I_cluster, Idx)
11     Ω_cand_temp ← GET_DEPLOY_PATH(I_cluster_temp)
12     T_cand_temp ← CALC_TIME(Ω_cand_temp)
       /* Compare and choose                                         */
13     if T_cand_temp < T_cand then
14         T_cand ← T_cand_temp
15         Ω_cand ← Ω_cand_temp
16         I_cluster ← I_cluster_temp
17     if T_cand ≤ T_min then
18         T_min ← T_cand
19         Ω ← Ω_cand
20     else
21         I_cluster ← MERGE_CLUSTERS(I_cluster, Idx)
22         Num_cluster ← Num_cluster − 1
23         Idx ← Idx + 1
   /* Return result                                                  */
24 return Ω, T_min
```



(a) 1st phase (b) 2nd phase (c) 3rd phase

(d) 4th phase (e) 5th phase (f) 6th phase

Figure 4.7: Example procedure of the proposed algorithm for deployment (a) Initial state (b) Two clusters are created by finding two farthest tasks (red diamonds) in (a), and the corresponding path is generated (c) Cluster $\mathbf{P}_{\omega_1}$ in (b) is divided again, and the deployment path is also acquired (d) The path is overwritten by newly created path (e) The clusters and the path are updated again. However this result is not chosen (f) Final path for deployment is determined if there is no more cluster which can be divided.

In Figure 4.7a, among all tasks, two farthest tasks are found as red circles. Since the distance between two tasks is longer than d_{deploy}, two clusters are initially created and deployment path is acquired as in Figure 4.7b. The earlier cluster

$\mathbf{P}_{\omega_1}$ in Figure 4.7b is again divided as in Figure 4.7c. However, assuming the duration of the newly acquired path is slower than the duration of the previous one, two divided clusters are merged again. Then another two farthest tasks are found in the next cluster. The deployment path is updated in Figure 4.7d, however, the path is not updated in Figure 4.7e according to the result. Finally, there is no cluster left to try in Figure 4.7f, and the path, which is the same as in Figure 4.7d, is chosen.

4.5 Simulation result

In this section, we demonstrate the simulation result of the proposed deployment method.

4.5.1 Simulation setup

First, the simulation setup is presented which will be used in the rest of all simulation part. The specification of the computer used for simulation is shown in Table 4.1. The simulation program runs in Matlab, and is executed on a computer with dual-core 2.90GHz Intel Core i5-5287U CPU, 8GB RAM, and Windows 8.1 64bit operating system.

In the program, parallel processing is not used. However, the implementation of the deployment algorithm is partially improved in speed of processing

Table 4.1: Specification of the simulation computer

Processor	Intel Core i5-5287U dual-core 2.90GHz
Memory	8GB DDR3
OS	Windows 8.1 (64bit)

from the previous study [60]. The path is generated varying several conditions. First, to investigate feasibility, a few tasks or a few rovers are manually located based on a small number of simple scenarios. Then, according to more complex scenarios, the locations of the tasks or those of the rovers are randomly generated and those paths are computed.

4.5.2 Deployment scenarios in 2D space

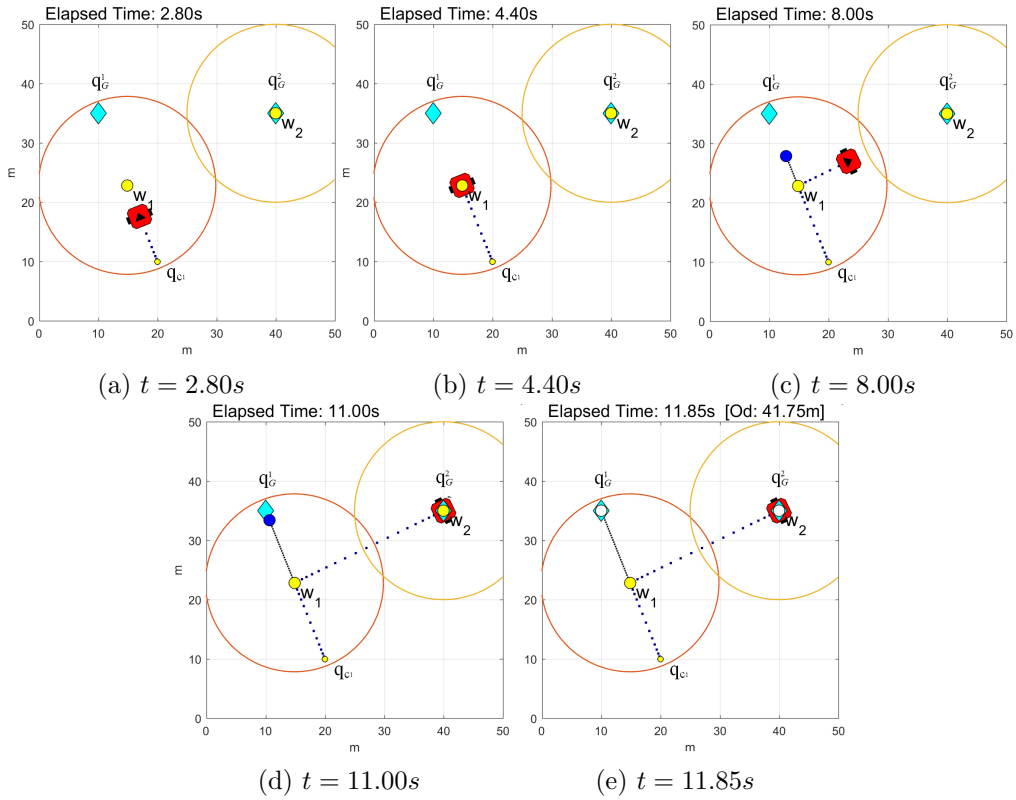
Deployment for two tasks

Figure 4.8 demonstrates the deployment simulation for two tasks. Initial location of the carrier is $(20, 10)$, and initial locations of two tasks are respectively $(10, 35)$ and $(40, 35)$ in $50m \times 50m$ space. All the parameters used for this simulation is listed in Table 4.2.

Table 4.2: Initial parameters in deployment for two tasks

Specification	carrier	$v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 5.0m/s^2$ $t_u = 1.0s$
	rover	$v_r^{max} = 2.0m/s$, $d_{deploy} = 15.0m$
Location	carrier	$\mathbf{q}_{c_1} = (20, 10)$
	tasks	$\mathbf{q}_G^1 = (10, 35)$, $\mathbf{q}_G^2 = (40, 35)$

First, the carrier moves to the first deployment location $\mathbf{q}_{\omega_1}$. As soon as the carrier arrives at $\mathbf{q}_{\omega_1}$, the carrier unloads a rover. After unloading is done, the carrier moves to the next deployment location $\mathbf{q}_{\omega_1}$. At the same time, the deployed rover moves to the first task location which is presented as light-blue diamond in Figure 4.8. In Figure 4.8d, the carrier is arrived at $\mathbf{q}_{\omega_2}$, and unloading the second rover. Finally, in Figure 4.8e, two rovers are unload and reached at their task locations respectively.

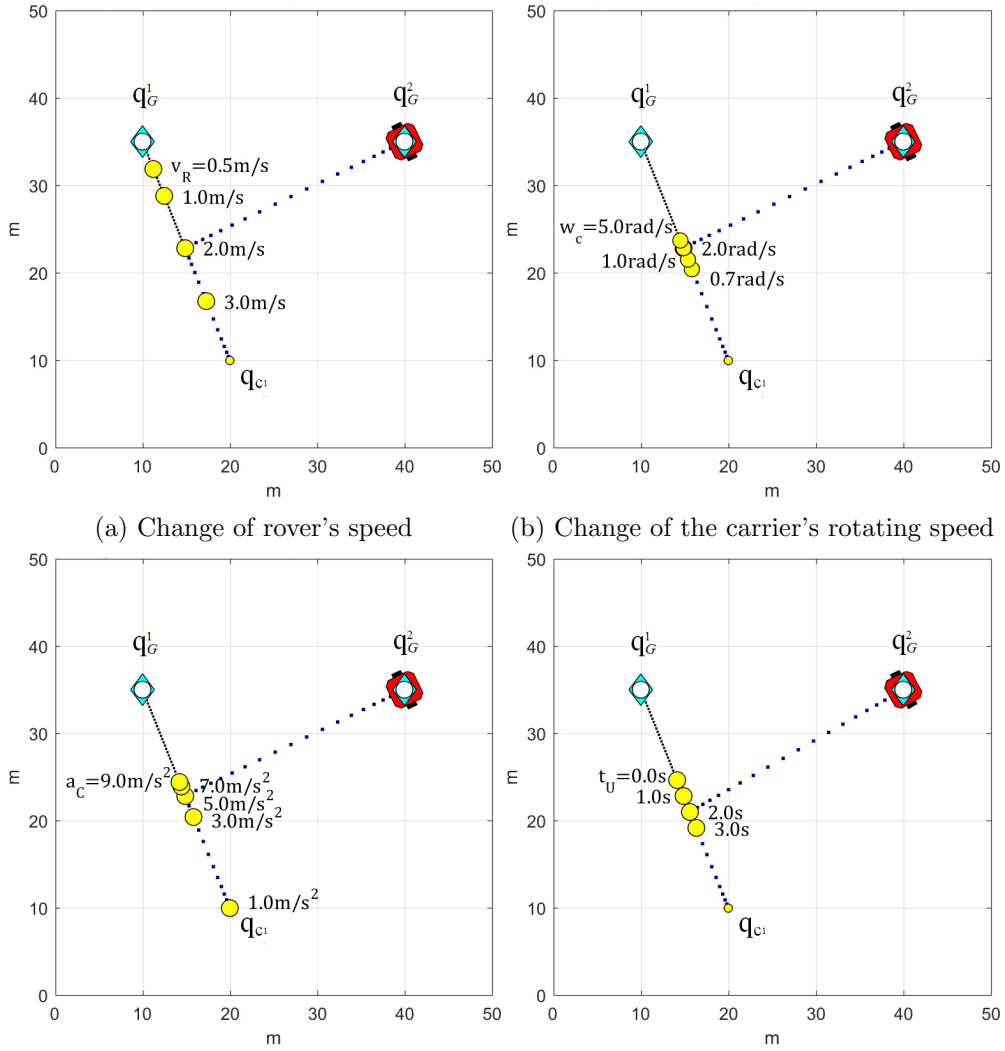


(a) $t = 2.80s$ (b) $t = 4.40s$ (c) $t = 8.00s$

(d) $t = 11.00s$ (e) $t = 11.85s$

Figure 4.8: Example procedure of rovers deployment for two tasks in $50m \times 50m$ space ($v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 5.0m/s^2$, $t_u = 1.0s$, $v_r^{max} = 2.0m/s$, and $d_{deploy} = 15.0m$) (a) Carrier is moving to the first deployment location (b) The carrier arrives and deploys a rover (c) Two robots are moving to their destination (d) The carrier arrives the second deployment location (e) All rovers are deployed and arrived the locations

Figure 4.9 shows how the deployment location is adjusted according to the change of each parameter. For flexible adjustment of deployment locations in the maximum traveling distance of rover, we set up $d_{deploy} = 100.0m$. The other conditions except for the changing condition are the same as in Figure 4.8, and the same deployment path is drawn in Figure 4.9.



(a) Change of rover's speed

(b) Change of the carrier's rotating speed

(c) Change of the carrier's acceleration/deceleration

(d) Change of the carrier's unloading duration

Figure 4.9: Effect of parameter change on the deployment result. Deployment location is adjusted according to the change of each parameter.

In Figure 4.9a, if v_r^{max} increases, the deployment location becomes closer to the carrier. In the similar way, the deployment location also becomes closer to the carrier as t_u increases in Figure 4.9d. On the other hand, if w_c or a_c increase,

the carrier needs less time to move to the next location after deploying the first rover. Therefore, the deployment location becomes closer to the first task location. In Figure 4.9b, if w_c is $0.6rad/s$ or less, the deployment location becomes the point between two tasks since deploying two rovers in the location becomes faster than deploying one by one. In Figure 4.9c, in case $a_c = 1.0m/s^2$, the deployment location becomes $\mathbf{q}_{c_1}$, which means the carrier should immediately deploy the first rover at the beginning.

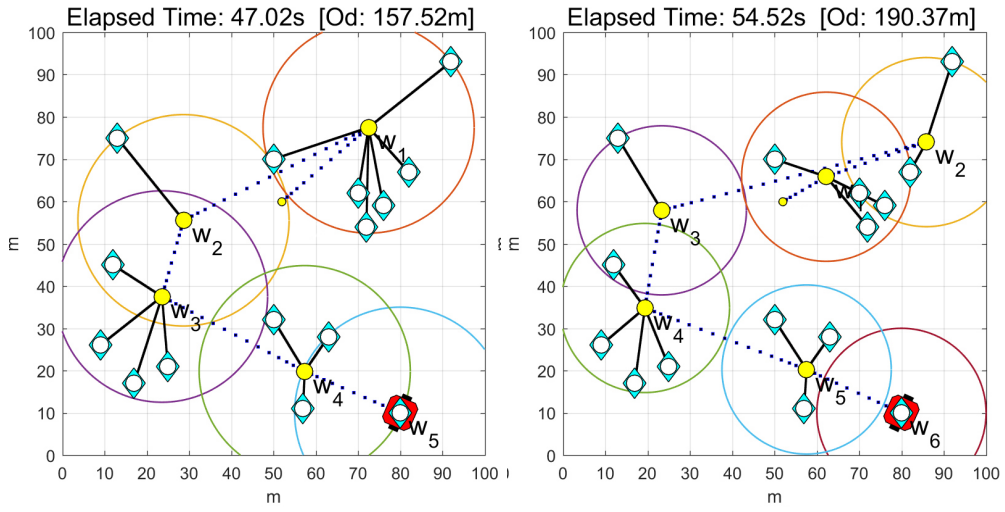
Deployment for 15 tasks

Second deployment scenario is shown in Figure 4.10. There are 15 tasks in $100m \times 100m$ space. All the parameters used for this simulation is listed in Table 4.3.

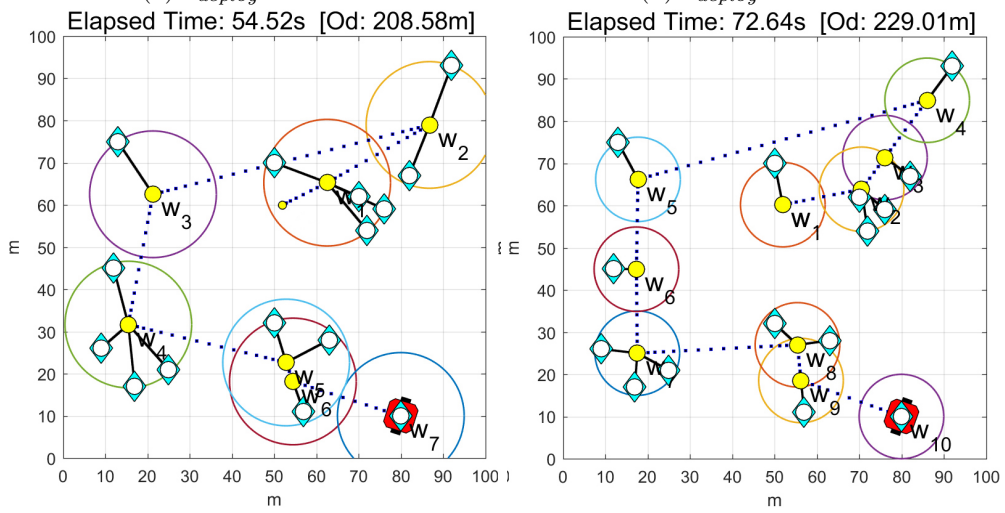
Table 4.3: Initial parameters in deployment for 15 tasks

Specification	carrier	$v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$ $t_u = 4.0s$
	rover	$v_r^{max} = 1.0m/s$, $d_{deploy} = 10.0 - 25.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (52, 60)$
	tasks	$\mathbf{q}_G^1 = (57, 11)$, $\mathbf{q}_G^2 = (76, 59)$, $\mathbf{q}_G^3 = (17, 17)$ $\mathbf{q}_G^4 = (13, 75)$, $\mathbf{q}_G^5 = (9, 26)$, $\mathbf{q}_G^6 = (50, 70)$ $\mathbf{q}_G^7 = (82, 67)$, $\mathbf{q}_G^8 = (70, 62)$, $\mathbf{q}_G^9 = (50, 32)$ $\mathbf{q}_G^{10} = (80, 10)$, $\mathbf{q}_G^{11} = (25, 21)$, $\mathbf{q}_G^{12} = (92, 93)$ $\mathbf{q}_G^{13} = (12, 45)$, $\mathbf{q}_G^{14} = (63, 28)$, $\mathbf{q}_G^{15} = (72, 54)$

The maximum traveling distance of rover, d_{deploy} varies from $25.0m$ to $10.0m$. Figure 4.10a is the result when $d_{deploy} = 25.0m$. The tasks are initially divided into four clusters, and then one of them is divided again until there are five clusters. This deployment takes $47.02s$, and the carrier travels



(a) $d_{deploy} = 25.0m$ (b) $d_{deploy} = 20.0m$



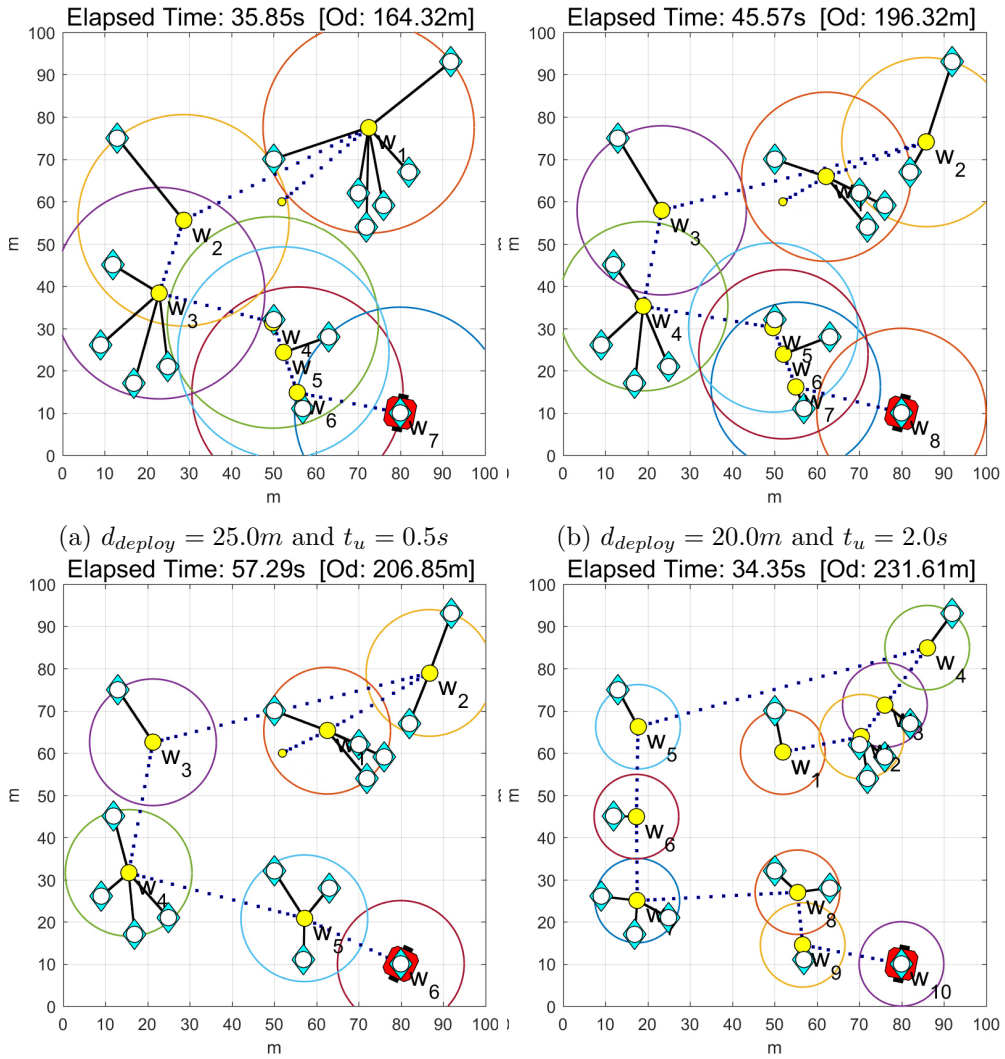
(c) $d_{deploy} = 15.0m$ (d) $d_{deploy} = 10.0m$

Figure 4.10: Deployment scenario with 15 tasks according to the traveling distance constraint of rovers ($v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, $t_u = 4.0s$, and $v_r^{max} = 1.0m/s$.) Initial location of the carrier, $\mathbf{q}_{c_1} = (52, 60)$

157.52m. Next, Figure 4.10b demonstrates the result when $d_{deploy} = 20.0m$. As the rovers which is spreading from $\mathbf{q}_{\omega_1}$ in Figure 4.10a move more than 20.0m, the cluster with respect to $\mathbf{q}_{\omega_1}$ in Figure 4.10a is divided into two clusters, $\mathbf{q}_{\omega_1}$

and $\mathbf{q}_{\omega_2}$ in Figure 4.10b. Compared with the time in Figure 4.10a, the elapsed time for the deployment increases to $54.52s$ due to the increase of travel distance of the carrier. In Figure 4.10c, $d_{deploy} = 15.0m$ where one cluster is added from Figure 4.10b. The length of the carrier's path slightly increases as the carrier should move closer to each task (for example, see $\mathbf{q}_{\omega_3}$ in Figure 4.10b and 4.10c). Lastly, Figure 4.10d shows the result when $d_{deploy} = 10.0m$. Since d_{deploy} is tightly limited than other cases, the number of deployment locations increases to ten locations from five locations in Figure 4.10a, and the carrier's travel distance also increases to $229.01m$. As the task in first cluster is near the initial location of the carrier, the carrier immediately deploys first rover.

Based on the result in Figure 4.10, the unloading time t_u varies by $0.5s$ for each time to see if there is any change of clustering. The result is shown in Figure 4.11. Figure 4.11a shows the different result from Figure 4.10a. Since t_u gets smaller, the cluster for $\mathbf{q}_{\omega_4}$ in Figure 4.10a is divided into three clusters in Figure 4.11a. While the travel distance of the carrier increases, overall duration for deployment decreases. The similar change is observed in Figure 4.11b. As t_u becomes half, the carrier visits more locations for deployment in the latter part of moving. Consequently, the travel distance of the carrier increases to $196.32m$. On the other hand, in Figure 4.11c, t_u increases by $0.5s$. Therefore, two clusters $\mathbf{q}_{\omega_5}$ and $\mathbf{q}_{\omega_6}$ in Figure 4.10c are merged into one cluster in Figure 4.11c, to reduce the duration for unloading. In Figure 4.11d, there is no change in clusters even the unloading time is set as zero. However, the elapsed time for deployment is reduced. Even if the unloading time t_u becomes slow, the clusters cannot merge as d_{deploy} is too short. As a result, in the presented cases,



(a) $d_{deploy} = 25.0m$ and $t_u = 0.5s$

(b) $d_{deploy} = 20.0m$ and $t_u = 2.0s$

(c) $d_{deploy} = 15.0m$ and $t_u = 4.5s$

(d) $d_{deploy} = 10.0m$ and $t_u = 0.0s$

Figure 4.11: Deployment scenario with 15 tasks according to the unloading time of the carrier ($v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, and $v_r^{max} = 1.0m/s$.)

the generated paths seem to be efficient.

4.5.3 Deployment scenarios in 3D space

Deployment for two tasks

For the same two tasks in section 4.5.2, we give height to the tasks. The parameters for this simulation is shown in Table 4.4. As a result, the deployment

Table 4.4: Initial parameters in deployment for two tasks

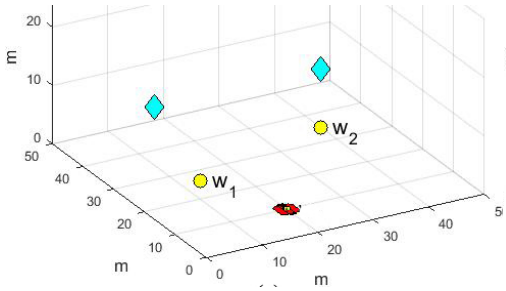
Specification	carrier	$v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 5.0m/s^2$ $t_u = 1.0s$
	rover	$v_r^{max} = 2.0m/s$, $d_{deploy} = 15.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (20, 10, 0)$
	tasks	$\mathbf{q}_G^1 = (10, 35, 10)$, $\mathbf{q}_G^2 = (40, 35, 10)$

procedure is demonstrated in Figure 4.12. First, the carrier is located in its initial location in Figure 4.12a. In Figure 4.12b, the carrier approaches to first deployment location $\mathbf{q}_{\omega_1}$. As the carrier arrives at $\mathbf{q}_{\omega_1}$, the first rover is deployed and it begins to fly in Figure 4.12c. After finishing all deployment, two rovers approach their assigned locations in Figure 4.12d. Finally, two rovers arrive the locations simultaneously. From this simulation, we verify the optimality of the proposed deployment method for arbitrary two tasks.

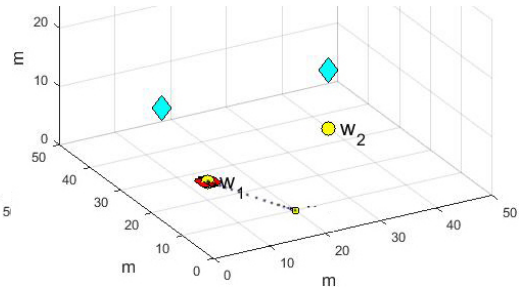
Deployment for six tasks

The example of more complex scenario is also tried with six tasks. The parameters for this simulation is listed in Table 4.5. The deployment procedure is demonstrated in Figure 4.13. In the figure, the spheres imply the maximum traveling distance of the rover from each deployment location.

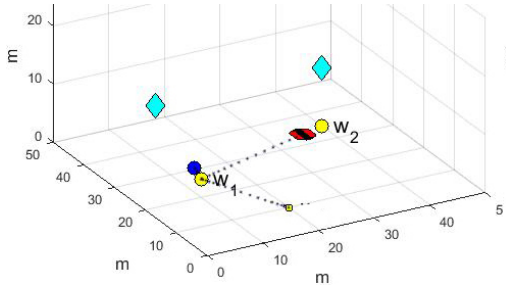
According to d_{deploy}, six tasks are separated into six clusters. The carrier



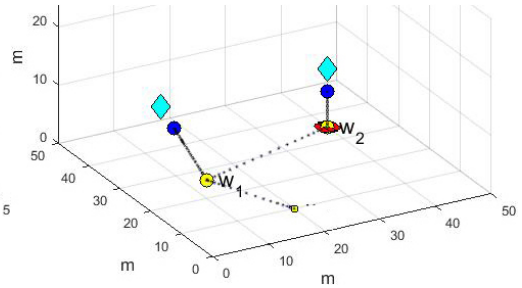
(a) Initial state



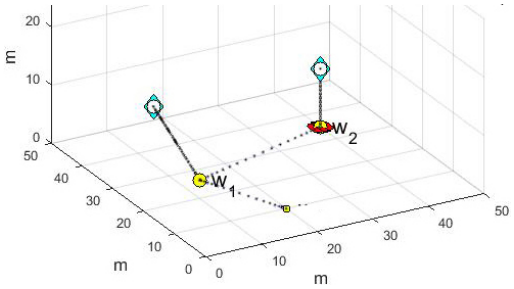
(b) Carrier approaches to $\mathbf{q}_{\omega_1}$



(c) Carrier approaches to $\mathbf{q}_{\omega_2}$, and first rover moves to first task location



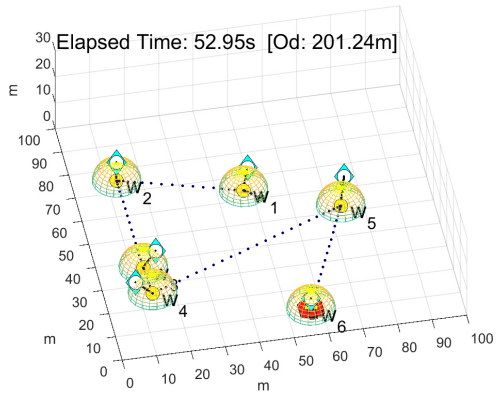
(d) Two rovers are approaching



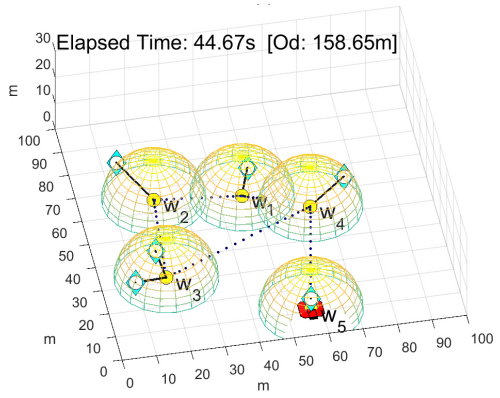
(e) All rovers reach their assigned locations

Figure 4.12: Deployment procedure for two tasks in 3D space

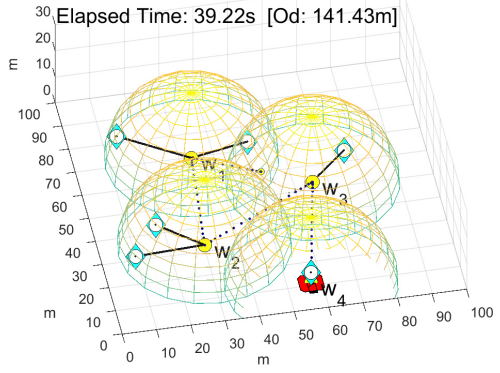
travels $201.24m$, and it takes $52.95s$ for all the rovers reach task locations. Next, the maximum traveling distance increases to $15.0m$ in Figure 4.13b. As a result, two tasks with respect to $\mathbf{q}_{\omega_3}$ and $\mathbf{q}_{\omega_4}$ in Figure 4.13a are merged into one cluster. In addition, both the travel distance of the carrier and the total duration of time for deployment decreases. Figure 4.13c shows the result when



(a) $d_{deploy} = 7.0m$



(b) $d_{deploy} = 15.0m$



(c) $d_{deploy} = 25.0m$

Figure 4.13: Deployment example for six tasks in $(100m \times 100m \times 30m)$. We set $v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, $t_u = 4.0s$, and $v_r^{max} = 1.0m/s$

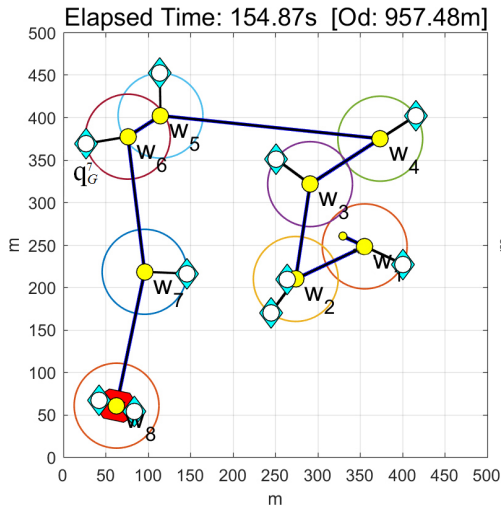
$d_{deploy} = 25.0m$. In the same manner, both the distance of the carrier and the total duration also decreases, and another two tasks are merged into one cluster. By using the proposed method, the efficient path generation for deployment is shown.

4.5.4 Deployment in a dynamic environment

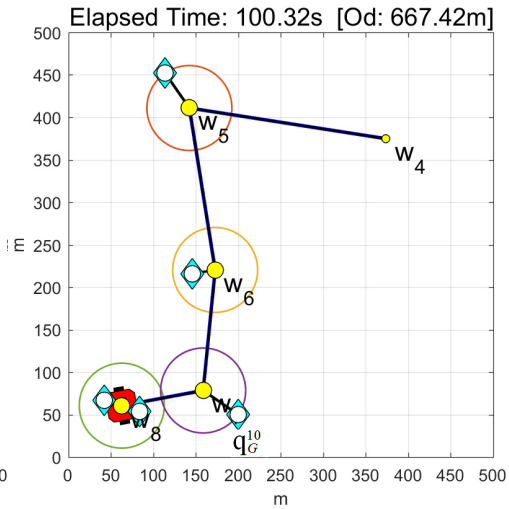
The proposed methods for deployment is used in initial state, before the beginning of the mission. However, in practice, a dynamic environment should be considered, where new task is created, existing task disappears, and rover is lost or disconnected. Therefore the simulation is conducted based on this criteria. Figure 4.14 demonstrates a deployment example in $500m \times 500m$ space, when the set of tasks changes during the deployment procedure. In Figure 4.14a, the deployment path is initially generated for ten tasks first. Then, while a rover is deployed at $\mathbf{q}_{\omega_4}$, the original task $\mathbf{q}_G^7$ disappears and new task $\mathbf{q}_G^{10}$ is created. Therefore, the rest of the deployment path is thrown away, and new path is generated from $\mathbf{q}_{\omega_4}$ in Figure 4.14b. If $\mathbf{q}_G^7$ does not exist and $\mathbf{q}_G^{10}$ exists at the beginning, deployment path is generated as in Figure 4.14c. However, since the

Table 4.5: Initial parameters in deployment for six tasks

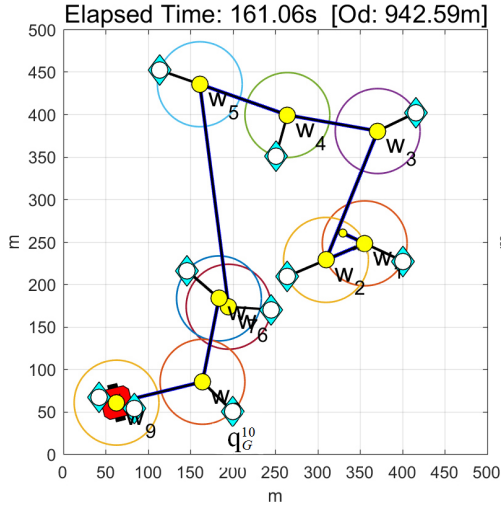
Specification	carrier	$v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$ $t_u = 4.0s$
	rover	$v_r^{max} = 1.0m/s$, $d_{deploy} = 7.0 - 25.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (52, 60, 0)$
	tasks	$\mathbf{q}_G^1 = (57, 11, 4)$, $\mathbf{q}_G^2 = (76, 59, 5)$ $\mathbf{q}_G^3 = (17, 37, 6)$, $\mathbf{q}_G^4 = (13, 75, 7)$ $\mathbf{q}_G^5 = (9, 26, 5)$, $\mathbf{q}_G^6 = (50, 70, 3)$



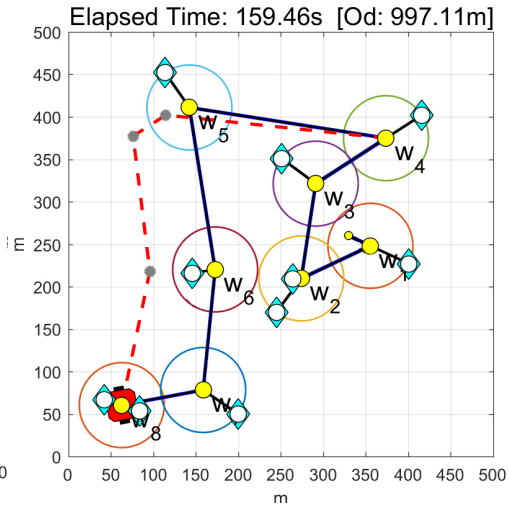
(a) Deployment path generation for initial set of tasks



(b) Deployment path generation for updated set of tasks at $\mathbf{q}_{\omega_4}$



(c) Deployment path generation without $\mathbf{q}_G^7$ in (a) and with $\mathbf{q}_G^{10}$



(d) Final deployment path

Figure 4.14: Dynamic deployment scenario with ten tasks in $500m \times 500m$ space ($v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 3.0m/s^2$, $t_u = 2.5s$, $v_r^{max} = 3.0m/s$, and $d_{deploy} = 50.0m$.)

state changes when the carrier is at $\mathbf{q}_{\omega_4}$, the deployment path before $\mathbf{q}_{\omega_4}$ in Figure 4.14a is used by the carrier. Then new path in Figure 4.14b is used by the carrier, and the final deployment path is shown in Figure 4.14d. The red dashed lines present the old path with $\mathbf{q}_G^7$ and without $\mathbf{q}_G^{10}$. Therefore, if a deployment path is calculated in real-time, the change of tasks can be reflected in the path of the carrier at any time. As a result, the deployment path can be updated in any case so that the path maintains their efficiency in a dynamic environment.

4.6 Performance evaluation

The performance of the proposed algorithm is evaluated in two ways:

1) Computation time;

2) Efficiency of the path.

To generate random conditions, the Monte Carlo method is used since not all conditions can be tested. However, the simulation results are shown for 2D space only because the dimensional difference had shown almost no effects on computation time.

4.6.1 Computation time

To evaluate performance of the proposed algorithms, computation time is measured in $200m \times 200m$ space. (Note that, changing the size of the map is not considered, as it is expected that the size of the map would not affect the computation time.) All locations of the carrier, rovers, and tasks are randomly generated with a uniform distribution, and the number of rovers or tasks varies

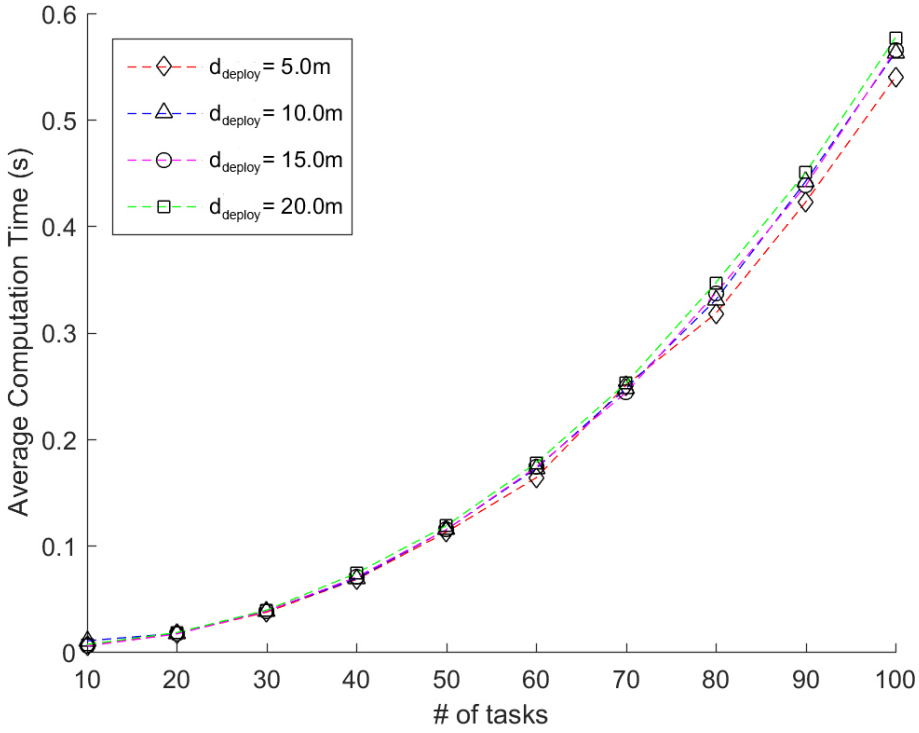


Figure 4.15: Average computation time of proposed deployment algorithm. All parameters and locations are randomly generated adopting the Monte Carlo method

from ten to 100. The maximum traveling distances of rover, d_{deploy} varies from $5.0m$ to $20.0m$. Adopting the Monte Carlo method, the parameters of v_c^{max}, w_c, a_c, t_u, and v_r^{max} are also randomly generated with a normal distribution, with mean $15.0m/s$, $3.0rad/s$, $5.0m/s^2$, $3.0s$, and $3.0m/s$ respectively. The duration is measured 100 times and the average value is acquired under each distinct condition.

The average computation time of the proposed deployment algorithm is depicted in Figure 4.15. As the number of tasks increases, the slope of the average computation time tends to increase. Although the maximum traveling

distance of rover differs, there is almost no deviation in the result. As a result, Figure 4.15 shows that the computations can be done quickly. By using this, near real-time system can be implemented, depending on the number of tasks and sampling time of the system. In addition, reducing the computation time may be possible if two options are considered: 1) implementing the simulation code with faster programming languages such as visual c++; and 2) optimizing the data structure and using parallel processing.

4.6.2 Efficiency of the path

As mentioned in chapter 1, finding optimal solution for the problems requires a lot of computation. Brute-force search [86] over all paths may take hours or even days for tens of tasks or rovers. Therefore, to investigate the efficiency of the proposed algorithms, the computed path is compared with the greedy two-opt solution which is the one of the TSP solving algorithms [53], in $300m \times 300m$ space.

Intuitively, the efficiency of the path is mainly affected by the traveling distance of rovers, and by the ratio of carrier's speed to rovers' speed. Therefore, the distance d_{deploy} varies from $10.0m$ to $40.0m$. The number of tasks and rovers is set from ten to 100, and all locations of them are randomly generated. For the other parameters, the Monte Carlo method is also used with a normal distribution, and each distinct condition is repeated 100 times. The result shown in Figure 4.16 is the relative efficiency of the proposed deployment algorithm, compared to greedy two-opt solution. When $d_{deploy} = 10.0m$, some condition show that the proposed method is even inefficient than greedy two-opt. This

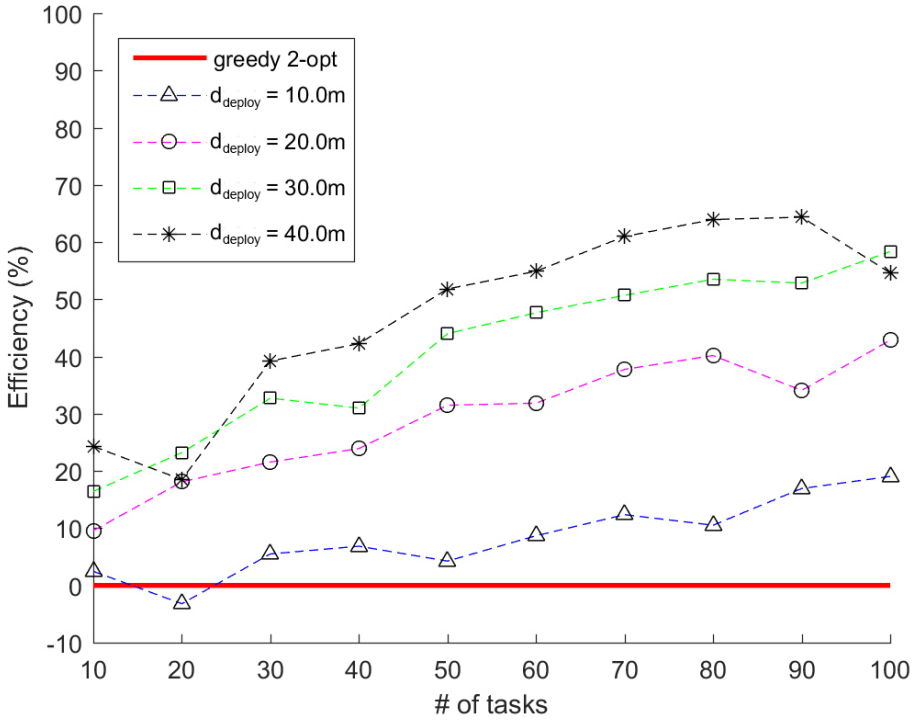


Figure 4.16: Efficiency of the deployment path comparing with the solution from greedy two-opt algorithm, in $300m \times 300m$ space

is resulted from the simple clustering method. However, as d_{deploy} increases, the inefficiency of the cluster dividing method can decrease. As the number of tasks increases, the efficiency tends to increase since more tasks are likely to be merged into a cluster. In the given section, total average of the efficiency values is 25.20%. This efficiency comes from in part by the adjacency of tasks as they can be put into a cluster. Therefore the smaller space may result in the more efficient path, while the bigger space may result in the similar path as the path from greedy two-opt algorithm.

Chapter 5

Collection of a Marsupial Robot Team

In this chapter we find a feasible solution to the path planning of a marsupial robot team for collection. Some parts, including the basic concept, are similar to the method for deployment. However, since the task location reached by the rover during deployment is the starting point of the task, the position of the rover will be changed when the task finishes. Therefore a re-planning is required for collection. The biggest difference between when collecting and deploying rovers is that the remaining energy is different. In other word, it is assumed that all rovers are fully charged and have the same maximum travel distance value, however that the remaining energy of rover is different at the time of collection. Therefore, it is necessary to estimate the common area of a circle having a maximum distance. In addition, the carrier can unload rovers at deployment location and leave immediately, however the carrier may wait rovers

at collection location. As in the previous chapter, we first draw the objective and analyze the computational complexity of the problems for optimal solutions. The work in this chapter is based on the journal article [61].

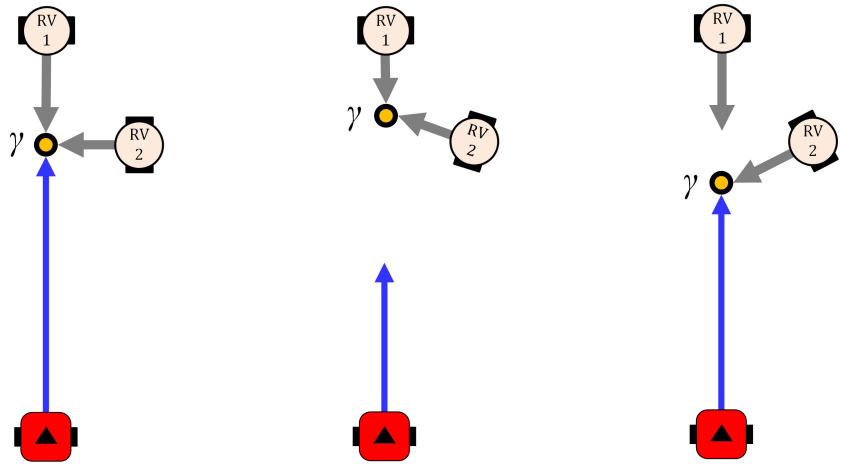
5.1 Problem definition

The objective of the problem is to retrieving all rovers, which have completed the assigned tasks, in minimum time. Let $\mathbf{q}_G$ be the final location to where all rovers return. Then the general objective function that each rover returns by itself is formulated as follows:

$$\begin{aligned} &\text{minimize} \quad \max_{i=1,\ldots,n} (\|\mathbf{q}_G - \mathbf{q}_{r_i}\|) \\ &\text{subject to} \quad \|\mathbf{q}_G - \mathbf{q}_{r_i}\| \leq d^i_{collect} \quad (i = 1, \ldots, n). \end{aligned} \tag{5.1}$$

In case the carrier can meet and retrieve rovers, the duration can be calculated as in chapter 4. The duration of the carrier from the previous collecting location to the next collecting location can be computed by using (4.2). However, overall duration for collecting rovers from the initial location of the carrier is different from (4.3). The reason is that the carrier deploying rovers does not have to wait after finishing deployment at one location, whereas the carrier collecting rovers may have to wait for rovers coming. There are three cases that the carrier meets rovers at a collection location, as shown in Figure 5.1:

- Case I: Both the carrier and the last rover in the cluster arrive at the collection location simultaneously.
- Case II: All rovers in the cluster arrive at the collection location before



(a) Case I (b) Case II (c) Case III

Figure 5.1: Three cases that the carrier meets rovers at a collection location.

the carrier.

- Case III: The carrier arrives at the collection location before the last rover in the cluster.

In case I and II, the carrier does not wait for loading rovers. However, in case III, the carrier should wait for rovers approaching. (Note that the case III can be modified to either the case I or II by repositioning the collection location. However it does not guarantee that the total collection time will be reduced.) Therefore, the total collection duration η_δ, where $1 \leq \delta \leq \beta$, from the initial location is deducted as follows:

$$
\begin{aligned}
\eta_1 &= max(f(\mathbf{q}_{\gamma_0}, \mathbf{q}_{\gamma_1}), g(\mathbf{q}_{\gamma_1})) + t_l, \\
\eta_2 &= max(\eta_1 + f(\mathbf{q}_{\gamma_1}, \mathbf{q}_{\gamma_2}), g(\mathbf{q}_{\gamma_2})) + t_l, \\
\eta_3 &= max(\eta_2 + f(\mathbf{q}_{\gamma_2}, \mathbf{q}_{\gamma_3}), g(\mathbf{q}_{\gamma_3})) + t_l, \\
&\vdots \\
\eta_{\delta-1} &= max(\eta_{\delta-2} + f(\mathbf{q}_{\gamma_{\delta-2}}, \mathbf{q}_{\gamma_{\delta-1}}), g(\mathbf{q}_{\gamma_{\delta-1}})) + t_l, \\
\eta_\delta &= max(\eta_{\delta-1} + f(\mathbf{q}_{\gamma_{\delta-1}}, \mathbf{q}_{\gamma_\delta}), g(\mathbf{q}_{\gamma_\delta})) + t_l,
\end{aligned}
\tag{5.2}
$$

where $\mathbf{q}_{\gamma_0} = \mathbf{q}_{c_1}$. Finally, the objective of the problem using a carrier can be formulated as the following optimization problem:

$$
\Gamma^\star = \underset{\Gamma \in \mathbb{R}^3}{\arg\min} \Big(\max(\eta_1, \eta_2, \ldots, \eta_\beta) \Big) \tag{5.3}
$$

5.2 Complexity analysis

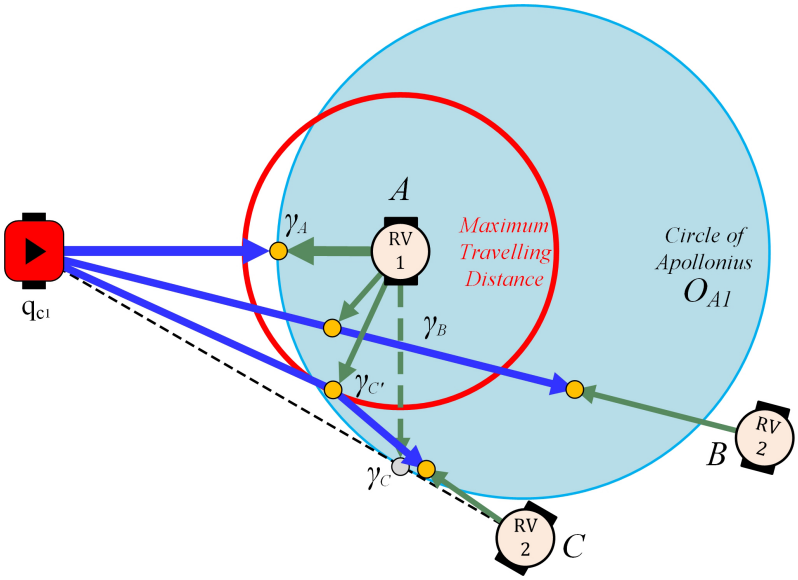
The computational complexity for collecting of a marsupial robot team can be done in the same way as the computational complexity analysis for the deployment of the team. In this case, the computational complexity is bounded above by $O(g^n)$ for m rovers. This brute-force methods for optimality require a tremendous computation. A simpler method is that all rovers to be collected are in place and the carrier visits all rovers, which is the same as the TSP problem. However, this method will inevitably require more execution time. Therefore, heuristic but efficient algorithms are needed.

5.3 Optimal collection path planning for two rovers

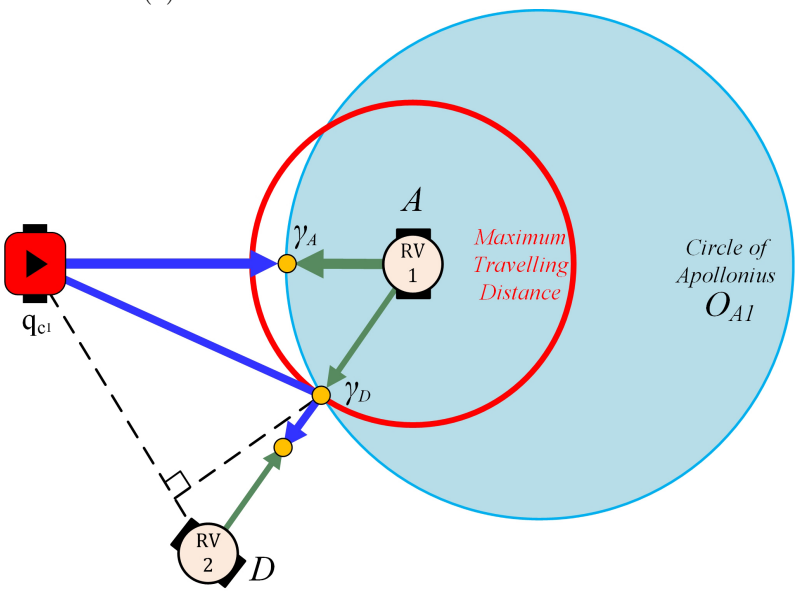
Once the rovers finish their tasks, they should be collected for recharging and reusing. Basically, the approach to collection path planning is similar to deployment path planning. To investigate efficient solution for this collection problem, the simple collecting case of two rovers in 2D space is analyzed first where the optimal solution can be obtained easily. Then we extend the case to 3D case, two-cluster case, and general case.

5.3.1 Collection for two rovers in 2D space

Consider two rovers at $\mathbf{q}_{r_1}^1$ and $\mathbf{q}_{r_1}^2$, to be collected one by one as given in Figure 5.2. Basically, the carrier should approach and load RV1 in the figure as soon as possible. If the acceleration/deceleration and rotating time of the carrier are neglected, the carrier will meet RV1 at some point on the *circle of Apollonius* O_{A1} which has a given ratio of distances $|v_c^{max}|/|v_r^{max}|$ to the locations of the carrier and $\mathbf{q}_{r_1}^1$. The shortest path where the carrier can meet RV1 is $\mathbf{q}_{\gamma_A}$ in Figure 5.2a. However, depending on the location of the second rover, the first optimal collection location is varied. For example, if the second rover is at B as in Figure 5.2a, the first optimal location can be at any point between two intersection points of $\overline{\mathbf{q}_{c_1}\mathbf{q}_{r_1}^2}$ and O_{A1}. Maybe, the best location for $\mathbf{q}_{\gamma_B}$ will be the point which makes the line $\overline{\mathbf{q}_{r_1}^1\mathbf{q}_{\gamma_B}}$ to be the shortest. If RV2 is at C as in Figure 5.2a, $\mathbf{q}_{\gamma_C}$ is the only one optimal collection location. However, RV1 cannot reach $\mathbf{q}_{\gamma_C}$ due to its limitation of traveling distance, which is presented as a red circle in Figure 5.2a. Therefore, they should instead meet at $\mathbf{q}_{\gamma_C'}$. On the other hand, if RV2 is at D as in Figure 5.2b, there is no



(a) Two cases the carrier collects two rovers



(b) The other case the carrier collects two rovers

Figure 5.2: Optimal path planning of a carrier for two rovers. The collection location is calculated by considering the location of the next rover and the maximum traveling distances of rovers

intersection point between $\overline{\mathbf{q}_{c_1}\mathbf{q}_{r_1}^2}$ and O_{A1}. Therefore, the closest point $\mathbf{q}_{\gamma_D}$ on O_{A1} to $\overline{\mathbf{q}_{c_1}\mathbf{q}_{r_1}^2}$ is chosen for the optimal collection location. Note that, if $90^\circ < \angle\mathbf{q}_{r_1}^1\mathbf{q}_{c_1}\mathbf{q}_{r_1}^2 < 270^\circ$, the optimal collection location again becomes $\mathbf{q}_{\gamma_A}$. The procedure to find collection locations for two rovers can be summarized as follows:

1) Find a rover farther than another;

2) Find a line segment which connects the carrier and the farther rover;

3) Select the point closest to the found line as the collection location while satisfying both the circle of maximum travel distance and the circle of Apollonius.

When deploying two rovers, the circle of Apollonius is used between two rovers. However, in the case of collecting two rovers, this circle is used between the carrier and the first rover that the carrier approach.

As a next step, the previous optimal locations should be adjusted so that the effects of acceleration/deceleration and rotating time of the carrier are applied. Figure 5.3 shows how the collection location is repositioned from $\mathbf{q}_{\gamma_{1old}}$ to $\mathbf{q}_{\gamma_{1new}}$. Assume that the carrier and RV1 from A arrive at $\mathbf{q}_{\gamma_{1new}}$ at the same time with duration of λ. Then the duration λ is represented as follows:

$$\lambda = f(\mathbf{q}_{c_1}, \mathbf{q}_{\gamma_{1new}}) = \frac{|\theta_c| + |\Delta\theta|}{w_c} + \frac{S}{v_c^{max}} + \frac{v_c^{max}}{a_c}, \tag{5.4}$$

$$\lambda = \frac{\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{1new}}\|}{v_r^{max}}, \tag{5.5}$$

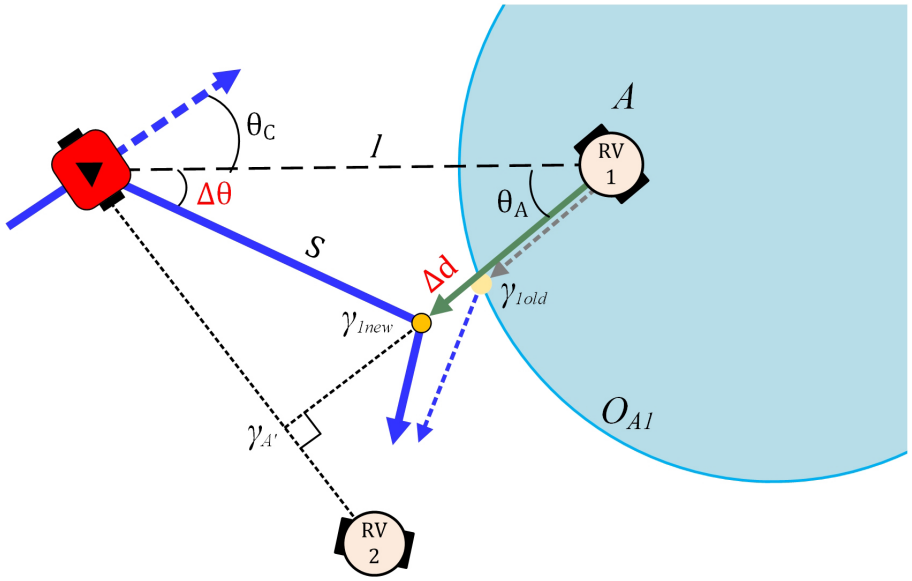


Figure 5.3: Path modification of the carrier. The previous collection location $\mathbf{q}_{\gamma_{1old}}$ is no longer optimal as the location is adjusted to $\mathbf{q}_{\gamma_{1new}}$ to reflect more dynamics of the carrier

in condition that the carrier gets its maximum linear speed as $\|\mathbf{q}_{c_1} - \mathbf{q}_{\gamma_{1new}}\| \geq v_c^{max}/a_c$. Combining (5.4) and (5.5) yields:

$$\frac{|\theta_c| + |\Delta\theta|}{w_c} + \frac{S}{v_c^{max}} + \frac{v_c^{max}}{a_c} = \frac{\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{1new}}\|}{v_r^{max}}. \tag{5.6}$$

And the *law of cosines* yields the following equation:

$$S^2 = l^2 + (\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{1old}}\| + \Delta d)^2 - 2l(\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{1old}}\| + \Delta d)\cos\theta_A. \tag{5.7}$$

On the other hand, in Figure 5.3, the relation between $\Delta\theta$ and Δd can be represented as follows:

$$\tan(90 - \theta_A - \Delta\theta) = \frac{\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{A'}}\| - \|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{A'}}\| - \Delta d}{\|\mathbf{q}_{c_1} - \mathbf{q}_{\gamma_{A'}}\|}. \tag{5.8}$$

Summarizing the equation for $\Delta\theta$:

$$\Delta\theta = 90 - \theta_A - \arctan\Big(\frac{\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{A'}}\| - \|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{A'}}\| - \Delta d}{\|\mathbf{q}_{c_1} - \mathbf{q}_{\gamma_{A'}}\|}\Big). \tag{5.9}$$

Therefore, by replacing $\Delta\theta$ in (5.6) with (5.9), (5.7) can be solved for Δd. To simplify the problem with arc tangent, $\Delta\theta$ can be approximated as follows:

$$\Delta\theta = \frac{(90 - \theta_A)(\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{1old}}\| + \Delta d)}{\|\mathbf{q}_{r_1} - \mathbf{q}_{\gamma_{A'}}\|}. \tag{5.10}$$

Then $\Delta\theta$ in (5.6) is replaced by (5.10), and Δd can be acquired by combining (5.6) and (5.7).

5.3.2 Collection for two rovers in 3D space

By expanding the idea of collection in the previous chapter, the collection for two rovers in 3D space can be established. Assume the rover RV1 is hovering at height z_1, as depicted in Figure 5.4. Then the rover should fly and land on the collection location. At the same time, the carrier should move to the same location for retrieving. Therefore, by equalizing the duration that the carrier moves and the rover moves, the collection location $\mathbf{q}_{\gamma_{1new}}$ can be calculated as follows:

$$\frac{|\theta_c| + |\Delta\theta|}{w_c} + \frac{S}{v_c^{max}} + \frac{v_c^{max}}{a_c} = \frac{\sqrt{\Delta d^2 + z_1^2}}{v_r^{max}}. \tag{5.11}$$

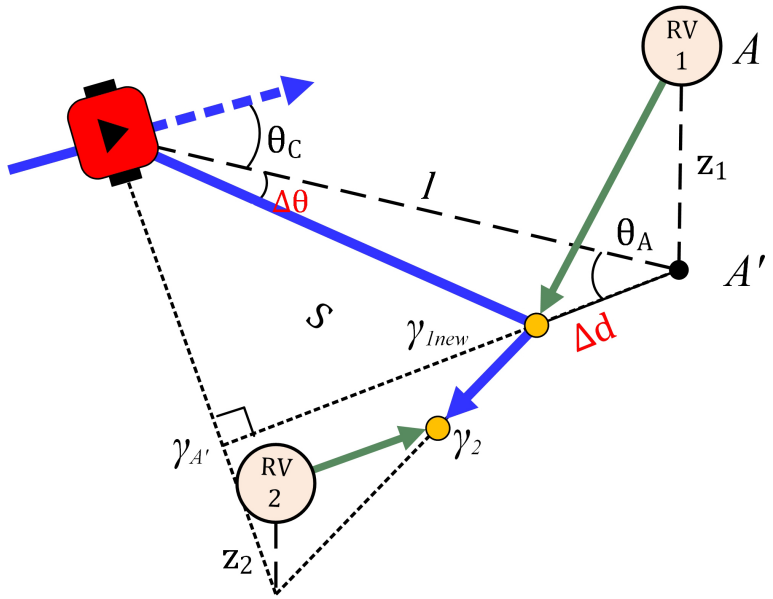
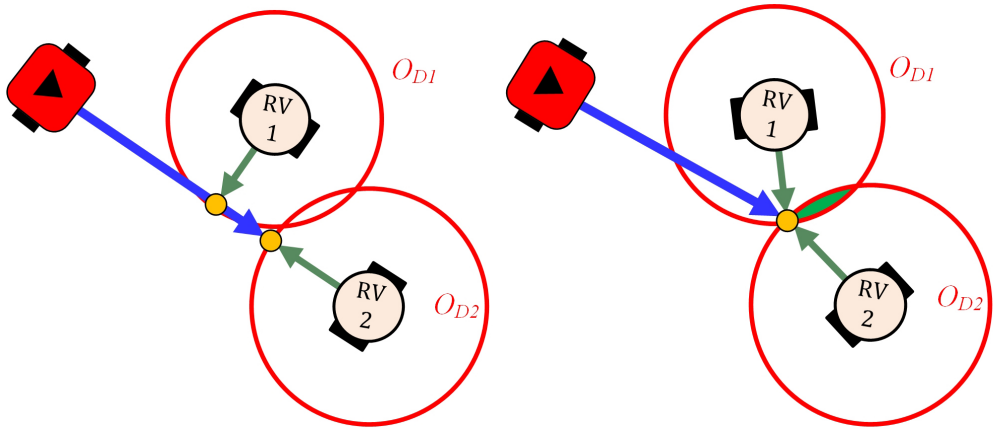


Figure 5.4: Two rovers collection in a 3D space

Once the height factor of a rover is applied to an equation, the other method is basically the same.

5.4 Path planning algorithm of a marsupial robot team for collection

In the same manner as deployment problem in chapter 4, clustering rovers is important to generate an efficient path. Figure 5.5 shows an example of collection of two rovers, which explains how clustering affects the collection result. In Figure 5.5a, two rovers are collected one by one as previously analyzed. On the other hand, two rovers in Figure 5.5b are collected together at one location. If the loading time t_u increases, the latter path becomes more advantageous than the path in Figure 5.5a. (An opposite case may occur if t_u decreases.)



(a) Two rovers are collected one by one and clustering is not considered

(b) Two rovers in one cluster are gathered and collected together

Figure 5.5: Examples of two rovers collection. As the loading time gets longer, collection time for (a) is expected to take longer than the time for (b)

Therefore all rovers for collecting should be divided into several clusters so that some rovers are gathered and collected at once. In this study, the same clustering method used for deployment problem is applied for clustering of collection problem.

To include multiple rovers in a cluster, there must be the intersection area where all relevant rovers can reach. In this manner, all rovers to be collected are initially assigned to one cluster first. Then the cluster is recursively divided into two clusters until there is no cluster which has no intersection area. In β-th cluster for collection $\mathbf{P}_{\gamma_\beta}$, the intersection area I_β is found by using radii and centers of the circles, made by the maximum traveling distances of rovers as below:

$$I_\beta = O^\beta_{D1} \cap O^\beta_{D2} \cap O^\beta_{D3} \cap \ldots\ldots \cap O^\beta_{Dk}. \tag{5.12}$$

Incidentally, calculating the exact intersection area requires a lot of computation [117]. Note that the purpose of this calculating is to find the closest point in the area from the line between the previous collection location and the next collection location of the carrier. In this point of view, the problem finding intersection area can be converted into convex problem because circle is convex and the intersection of convex polygons is convex. Let $\mathbf{q}_{\gamma_{\beta-1}}$ and $\mathbf{q}_{\gamma_{\beta+1}}$ be the previous and the next collection location respectively. As the next collection location $\mathbf{q}_{\gamma_{\beta+1}}$ is not known yet, it is temporarily determined as a midpoint of all intersection points of the circles in the next cluster $\mathbf{P}_{\gamma_\beta}$. Then the desired collection location $\mathbf{q}_{\gamma_\beta}$ is acquired by solving the following convex problem:

$$\begin{aligned} &\text{minimize} && |(x_{\gamma_\beta} - x_{\gamma_{\beta-1}})d_y - (y_{\gamma_\beta} - y_{\gamma_{\beta-1}})d_x| \\ &\text{subject to} && \|\mathbf{q}_{\gamma_\beta} - \mathbf{q}_r^j\| \leq d_{collect}^j, \end{aligned} \tag{5.13}$$

where

$$d_x = x_{\gamma_{\beta+1}} - x_{\gamma_{\beta-1}}, \tag{5.14}$$

$$d_y = y_{\gamma_{\beta+1}} - y_{\gamma_{\beta-1}}, \tag{5.15}$$

and $\mathbf{q}_r^j \in P_{\gamma_\beta}$ for all j. If the acquired location from (5.13) is beyond the end of segment, the alternative collection location again calculated as follows:

$$\begin{aligned} &\text{minimize} && \min(\|\mathbf{q}_{\gamma_\beta} - \mathbf{q}_{\gamma_{\beta-1}}\|, \|\mathbf{q}_{\gamma_\beta} - \mathbf{q}_{\gamma_{\beta+1}}\|) \\ &\text{subject to} && \|\mathbf{q}_{\gamma_\beta} - \mathbf{q}_r^j\| \leq d_{collect}^j. \end{aligned} \tag{5.16}$$

On the other hand, using convex optimization requires more computing time. To reduce the amount of computation, only intersection points of the circles in cluster $\mathbf{P}_{\gamma_\beta}$ can be considered as candidate collection locations $\mathbf{q}_{\gamma_\beta}$. Figure

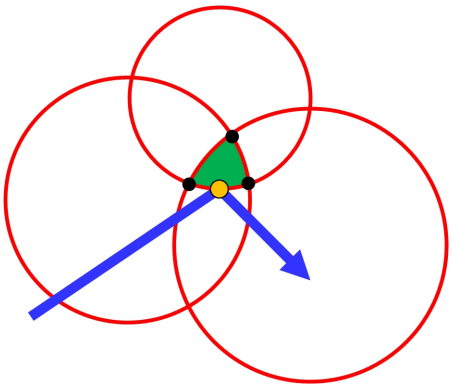


Figure 5.6: Example of a collection location decision. The yellow dot can be found by using convex optimization, however the black dots which are the intersection of the circles can be considered as an alternative collection location to reduce the calculation amount

5.6 shows an example. If the carrier comes from the bottom left as shown by the blue arrow, the convex optimization will set yellow dot to be the collection location. However, since solving the requires a lot of computation, three black dots that intersect the three circles can be considered as alternate locations. The overall algorithm for collection is described in Algorithm 5.1 and 5.2. Basically, the principle of the algorithm is almost the same as Algorithm 4.1 and 4.2. In Algorithm 4.1 and 4.2, the initial clustering is done by finding intersection points in each cluster.

The procedure example of the collection path generation is illustrated in Figure 5.7. In Figure 5.7a, rovers to be collected after finishing their tasks are initially scattered. The red circles show the maximum traveling distances of rovers.

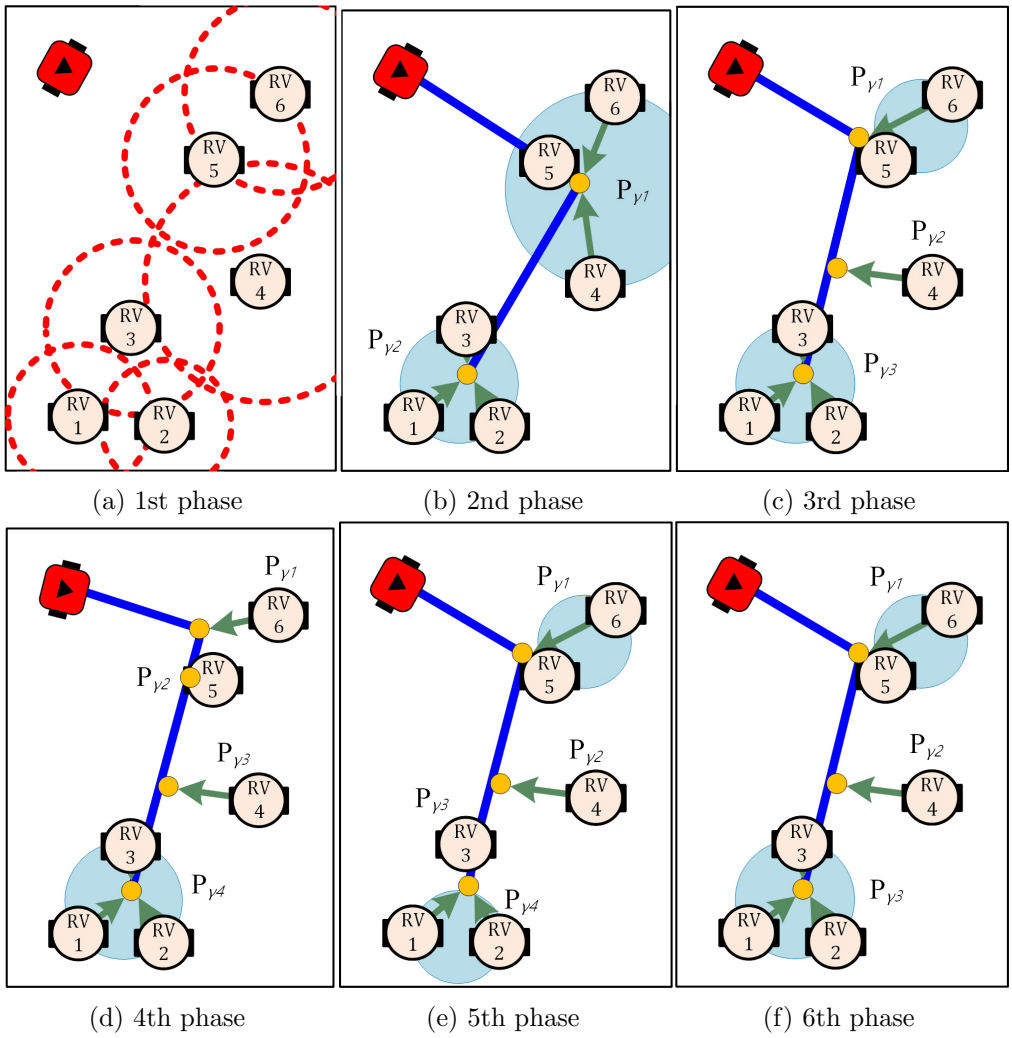
Algorithm 5.1: Collection path generator

Input: carrier's initial location $(x_{c_1}, y_{c_1}, 0, \theta_c)$
rovers' information $\mathcal{R}$
Output: Collection path Γ, Elapsed Time T_{min}

```
/* Initialization                                          */
```
$T_{min} \leftarrow T_{max}$, $Idx \leftarrow 1$, $Num_{cluster} \leftarrow 1$
$I_{cluster} \leftarrow \mathcal{R}$
```
/* Initial Clustering                                      */
```
while $Idx \leq Num_{cluster}$ **do**
    $Itsc\_Pts[Idx] \leftarrow$ GET_INTERPTS(Idx)
    if $Itsc\_Pts[Idx]$ *is NULL* **then**
        DIVIDE_CLUSTER($I_{cluster}, Idx$)
        $Num_{cluster} \leftarrow Num_{cluster} + 1$
    else
        $Idx \leftarrow Idx + 1$
$Idx \leftarrow 1$
```
/* Initial path generation                                 */
```
$\Gamma \leftarrow$ GET_COLLECT_PATH($Itsc\_Pts$)
$T_{min} \leftarrow$ CALC_TIME(Γ)

Algorithm 5.2: Collection path generator (continue)

```
   /* Compare and update of collection path                    */
 1 while Idx ≤ Num_cluster do
       /* A cluster consists of only one rover cannot be divided
          */
 2     R_cnt ← COUNT_ROBOTS(Idx)
 3     if R_cnt = 1 then
 4         Idx ← Idx + 1
 5         continue
 6     DIVIDE_CLUSTER(I_cluster, Idx)
 7     Itsc_Pts[Idx] ← GET_INTERPTS(Idx)
 8     Itsc_Pts[Idx + 1] ← GET_INTERPTS(Idx + 1)
 9     Num_cluster ← Num_cluster + 1
       /* Generate a path                                      */
10     Γ_cand ← GET_COLLECT_PATH(Itsc_Pts)
11     T_cand ← CALC_TIME(Γ_cand)
       /* Generate alternative path from the swapped cluster   */
12     I_cluster_temp ← SWAP_CLUSTERS(I_cluster, Idx)
13     Γ_cand_temp ← GET_COLLECT_PATH(Itsc_Pts)
14     T_cand_temp ← CALC_TIME(Gamma_cand)
       /* Compare and choose                                   */
15     if T_cand_temp < T_cand then
16         T_cand ← T_cand_temp
17         Γ_cand ← Γ_cand_temp
18         I_cluster ← I_cluster_temp
19     if T_cand ≤ T_min then
20         T_min ← T_cand
21         Γ ← Γ_cand
22     else
23         MERGE_CLUSTERS(I_cluster, Idx)
24         Num_cluster ← Num_cluster − 1
25         Idx ← Idx + 1
   /* Return result                                            */
26 return Γ, T_min
```



(a) 1st phase (b) 2nd phase (c) 3rd phase

(d) 4th phase (e) 5th phase (f) 6th phase

Figure 5.7: Example procedure of the collection path generation algorithm (a) Initially scattered rovers to be collected (b) Initial clustering and first collection path (c) Second generated collection path (d) and (e) Third and fourth generated collection path. However these results are not chosen (f) Final path for collection is determined if there is no more cluster which can be divided

Based on the geometrical information, the rovers are divided into two clusters so that there is always an intersection area for each cluster, and then initial collection path is computed in Figure 5.7b. In Figure 5.7c, the cluster $\mathbf{P}_{\gamma_1}$ in

Figure 5.7b is divided into $\mathbf{P}_{\gamma_1}$ and $\mathbf{P}_{\gamma_2}$. Consequently, $\mathbf{P}_{\gamma_2}$ in Figure 5.7b is updated to $\mathbf{P}_{\gamma_3}$. Then the algorithm finds collection path for the updated set of clusters and the path is chosen as the time for collection is shorter than the previous computed time. In Figure 5.7d, $\mathbf{P}_{\gamma_1}$ is again divided, however this third generated path is not chosen according to the time. Then two divided clusters are merged again. In Figure 5.7e, $\mathbf{P}_{\gamma_2}$ cannot be divided as there is only on robot. Therefore, the next cluster is divided into $\mathbf{P}_{\gamma_3}$ and $\mathbf{P}_{\gamma_3}$. Assuming that the elapsed time for Figure 5.7e is slower than the previous one, they are merged again. In Figure 5.7f, since there is no more cluster to be divided and computed, final collection path is determined. The result is exactly the same as Figure 5.7c because there has been no update after Figure 5.7c.

5.5 Simulation result

According to the assumption mentioned in section 3.2 , each rover has different remaining energy when rovers are collected after finishing their given tasks. Therefore, in this simulation, the maximum traveling distances of all rovers are randomly set by using mean $d^M_{collect}$ and standard deviation σ. [1]

5.5.1 Collection scenarios in 2D space

Collection of 15 rovers

In chapter 4.5.2, the rovers are deployed for 15 tasks in $100m \times 100m$ space. For those deployed rovers, collection is performed by the proposed method. Initial location of the carrier is set as the last deployment location, and all the conditions are set the same as in Figure 4.10. All initial parameters including

[1] We use $d^M_{collect}$ as the average of remaining energies of rovers.

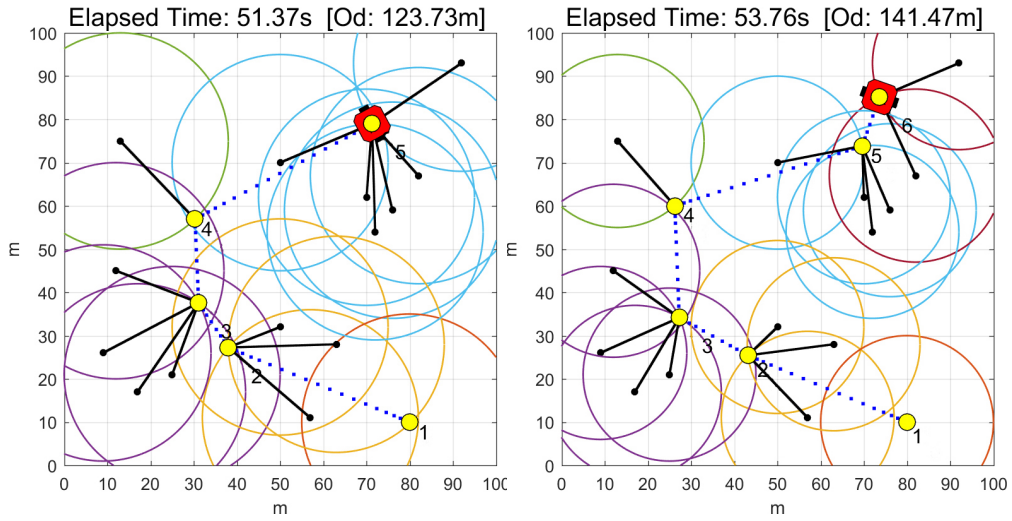
the initial locations are listed in Table 5.1. First, the remaining energies of rovers are assumed to be the same without deviation.

Table 5.1: Initial parameters in collection for 15 rovers

Specification	carrier	$v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$ $t_u = 4.0s$
	rover	$v_r^{max} = 1.0m/s$, $d_{collect}^{M} = 10.0 - 25.0m$ $(\sigma = 0)$
Locations	carrier	$\mathbf{q}_{c_1} = (80, 10)$
	rovers	$\mathbf{q}_r^1 = (57, 11)$, $\mathbf{q}_r^2 = (76, 59)$, $\mathbf{q}_r^3 = (17, 17)$ $\mathbf{q}_r^4 = (13, 75)$, $\mathbf{q}_r^5 = (9, 26)$, $\mathbf{q}_r^6 = (50, 70)$ $\mathbf{q}_r^7 = (82, 67)$, $\mathbf{q}_r^8 = (70, 62)$, $\mathbf{q}_r^9 = (50, 32)$ $\mathbf{q}_r^{10} = (80, 10)$, $\mathbf{q}_r^{11} = (25, 21)$, $\mathbf{q}_r^{12} = (92, 93)$ $\mathbf{q}_r^{13} = (12, 45)$, $\mathbf{q}_r^{14} = (63, 28)$, $\mathbf{q}_r^{15} = (72, 54)$

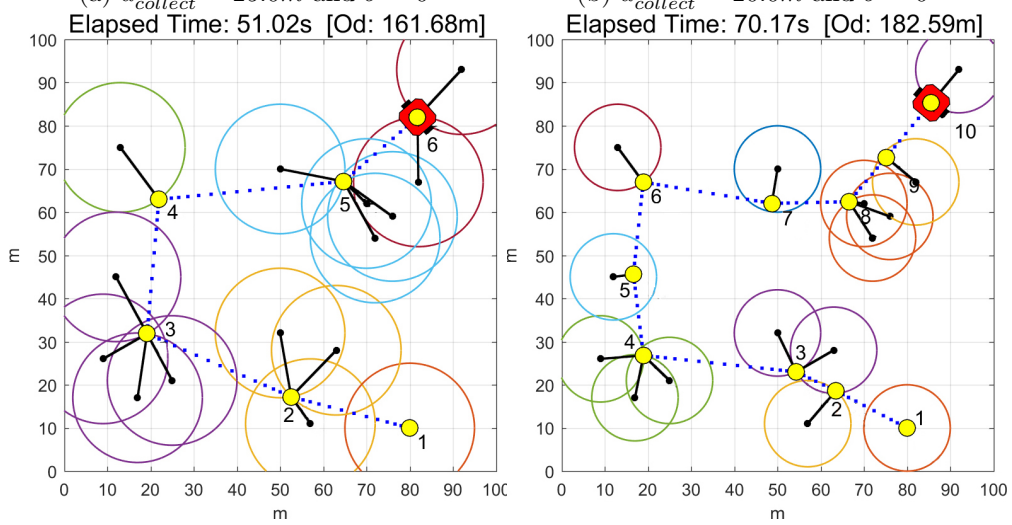
The resulted collecting path for the problem is shown in Figure 5.8. Since the carrier begins collecting from the last deployment location, the first collection location $\mathbf{q}_{\gamma_1}$ and the initial location of the carrier $\mathbf{q}_{c_1}$ are the same for all in Figure 5.8. While the collection locations are not exactly matched with the deployment locations, the overall path seems to be similar as the inversed deployment path in Figure 4.10. Except for the result in Figure 5.8c, the number of generated clusters is also exactly the same as in Figure 4.10. On the other hand, the carrier travels $123.73m$ in Figure 5.8a whereas it travels $157.52m$ in Figure 4.10a. However, the carrier moves $26.97m$ from its initial location to the first deployment location $\mathbf{q}_{\omega_1}$ in Figure 4.10a. Therefore, excluding the distance, the gap between two paths is $6.73m$. The travel distance of the carrier tends to increase as the maximum traveling distance of rover decreases.

Next, for the same set of rovers, the remaining energies are assumed to



(a) $d^M_{collect} = 25.0m$ and $\sigma = 0$

(b) $d^M_{collect} = 20.0m$ and $\sigma = 0$

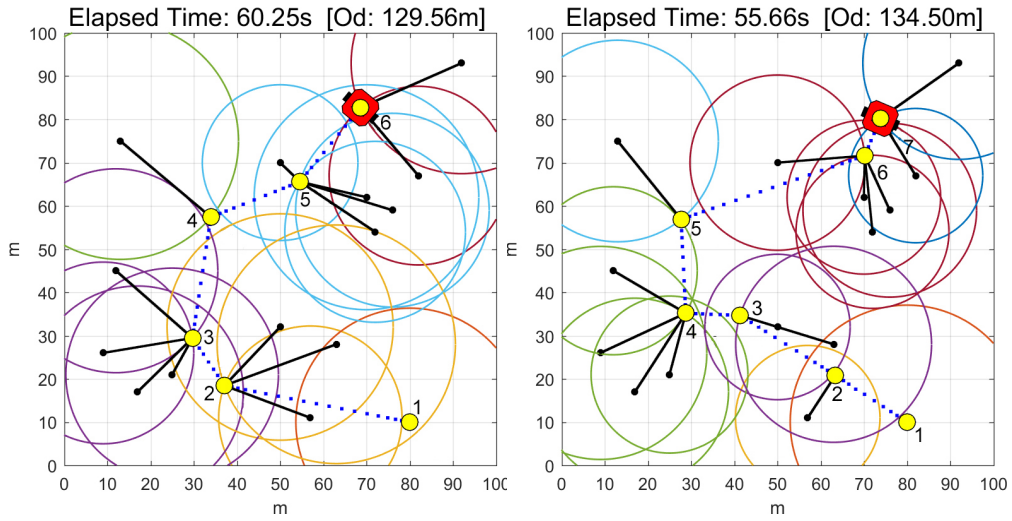


(c) $d^M_{collect} = 15.0m$ and $\sigma = 0$

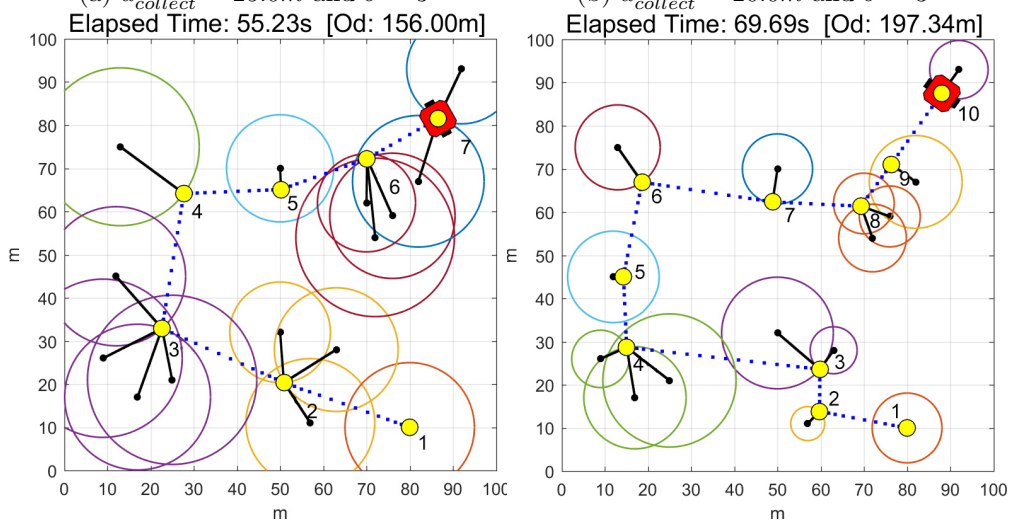
(d) $d^M_{collect} = 10.0m$ and $\sigma = 0$

Figure 5.8: Collection scenario with 15 rovers according to the traveling distance constraint of rovers. All rovers have the same maximum traveling distance. ($v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, $t_l = 4.0s$, and $v_r^{max} = 1.0m/s$.)

be different with standard deviation 3. Figure 5.9 presents the collection paths generation result with this scenario. In Figure 5.9a, one more collection location



(a) $d^M_{collect} = 25.0m$ and $\sigma = 3$ (b) $d^M_{collect} = 20.0m$ and $\sigma = 3$



(c) $d^M_{collect} = 15.0m$ and $\sigma = 3$ (d) $d^M_{collect} = 10.0m$ and $\sigma = 3$

Figure 5.9: Collection scenario with 15 rovers according to the traveling distance constraint of rovers, which are randomly generated with standard deviation 3 ($v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, $t_l = 4.0s$, and $v_r = 1.0m/s$.)

$\mathbf{q}_{\gamma_6}$ is added as not all rovers which belong to $\mathbf{q}_{\gamma_5}$ in Figure 5.8a can gather at one location in Figure 5.9a. Both the elapsed time for collection and travel

distance of the carrier increase to $60.25s$ and $129.56m$ respectively. In Figure 5.9b and 5.9c, one more collection location is also added, however, the travel distance of the carrier is reduced from $147.47m$ to $134.50m$ and from $161.68m$ to $156.00m$ respectively. This collection algorithm has room for improvement. For example, in Figure 5.9d, the collection location $\mathbf{q}_{\gamma_5}$ can be adjusted to the right so that the location lies on the line segment $\overline{\mathbf{q}_{\gamma_4}\mathbf{q}_{\gamma_6}}$. However, at the time when $\mathbf{q}_{\gamma_5}$ is calculated, the center of next cluster is used since the location for $\mathbf{q}_{\gamma_6}$ is not known. Once all locations for collection are computed, then this kind of location can be found and re-calculated. However, this strategy should be carefully selected as it requires additional computation.

Collection of 25 rovers

Table 5.2 shows initial parameters for a scenario of collection path generation.

Table 5.2: Initial parameters in collection for 25 rovers

Specification	carrier	$v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 3.0m/s^2$ $t_l = 2.0s$
	rover	$v_r^{max} = 3.0m/s$, $d_{collect}^{M} = 20.0m$ $(\sigma = 2)$
Locations	carrier	$\mathbf{q}_{c_1} = (20, 20), (180, 20), (20, 180), (100, 100)$
	rovers	$\mathbf{q}_r^1 = (140, 179)$, $\mathbf{q}_r^2 = (28, 30)$, $\mathbf{q}_r^3 = (51, 163)$ $\mathbf{q}_r^4 = (70, 40)$, $\mathbf{q}_r^5 = (95, 71)$, $\mathbf{q}_r^6 = (110, 184)$ $\mathbf{q}_r^7 = (151, 77)$, $\mathbf{q}_r^8 = (11, 107)$, $\mathbf{q}_r^9 = (26, 114)$ $\mathbf{q}_r^{10} = (68, 33)$, $\mathbf{q}_r^{11} = (106, 34)$, $\mathbf{q}_r^{12} = (131, 138)$ $\mathbf{q}_r^{13} = (17, 46)$, $\mathbf{q}_r^{14} = (166, 108)$, $\mathbf{q}_r^{15} = (89, 22)$ $\mathbf{q}_r^{16} = (155, 164)$, $\mathbf{q}_r^{17} = (80, 52)$, $\mathbf{q}_r^{18} = (183, 37)$ $\mathbf{q}_r^{19} = (28, 174)$, $\mathbf{q}_r^{20} = (29, 171)$, $\mathbf{q}_r^{21} = (103, 81)$ $\mathbf{q}_r^{22} = (25, 37)$, $\mathbf{q}_r^{23} = (10, 181)$, $\mathbf{q}_r^{24} = (98, 68)$ $\mathbf{q}_r^{25} = (23, 157)$

There are 25 rovers in $200m \times 200m$ space. All locations of rovers are randomly generated within the space, and the maximum traveling distance of rover is also randomly generated with mean $20.0m$ and standard deviation 2. The result is shown in Figure 5.10.

The paths of the carrier are differently generated as the initial location of the carrier is changed, whereas the other conditions and locations of rovers remain the same. The clustering results are almost the same to each other, however, total elapsed time for collection and travel distance of the carrier vary depending on the initial location of the carrier. As a result, it is shown that the feasible and efficient path can be generated by using the proposed algorithm.

5.5.2 Collection scenarios in 3D space

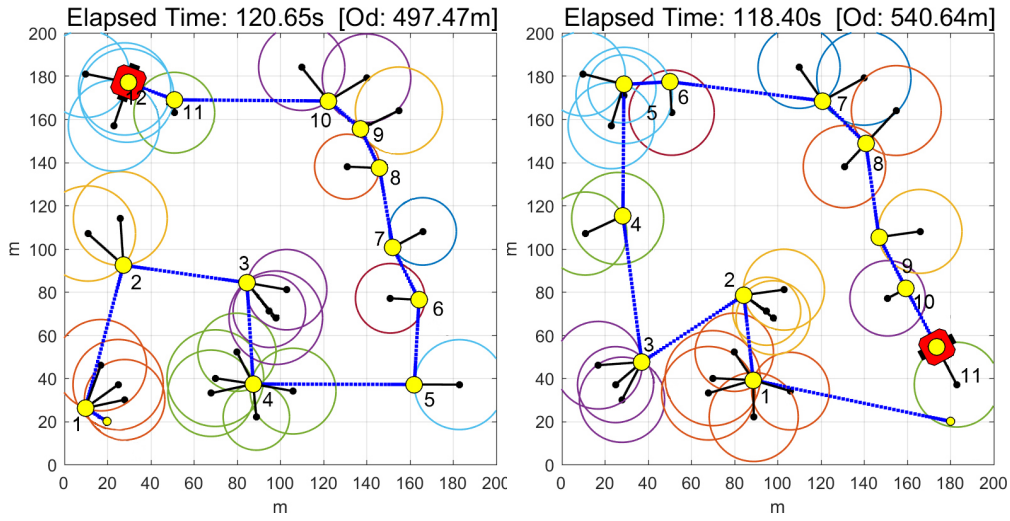
The proposed collection method is also tried in 3D space. We set ten rovers in $100m \times 100m \times 30m$ space. The detailed settings for the simulation are shown in Table 5.3.

Table 5.3: Initial parameters in collection for ten rovers in 3D space

Specification	carrier	$v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$ $t_l = 4.0s$
	rover	$v_r^{max} = 1.0m/s$, $d_{collect}^{M} = 13.0m$ $(\sigma = 2)$
Locations	carrier	$\mathbf{q}_{c_1} = (90, 20, 0)$
	rovers	$\mathbf{q}_r^1 = (76, 59, 8)$, $\mathbf{q}_r^2 = (17, 17, 8)$, $\mathbf{q}_r^3 = (13, 75, 8)$ $\mathbf{q}_r^4 = (50, 70, 8)$, $\mathbf{q}_r^5 = (82, 67, 8)$, $\mathbf{q}_r^6 = (70, 62, 8)$ $\mathbf{q}_r^7 = (50, 32, 8)$, $\mathbf{q}_r^8 = (25, 21, 8)$, $\mathbf{q}_r^9 = (92, 93, 8)$ $\mathbf{q}_r^{10} = (12, 45, 8)$

The collecting result for ten rovers is shown in Figure 5.11. In the figure, the meshed spheres represent the maximum travel distances of the rovers. The

result shows that two rovers are collected at $\mathbf{q}_{\gamma_3}$ and $\mathbf{q}_{\gamma_6}$. If the corresponding clusters are divided so that one is collected after another, total elapsed time



(a) $\mathbf{q}_{c_1} = (20, 20)$ (b) $\mathbf{q}_{c_1} = (180, 20)$

(c) $\mathbf{q}_{c_1} = (20, 180)$ (d) $\mathbf{q}_{c_1} = (100, 100)$

Figure 5.10: Collection scenario with 25 rovers according to initial location of the carrier ($v_c^{max} = 10.0m/s$, $w_c = 2.0rad/s$, $a_c = 3.0m/s^2$, $t_l = 2.0s$, $v_r^{max} = 3.0m/s$, $d_{collect}^M = 20.0m$, and $\sigma = 2.$)

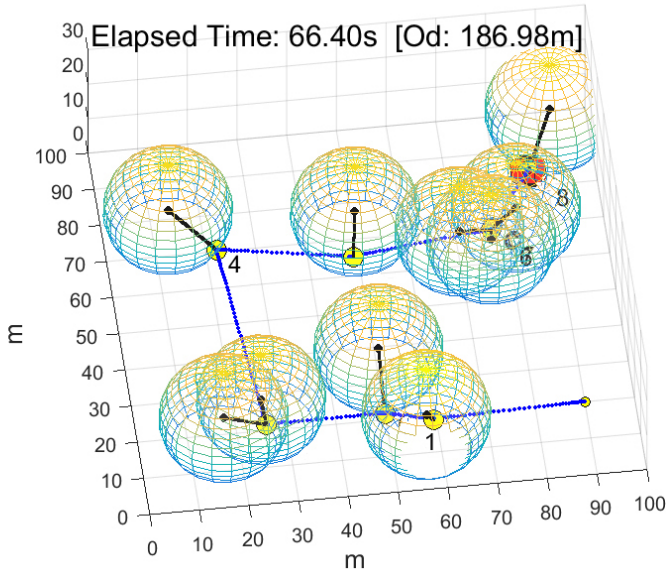


Figure 5.11: Collecting simulation result for ten rovers in 3D space. ($v_c^{max} = 15.0m/s$, $w_c = 3.0rad/s$, $a_c = 10.0m/s^2$, $t_l = 4.0s$, $v_r^{max} = 1.0m/s$, $d_{collect}^M = 13.0m$, and $\sigma = 0$.)

will increase. The number of clusters and the path will vary according to the dynamics of robots. For example, more loading time of rovers may result in less number of clusters.

To evaluate and compare the performance of the proposed algorithm, we execute four methods for the same configuration: 1) the rovers land at their location and stay, and the carrier visits all locations generated by greedy two-opt algorithm; 2) the rovers simply moves so that the distance to the carrier's initial location to be minimized; 3) the rovers move by the proposed algorithm; and 4) the rovers move by the proposed algorithm with convex problem solver. Figure 5.12 demonstrates four paths generation for the same rovers in Figure

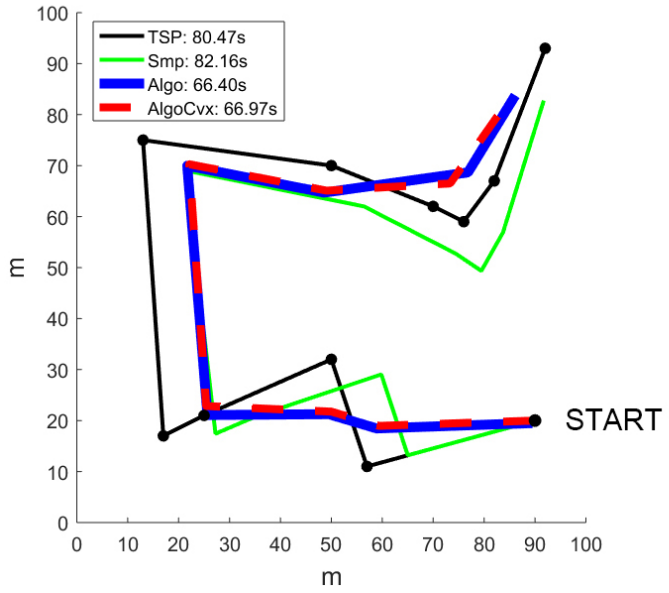
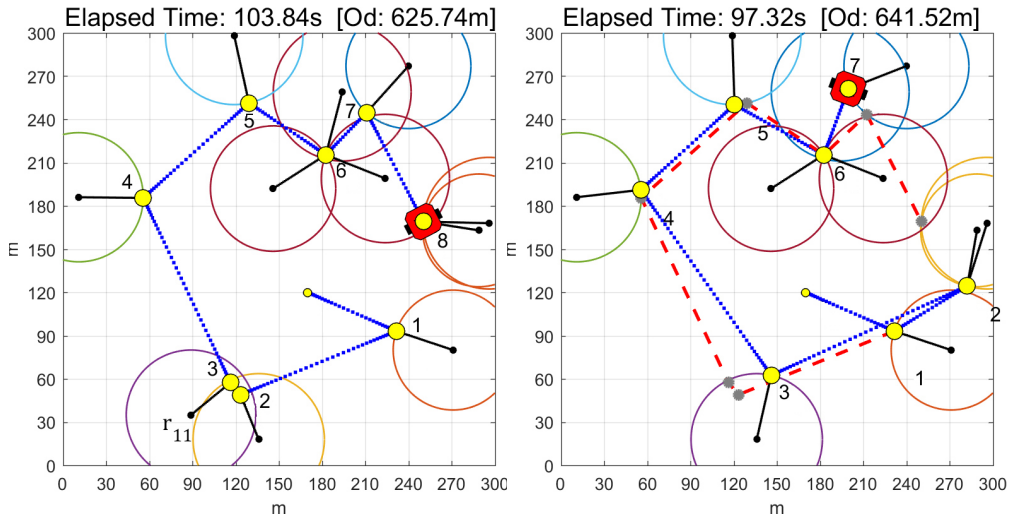


Figure 5.12: Carrier's paths and elapsed times from various approaches

5.11. The black dots represent initial locations of the rovers. This shows that the proposed algorithms generate more time-efficient paths than other approaches. Apparently, big difference between the third algorithm (Algo) and the fourth algorithm (AlgoCvx) is not observed in this result.

5.5.3 Collection in a dynamic environment

As in the case of robot deployment, the dynamic environment must be considered when the robot is collected. Even if a plan is made to collect the rovers in the initial state, communication with the rover to be collected may be disconnected or there may be rovers that do not need to be collected for some reason. Figure 5.13 presents a collection example in $300m \times 300m$ space, when the set of rovers changes during the collection procedure. Figure 5.13a shows



(a) Collection path generation for initial set of rovers

(b) Collection path generation for updated set of rovers at $\mathbf{q}_1$

Figure 5.13: Dynamic collection scenario with 11 rovers in $300m \times 300m$ space ($v_c^{max} = 20.0m/s$, $w_c = 4.0rad/s$, $a_c = 5.0m/s^2$, $t_l = 5.0s$, $v_r^{max} = 6.0m/s$, $d_{collect}^{M} = 45.0m$, and $\sigma = 2$.)

the collection path resulted from initial locations of 11 rovers. In this scenario, one rover RV11 disappears when the carrier is at $\mathbf{q}_{\gamma_1}$. In Figure 5.13b, the red dashed lines present the original path which is the same as the path in Figure 5.13a. If the path is updated at $\mathbf{q}_{\gamma_1}$ assuming that all rovers are at initial locations, the path is generated as blue dot lines. However, in this case, all other rovers are already reached the collection locations and they have no more energy available, when the carrier arrives at $\mathbf{q}_{\gamma_1}$. Therefore, the carrier has to follow the original path for collection excepting for $\mathbf{q}_{\gamma_3}$ in Figure 5.13a. In other case, if some rovers are moving and have energies left when the path needs to update, another path may be chosen for collection. As a result, the collection path can be updated in any case so that the path maintain its efficiency in a

dynamic environment.

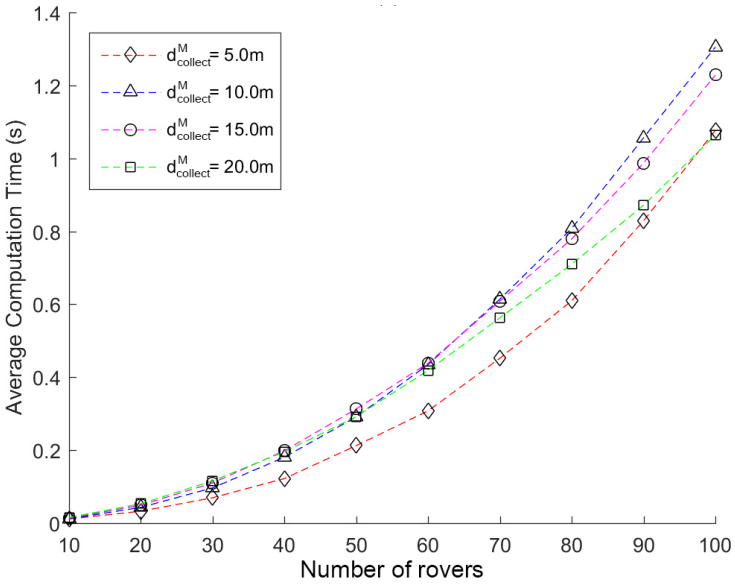
5.6 Performance evaluation

Performance evaluation of robot collection is also described in terms of computation time and path efficiency as in robot deployment.

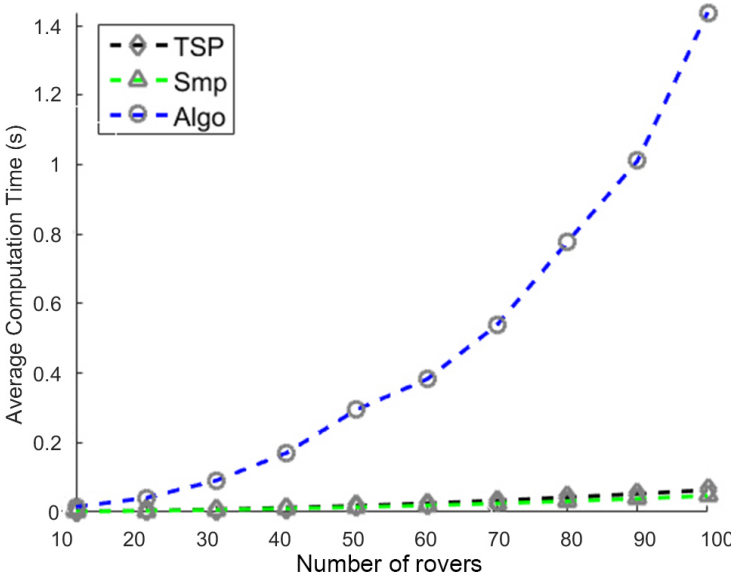
5.6.1 Computation time

To evaluate performance of the proposed algorithm, computation time is measured in $200m \times 200m$ space. All locations of the carrier and rovers are randomly generated with a uniform distribution, and the number of rovers varies from ten to 100. The maximum traveling distances of rover $d^M_{collect}$ vary from $5.0m$ to $20.0m$. Adopting the Monte Carlo method, the parameters of v_c^{max}, w_c, a_c, t_l, and v_r are also randomly generated with a normal distribution, with mean $15.0m/s$, $3.0rad/s$, $5.0m/s^2$, $3.0s$, and $3.0m/s$ respectively. The duration is measured 100 times and the average value is acquired under each distinct condition.

The average computation time of the proposed collection algorithm is depicted in Figure 5.14. The computation time of the collection algorithm increases as the number of rovers increases. If there exists more than 90 rovers, the computation time of the algorithm may exceed 1.0s. The computation of the collection algorithm takes the least time when $d^M_{collect} = 5.0m$ in Figure 5.13a. However any correlation between the maximum traveling distance and the average computation time is not observed. Approximately, the computation of the collection algorithm takes twice as long as that of the deployment



(a) Average computation time according to number of rovers and maximum travel distance



(b) Average computation time according to number of rovers and the approach

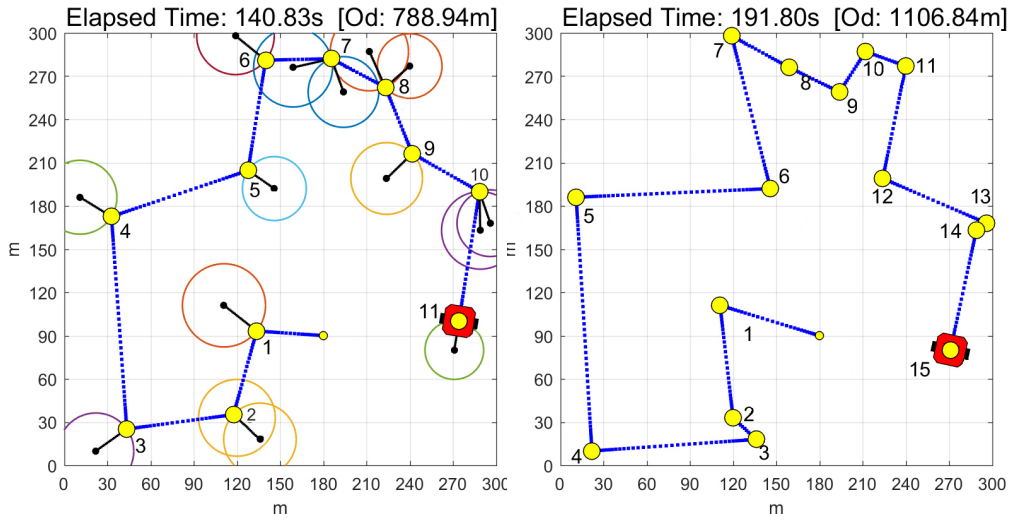
Figure 5.14: Average computation time of proposed collecting algorithms. All parameters and locations are randomly generated adopting the Monte Carlo method

algorithm which is measured in section 5.6. Since the clustering methods of the algorithms are basically the same, the difference may come from the calculations for each cluster. On the other hand, Figure 5.13b shows the result of computation times for three methods introduced in section 5.5.2. Although the proposed method requires more computation than the other simple methods, it is confirmed that when a limited number of robots are operated, the results are still close to real time. However, using the convex problem to find a more accurate collecting point take much more time and did not appear in the graph. When using the convex problem for the same configuration, the average computation times required for the rovers from ten to 100 are 2.60s, 10.16s, 19.81s, 33.14s, 49.14s, 72.49s, 101.91s, 128.75s, 185.63s, and 193.60s respectively. Reducing the computation time also can be considered as one of future works.

5.6.2 Efficiency of the path

As introduced in chapter 4, the path from the proposed algorithm is compared with the greedy two-opt solution. Figure 5.15 demonstrates an example of two different collection solutions for the same problem of 15 rovers in $300m \times 300m$ space. As the rovers in Figure 5.15b cannot move to meet the carrier, the carrier's travel distance becomes larger than the distance in Figure 5.15a. Therefore, the elapsed time also increases. In this case, the path from the proposed algorithm is 26.58% faster than the other.

As the efficiency of the path is mainly affected by the traveling distance of rovers, and by the ratio of carrier's speed to rovers' speed, $d^M_{collect}$ varies from $10.0m$ to $40.0m$. The number of tasks and rovers is set from ten to 100, and



(a) The proposed (b) Greedy two-opt

Figure 5.15: Example of 15 rovers collection in $300m \times 300m$ ($v_c^{max} = 20.0m/s$, $w_c = 4.0rad/s$, $a_c = 5.0m/s^2$, $t_l = 5.0s$, $v_r^{max} = 6.0m/s$, $d_{collect}^M = 25.0m$, and $\sigma = 2$.)

all locations of them are randomly generated. For the other parameters, the Monte Carlo method is also used with a normal distribution, and each distinct condition is repeated 100 times. The relative efficiency of the proposed collection algorithm is presented in Figure 5.16, comparing with greedy two-opt solution. The efficiency tends to increase as the number of rovers increases. The total average of the efficiency is 26.75%. This efficiency comes from in part by the adjacency of rovers as they can be put into a cluster. Therefore the smaller space may result in the more efficient path, while the bigger space may result in the similar path as the path from greedy two-opt algorithm.

Finally, the ratio of mission time according to number of rovers is compared as in Figure 5.17. This figure shows the relative time of the other methods when the time of the method using TSP is taken as 100%. In the figure, the proposed

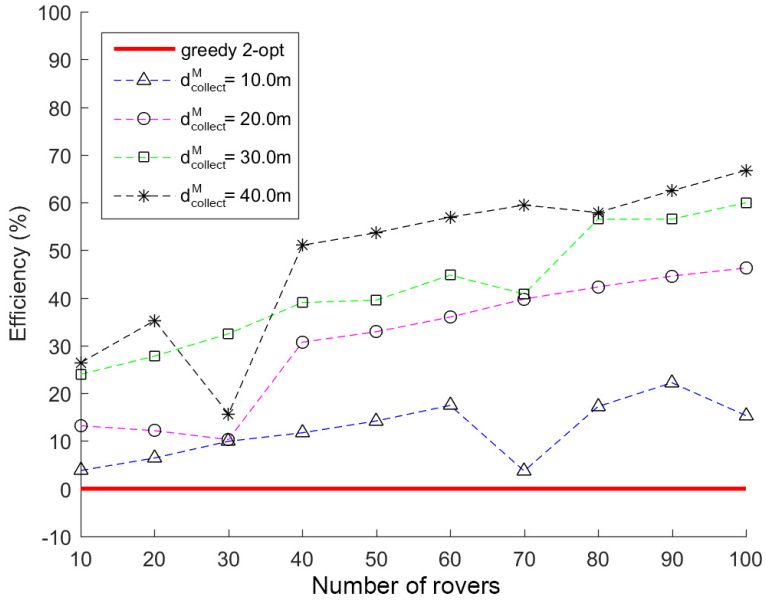


Figure 5.16: Efficiency of the collecting path comparing with the solution from greedy two-opt algorithm, in $300m \times 300m$ space

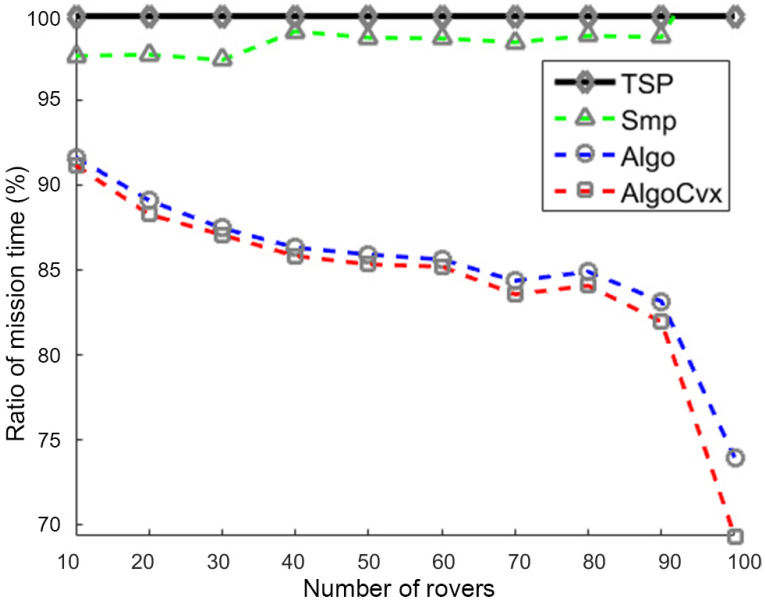


Figure 5.17: Ratio of mission time according to number of rovers and approaches

algorithm improves the duration approximately from 10 to 20 percent. However, the algorithm using convex solver makes no big difference while it requires a lot of computation time.

Chapter 6

Deployment of a Marsupial Robot Team using a Graph

6.1 Problem definition

The goal of the problem is to find a *route* of the carrier, which is a sequence of nodes, that minimizes the longest time at which each rover arrives at the given task location. (Once a route for the carrier is determined, the routes of rovers are also determined from the deployment locations.) In a graph $\mathcal{G}$, let the route of the carrier be $\mathcal{R}_c = (n_1, n_2, ..., n_j)$ where $n \in N$, and let $\mathcal{R}_c^i$ be the sub-route of $\mathcal{R}_c$ which means a path from one deployment location to next deployment location, such that $\bigcup \mathcal{R}_c^i = \mathcal{R}_c$. Then a function $s : \mathcal{R}_c^i \to \mathbb{R}_{\geq 0}$ that calculates the total length of the sub-route $\mathcal{R}_c^i$ can be formulated as the sum of Euclidean distances between two sequential nodes in the sub-route, as follows:

$$s(i) = \sum_{k=1}^{\beta(i)-1} \|e_{(\alpha(i,k),\alpha(i,k+1))}\|, \tag{6.1}$$

where $\alpha(i,k)$ is a mapping function to kth node in $\mathcal{R}_c^i$, $\beta(i)$ is a function that returns the number of nodes in $\mathcal{R}_c^i$, and $e \in E$ is the edge between two nodes. By using the function s, a function f_c that measures travel time for a carrier is defined as follows:

$$f_c(i) = \begin{cases} s(i)/v_c^{max} + v_c^{max}/a_c, & \text{if } s(i) \geq v_c^{max2}/a_c, \\ 2\sqrt{s(i)/a_c}, & \text{otherwise.} \end{cases} \tag{6.2}$$

Similarly, a function f_r for a rover can be defined by replacing the velocity and acceleration terms in (6.2) with those terms of rovers. In the first sub-route $\mathcal{R}_c^1$, the carrier travels from its initial location to the first deployment location. Then rovers are unloaded at the deployment location, and the rovers move to the assigned goal locations. Expanding on this concept, a function g that computes total time at which unloaded rovers arrive at each location is derived as follows:

$$\begin{aligned} g(1) &= f_c(1) + t_u + max(f_r^1), \\ g(2) &= f_c(1) + t_u + f_c(2) + t_u + max(f_r^2), \\ &\vdots \\ g(i) &= \textstyle\sum_{k=1}^{i} (f_c(k) + t_u) + max(f_r^i). \end{aligned} \tag{6.3}$$

Therefore, the objective is to finding an efficient route $\mathcal{R}_c^\star$ that minimizes the longest time of rovers as follows:

$$\mathcal{R}_c^\star = \arg\min_{\mathcal{R}_c} \max\{g(1), ..., g(i), ...\}. \tag{6.4}$$

To save time, at the former deployment locations, it can be expected that the carrier will unload rovers near the maximum range that the rovers can travel.

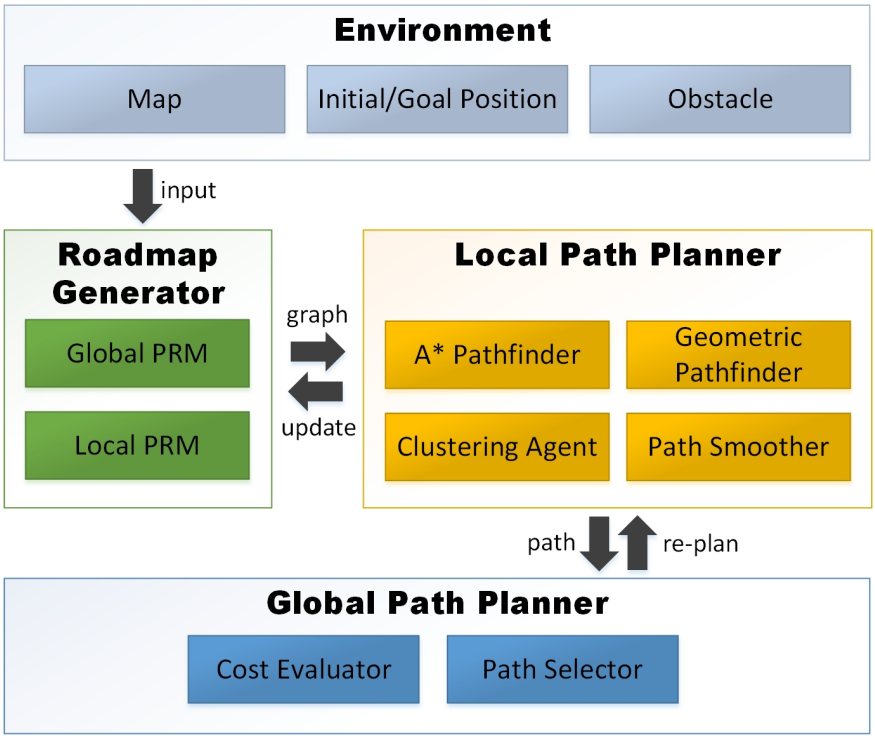


Figure 6.1: Framework for path planning of a marsupial robot team

6.2 Framework

The path planning of the marsupial robot team using the graph can be divided into three steps: 1) roadmap generation; 2) path finding in the graph; and 3) path smoothing. Taking the steps into account, our overall framework to find an efficient path is diagrammed as Figure 6.1. First, preliminary information on environment such as map, obstacles, initial locations, and goal locations is input to the roadmap generator. Then the roadmap generator creates a roadmap for the entire environment by using the PRM which is the multi-query sampling-based method. The local PRM generates additional nodes and edges in a specified

region of interest in open space. On the generated roadmap, the shortest path from one node to another node is found by using the A* pathfinder. By using the A* pathfinder, not only the nearest task location from the carrier but also the path from a deployment location to a task location can be found. On the other hand, the clustering agent finds near task locations based on a specific node which is also a task location, and the found tasks and the first task become a cluster. Then a deployment location associate with the cluster can be determined. In this process, the geometric pathfinder is used to compensate for the drawbacks of sampling based roadmap. For efficiency, the generated path along the graph nodes should be smoothed. This is done by the path smoother. Note that, path smoothing results in short-cutting edges in the graph, while creating a curved path is beyond the scope of this paper. Once a path is created, the time required for travel can be calculated in the cost evaluator by using the robot's attributes. The path selector in the global planner then chooses the path that minimizes the time. The details of the approach in this framework are explained in the next section.

6.3 Probabilistic roadmap generation

PRM is a multiple-query algorithm which has demonstrated to be efficient and general tools for motion planning. In a basic PRM, the number of nodes in the map and the maximum length of edges can be determined. As the number of nodes or edge length increases, the efficiency of the path increases. However, the search time also increases. On the contrary, if the number decreases, the search time can be accelerated, whereas the necessary path may not be generated. As

a result, it is important to determine the number and locations of nodes. In this study, we use the method with uniform distribution as the most basic PRM.

6.3.1 Global PRM

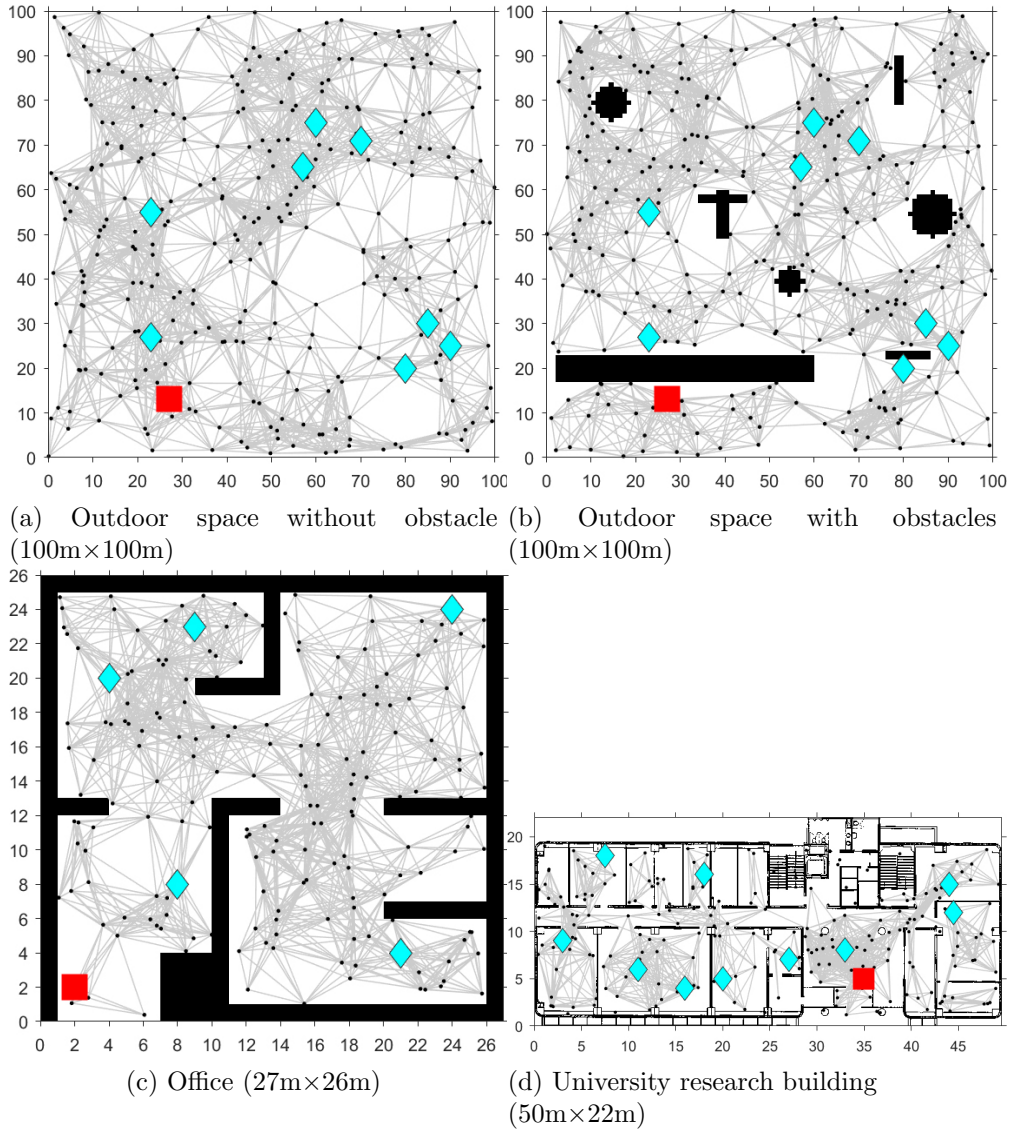
The initial number of nodes $\aleph_1$ for a map with height h and width w is determined by the following heuristic function:

$$\aleph_1 = (\sqrt{w \times h}) \times \xi_1 + \xi_2, \tag{6.5}$$

where ξ_1 and ξ_2 are non-negative constants respectively. If the coordinates of a generated node belong to obstacle area, the node is canceled and not included in the total node number $\aleph_1$. Therefore, it is generated until the number $\aleph_1$ is satisfied regardless of the obstacle of environment. Generally, the maximum length of edge is preferably determined by considering the size of the map. However, if the length is shorter than d_{deploy}, the rover may be unable to directly go to the task location. Hence, the maximum length of edge δ is calculated by the following function:

$$\delta = \max\left(d_{deploy}, \frac{(w+h)}{2}\phi\right), \tag{6.6}$$

where ϕ is a constant satisfies $0 < \phi < 1$. The roadmaps generated by global PRM for four different areas are demonstrated in Figure 6.2. In the figure, small black dots are nodes, grey lines are edges, and black blobs are obstacles or walls. The red rectangle and skyblue diamonds respectively represent the carrier and the tasks. Note that, the locations of the carrier and the tasks are all the same in Figure 6.2a and 6.2b. The parameters ξ_1 and ξ_2 in (6.5) and number of generated nodes and edges for each area is shown in Table 6.1. For



(a) Outdoor space without obstacle (100m×100m)

(b) Outdoor space with obstacles (100m×100m)

(c) Office (27m×26m)

(d) University research building (50m×22m)

Figure 6.2: Roadmap generation by global PRM. The roadmap is generated including the locations of carrier and tasks as nodes. In the map, small black dots are nodes, grey lines are edges, and black blobs are obstacles or walls. The red rectangle and skyblue diamonds respectively represent the carrier and the tasks.

Table 6.1: Number of nodes and edges for four areas

Area	ξ_1	ξ_2	Nodes	Edges
Outdoor w/o obstacle	1.5	150	300	2,761
Outdoor w/ obstacles			300	2,594
Office			190	1,659
Univ. building			220	1,248

the result, we set $d_{deploy} = 15.0m$ and $\phi = 0.15$. Comparing the space with obstacles to the space without an obstacle, it is confirmed that the obstacle causes less edge generation.

6.3.2 Local PRM

Local PRM is used to provide more time-efficient nodes. On the roadmap generated by global PRM, the local PRM creates additional nodes and links edges from the newly added nodes. The local PRM requires region of interest obtained from cluster information, whose width and height are respectively represented as w_{ROI} and h_{ROI}. Then, similarly as (6.5), the number of nodes $\aleph_{add}$ that the local PRM creates is calculated as the following equation:

$$\aleph_{add} = (\sqrt{w_{ROI} \times h_{ROI}}) \times \xi_3 + \xi_4, \tag{6.7}$$

where ξ_3 and ξ_4 are non-negative constants respectively. As a result of addition, the total number of nodes becomes $\aleph_1 + \aleph_{add}$. Meanwhile, the maximum length of edge δ is set as $\delta = d_{deploy}$. The example of local PRM is shown with clustering in the section 6.4.2.

6.4 Path planning strategy

This section describes strategies to path planning for a marsupial robot team.

6.4.1 Clustering scheme

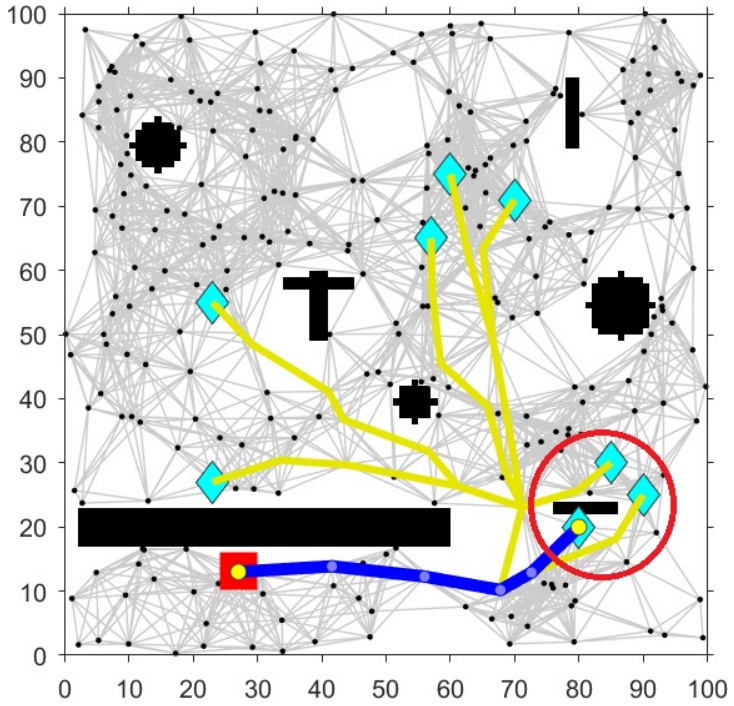
Assume that there are more than two tasks in the local area. Once the optimal location to reach each task location is found, the carrier should repeatedly go and stop at every deployment location and unload a rover. However, in this case, if there is a location where all the task locations can be reached by rovers, the carrier can stop only once and unload all the rovers, which may be more advantageous in terms of time. Therefore, clustering tasks based on location should be considered.

Cluster creation

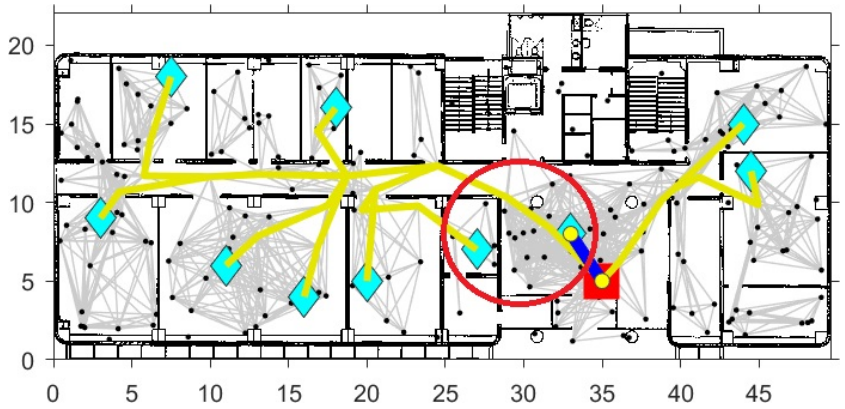
Creation of a cluster begins with a greedy search from the initial position of the carrier n_1. Let $\mathcal{C}_1$ denote an initially-empty set for a cluster, and let $A^*(a, b)$ be a function that returns the length of shortest path from n_a to n_b. Then the nearest task node $n_1^{nearest}$ from n_1 is obtained as follows:

$$n_1^{nearest} = \arg\min_{n_g^i}(A^*(n_1, n_g^i)). \tag{6.8}$$

This acquired node is added as an element of $\mathcal{C}_1$. Next, for the remaining task nodes, the Euclidean distance from $n_1^{nearest}$ is calculated. If the calculated distance is less than $2 \cdot d_{deploy}$, the corresponding node is also added to $\mathcal{C}_1$. The results of the above procedure for the maps in Figure 6.2b and 6.2d are respectively shown in Figure 6.3a and 6.3b. In Figure 6.3, the path from the carrier to task is indicated by yellow lines, and the shortest path is drawn with blue lines. From the nearest task node, the cluster is made as the red circle in Figure 6.3. If a deployment location for the cluster can be found, the location is to be



(a) Outdoor space with obstacles



(b) University research building

Figure 6.3: Path from carrier to task locations and first clustering result (red circle). Each path and the path to the nearest task are respectively drawn with yellow lines and blue lines.

the next location of the carrier. Then next cluster $\mathcal{C}_2, \mathcal{C}_3, ...$ can be made based on the previous deployment location.

Cluster division

Cluster creation is based on geographic information between tasks. Therefore, there is no problem if there is no obstacle. However, if obstacle exists in cluster, some rovers may not reach the assigned task location from the deployment location. Figure 6.3b shows an example. The nearest task is found from the carrier and a cluster is created containing other task. However, there is a wall between the two tasks. Therefore, if the maximum travel distance of the rover is shortened, finding a deployment location where both rovers can actually arrive is impossible. In this case, the cluster must be divided so that more than two deployment locations to be generated. The process of dividing the cluster is described in Algorithm 6.1. The task nodes excluded from the cluster by Algorithm 6.1 are clustered together with other task nodes at the next cluster creation.

Algorithm 6.1: ClusterDivider

Input: Cluster $\mathcal{C}$, Nearest task node $n^{nearest}$
Output: Cluster $\mathcal{C}$

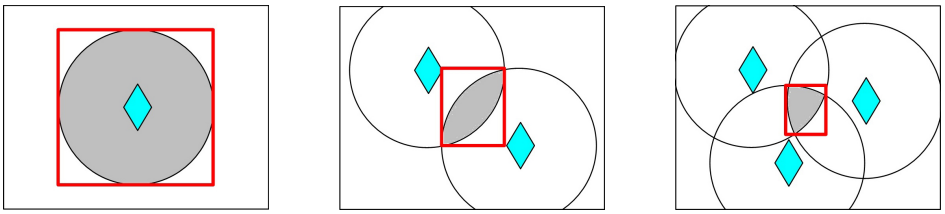
```
1 n^farthest ← FindFarthestNode(C, n^nearest)
2 C ← C \ n^farthest
3 for i=1:Num(C) do
4     if n_i ≠ n^farthest and n_i ≠ n^nearest then
5         path_near ← A*(n_i, n^nearest)
6         path_far ← A*(n_i, n^farthest)
7         if GetLength(path_far) ≤ GetLength(path_near) then
8             C ← C \ n_i
```

6.4.2 Determining deployment locations

Candidate nodes creation with the local PRM

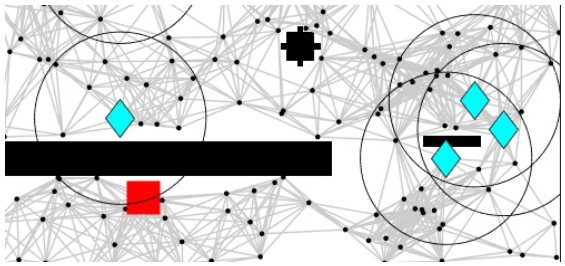
Once the next cluster to be visited is determined from the carrier's location, an efficient deployment node corresponding to that cluster must be found. Since each rover has a limited maximum travel distance, this node should be located at least within d_{deploy} from each task node. Then, the desired deployment node can only be in an intersection area of circles whose radius is d_{deploy} and center is each task node, as shown in Figure 6.4. However, the graph generated by the PRM may or may not have enough nodes to have optimal node in this area. Therefore, we use the local PRM to generate more possible nodes in the area. To simplify the computation, the intersection area of circles is approximated as a red rectangle in Figure 6.4.

Figure 6.5 shows the result of local PRM for the first cluster in outdoor space with obstacle which is depicted in Figure 6.2b. First, in Figure 6.5a, there are three task locations in a cluster, each with its maximum range indicated by a circle. In the intersection area, there are about seven nodes. Next, in Figure 6.5b, the local PRM is used to create random nodes in the intersection area. The generated nodes are indicated by green circles whereas the newly connected edges are not indicated. Although all these nodes are in the intersection area, on some nodes a rover deployed may not reach task location due to obstacles. Therefore, each candidate node is examined with the path-length to task location in the cluster by using A* algorithm, and re-selects only the feasible nodes as candidates. The selected nodes are red circles in Figure 6.5c.

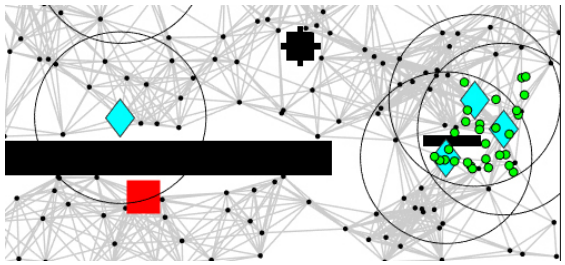


(a) One task (b) Two tasks (c) Three tasks

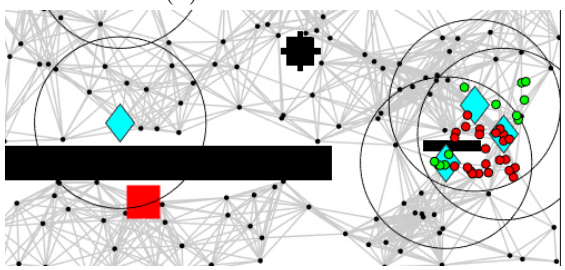
Figure 6.4: Deployable area (grey area) according to task configuration (skyblue diamond) in cluster. The area is approximated as a red rectangle.



(a) Before local PRM



(b) After local PRM



(c) Reachable nodes

Figure 6.5: Local PRM generation in the first cluster for outdoor space with obstacle. Maximum travel distance of rover is 15.0m

Candidate nodes creation with a deterministic method

Since the local PRM is a sampling-based approach, it is difficult to provide an optimal solution to this partial problem. The optimal solution may be acquired by generating a lot of samples, however it may also require a lot of computation which is almost infeasible in robotic real operation. On the other hand, if there is no obstacle around the interested cluster, then a candidate node can be additionally made in a deterministic way. This method is previously introduced in chapter 4, and as a result an additional candidate node is set at one point on the boundary of the intersection area.

Deployment node selection

The criterion for choosing a node to deploy rovers are to select the node that makes the fastest arrival at the next deployment location. However, there is no information about the next cluster at the time of determining the deployment location for the current cluster. Therefore, the next deployment location is temporarily set to the nearest task node from the center position of the tasks which belong to the current cluster. The center position is calculated by finding the minimum bounded circle as follows:

$$\begin{aligned} &\text{minimize} \quad r \\ &\text{subject to} \quad \|n_i - \lambda\| \leq r, \end{aligned} \tag{6.9}$$

where n_i is the node in the current cluster. If there is only one node in the cluster, the center position becomes the node itself, and if there are two nodes, the center position becomes the mid-point of two nodes. In case there are more than three nodes, convex hull [38] is characterized first so that only outer nodes

can be considered. Next, the center of bounded circle is computed by finding three points which satisfy the same equation of circle. By using the center location, the next nearest task node is found.

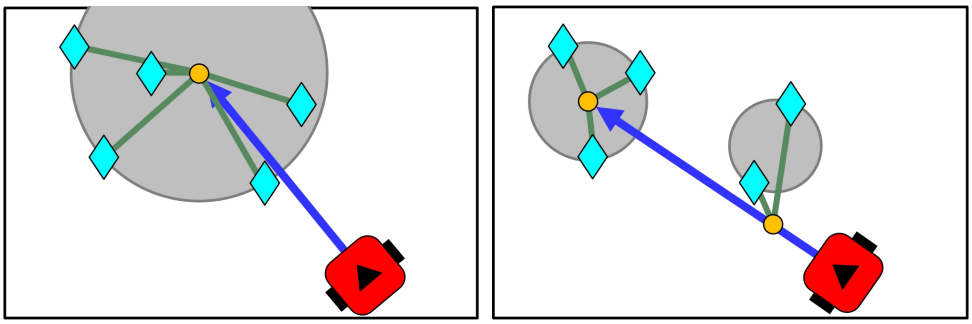
Let the initial location or the previous deploying node of the carrier be n_{i-1} and the next nearest task node be $n_{i+1}^{nearest}$. Then, the best node n_i to deploy rovers for the corresponding cluster can be determined by solving the following problem:

$$n_i = \arg\min_{n_{cand}}(A^*(n_{i-1}, n_{cand}) + A^*(n_{cand}, n_{i+1}^{nearest})), \tag{6.10}$$

where n_{cand} is the candidate deployment node. Then the path from n_{i-1} to n_i becomes a sub-route.

Recursive method for last cluster

If one cluster is created for tasks not belonging to the previous cluster, if there are no remaining tasks, the cluster is found to be the last cluster. In the last cluster, the deployment location should be the node that minimizes the maximum duration each rover in the cluster arrives at the task location. However, if there are more than two tasks in the last cluster, dividing the cluster into two should be considered as shown in Figure 6.6. First, a deployment location is determined as shown in Figure 6.6a. Next, the cluster is divided into two clusters based on the geographical adjacency of the tasks, and two deployment locations are determined as shown in Figure 6.6b. In Figure 6.6b, the former deployment location should be chosen so that the time the rovers travel in the former cluster is nearly equal to the time the carrier travel to the latter cluster plus the time for unloading plus the time the rovers travel in the latter cluster.



(a) One deployment location for last cluster

(b) Two deployment locations for divided clusters

Figure 6.6: Two case of deployment for last cluster. Last cluster can be divided into two clusters unless there is only one task. Yellow circles indicate deployment locations. Blue lines and green lines respectively indicate the path of the carrier and rovers.

As the time required for arriving all tasks can be calculated by (6.3), the better deployment location can be chosen. It is expected that slower linear velocity of the rover and faster unloading time may result in the division of the cluster.

6.4.3 Path smoothing

Sampling-based path planning tends to find jagged and longer path. In the case of PRM, this problem is also caused by limiting the edge distance as (6.6). However, finding a shortcut path from the deploying path result does not require a lot of computation. Therefore, for each sub-route, shortcutting algorithm is performed as a smoothing function. The shortcutting procedure is described in Algorithm 6.2. Since this algorithm is based on a greedy strategy, it does not guarantee optimal shortcuts. The example of smoothing in office area is depicted in Figure 6.7. Grey lines show the original path before path smoothing, blue lines show the smoothed path, and green lines show the rover's

Algorithm 6.2: Shortcutting

Input: Sub-route $\mathcal{R}_c^i$
Output: Smoothed sub-route ${}_s\mathcal{R}_c^i$
`/* Initialization */`
${}_s\mathcal{R}_c^i \leftarrow \emptyset$, $n \leftarrow \text{GetNodeNum}(\mathcal{R}_c^i)$
`/* Return sub-route if there are only two nodes */`
if $n = 2$ **then**
  ${}_s\mathcal{R}_c^i \leftarrow \mathcal{R}_c^i$
  return
$j \leftarrow 1$, ${}_s\mathcal{R}_c^i \leftarrow \text{AddNode}({}_s\mathcal{R}_c^i, j)$
while $j < n$ **do**
  for $k = n : -1 : j + 1$ **do**
    if *NoCollision*($\mathcal{R}_c^i$, j,k) **then**
      ${}_s\mathcal{R}_c^i \leftarrow \text{AddNode}({}_s\mathcal{R}_c^i, k)$
      $j \leftarrow k - 1$
      break
  $j \leftarrow j + 1$

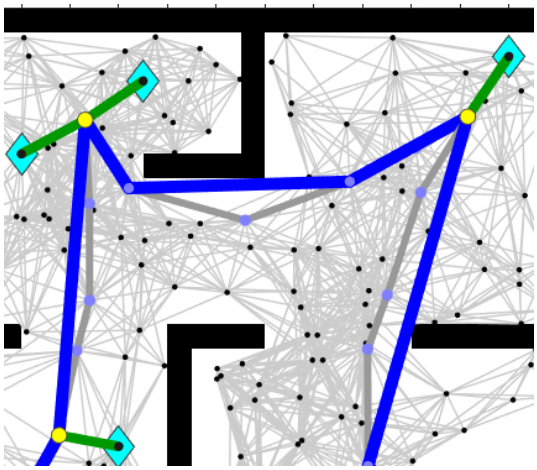


Figure 6.7: Shortcutting of path in office area. Thick grey lines show the original path and blue lines show the smoothed path of the carrier.

path from deployment locations. As shown in the figure, it can be seen that the somewhat zigzag grey path changes to a more efficient blue path without

colliding with obstacles.

6.4.4 Path planning algorithm for a marsupial robot team

Summarizing the methods above, the overall algorithm for path planning is briefly described in Algorithm 6.3. The route of rover is found from each deployment location to task location in GetRoverRoute function.

Algorithm 6.3: Path planning for a marsupial robot team

Input: map $\mathcal{M}$, robot's information $\mathcal{I}$, task locations $\mathbf{q}_{r_G}$
Output: Path of carrier and rovers
`/* Initialization */`
$\mathcal{G} \leftarrow$ GlobalPRM($\mathcal{M}$, $\mathcal{I}$, $\mathbf{q}_{r_G}$), $\mathcal{R}_c = \emptyset$, $i \leftarrow 1$
while do
  $\mathcal{I} \leftarrow$ UpdateCarrierLocation($\mathcal{R}_c$)
  $\mathcal{C}_i \leftarrow$ GetCluster($\mathcal{G}$, $\mathcal{I}$, $\mathbf{q}_{r_G}$)
  if *IsLastCluster() = false* **then**
    $n_{i+1}^{nearest} \leftarrow$ GetNextNearestTask($\mathcal{G}$, $\mathcal{C}_i$, $\mathbf{q}_{r_G}$)
    $n_i \leftarrow$ FindDeployNode($\mathcal{G}$, $\mathcal{C}_i$, $\mathcal{I}$, $n_{i+1}^{nearest}$)
    $\mathcal{R}_c \leftarrow$ GetSmoothedSubRoute($\mathcal{G}$, $\mathcal{I}$, n_i)
    $i \leftarrow i + 1$
  else
    $n_i^{cand} \leftarrow$ FindDeploymentNode($\mathcal{G}$, $\mathcal{C}_i$, $\mathcal{I}$)
    ($\mathcal{C}_i$, $\mathcal{C}_{i+1}$) $\leftarrow$ Divide($\mathcal{G}$, $\mathcal{C}_i$, $\mathbf{q}_{r_G}$)
    $n_{i+1}^{nearest} \leftarrow$ GetNextNearestTask($\mathcal{G}$, $\mathcal{C}_i$, $\mathbf{q}_{r_G}$)
    $n_i^{cand2} \leftarrow$ FindDeployNode($\mathcal{G}$, $\mathcal{C}_i$, $\mathcal{I}$, $n_{i+1}^{nearest}$)
    $n_{i+1}^{cand2} \leftarrow$ FindDeployNode($\mathcal{G}$, $\mathcal{C}_{i+1}$, $\mathcal{I}$)
    if *GetTime*(n_i^{cand}) < *GetTime*(n_i^{cand2}, n_{i+1}^{cand2}) **then**
      $\mathcal{R}_c \leftarrow$ GetSmoothedSubRoute($\mathcal{G}$, $\mathcal{I}$, n_i^{cand})
      break
    else
      $\mathcal{R}_c \leftarrow$ GetSmoothedSubRoute($\mathcal{G}$, $\mathcal{I}$, n_i^{cand2})
      $i \leftarrow i + 1$
GetRoverRoute($\mathcal{R}_c$, $\mathbf{q}_{r_G}$)

6.5 Simulation result

Our proposed method was implemented and simulated for the environments shown in Figure 6.2, outdoor space without obstacle, outdoor space with obstacles, office area, and university research building. For clear presentations, nodes and edges that do not belong to path are not drawn in the result figures.

6.5.1 Outdoor space without obstacle

First, the proposed algorithm is simulated for the outdoor space without obstacle shown in Figure 6.2a. All the parameters used for this simulation is listed in Table 6.2.

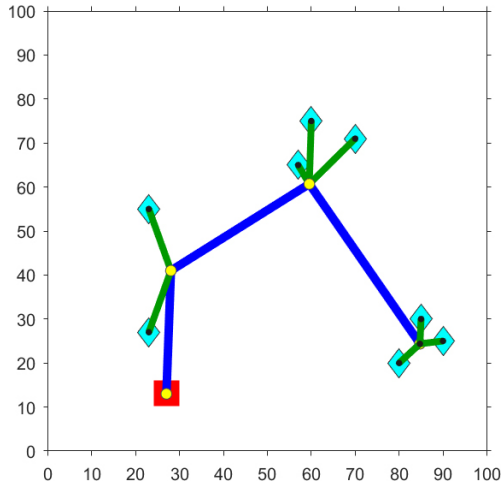
Table 6.2: Initial parameters in outdoor space without obstacle for eight tasks

Specification	carrier	$v_c^{max} = 10.0m/s$, $a_c = 5.0m/s^2$, $t_u = 3.0s$
	rover	$v_r^{max} = 5.0m/s$, $a_r = 5.0m/s^2$, $d_{deploy} = 15.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (27, 13)$
	tasks	$\mathbf{q}_G^1 = (23, 27)$, $\mathbf{q}_G^2 = (80, 20)$, $\mathbf{q}_G^3 = (90, 25)$ $\mathbf{q}_G^4 = (85, 30)$, $\mathbf{q}_G^5 = (70, 71)$, $\mathbf{q}_G^6 = (60, 75)$ $\mathbf{q}_G^7 = (57, 65)$, $\mathbf{q}_G^8 = (23, 55)$

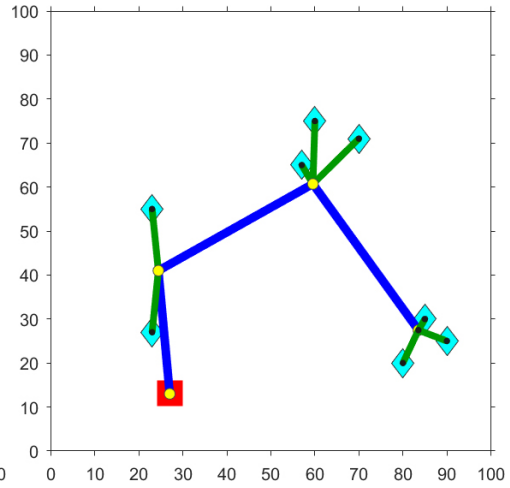
Figure 6.8 shows the path planning results for rovers deployment, and the related values are listed in Table 6.3. First, the path by using the proposed

Table 6.3: Number of nodes and edges, and elapsed time in outdoor space without obstacle

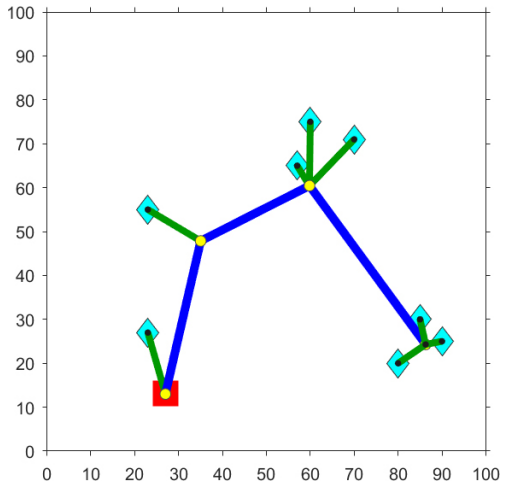
Method	Nodes	Edges	Time(s)
The proposed	393	6,238	27.14
The proposed without local PRM	300	2,761	27.58
Normal method with more nodes	393	4,761	30.35



(a) Proposed path planning



(b) Proposed path planning without local PRM



(c) Paths using the same number of nodes as the result of (a)

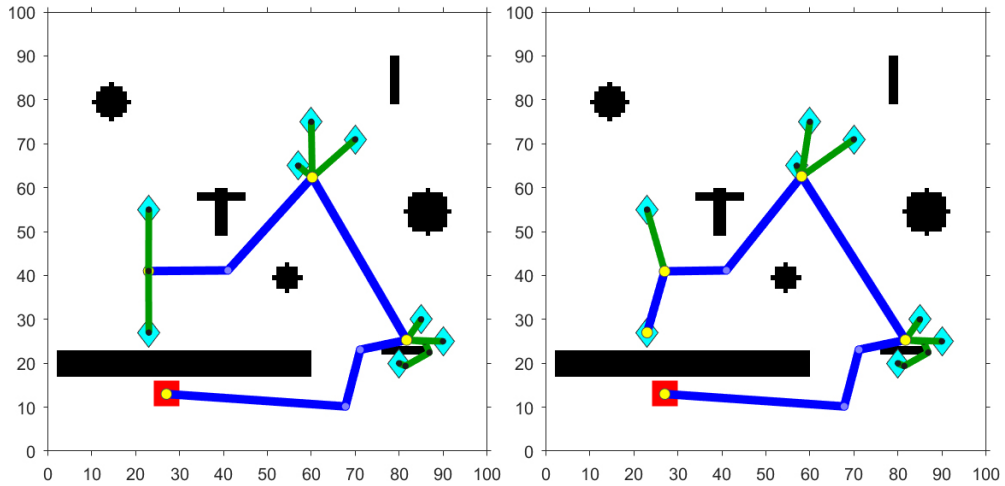
Figure 6.8: Deployment path planning in outdoor space without obstacle. ($v_c^{max} = 10.0m/s$, $a_c = 5.0m/s^2$, $v_r^{max} = 5.0m/s$, $a_r = 5.0m/s^2$, $t_u = 3.0s$, and $d_{deploy} = 15.0m$.)

method is shown in Figure 6.8a. Here, the number of nodes becomes 393 from the first 300, the number of edges is increased to 6,238, and the total time for the deployment is 27.14s. To compare this result with other results, in Figure

6.8b, we create the path without using the local PRM. As a result, the total deployment time is 27.58s. The time in Figure 6.8a and the time in Figure 6.8b show less difference than expected. In Figure 6.8c, instead of using local PRM, we use global PRM to reflect the increased number of nodes after using local PRM in Figure 6.8a. As a result, the number of nodes equals the first, and the number of edges is slightly less, however the time increases to 30.35s.

6.5.2 Outdoor space with obstacles

Second, the proposed algorithm is simulated for the outdoor space with obstacles shown in Figure 6.2b. All the parameters used for this simulation are the same as in Table 6.2, however v_r^{max} varies from $2.0m/s$ to $5.0m/s$. Figure 6.9 shows the path planning results for deployment. For eight tasks, three clusters are generated in Figure 6.9a. In the first cluster, the deployment location is



(a) $v_r^{max} = 5.0m/s$ (b) $v_r^{max} = 2.0m/s$

Figure 6.9: Deployment path planning in outdoor space with obstacles. ($v_c^{max} = 10.0m/s$, $a_c = 5.0m/s^2$, $a_r = 5.0m/s^2$, $t_u = 3.0s$, and $d_{deploy} = 15.0m$.)

determined at a node that can reach the tasks while avoiding the obstacle between tasks and move to the next cluster quickly. There is no obstacle around the second cluster. Therefore, the candidate node, made by the deterministic method, is chosen as the deployment location. For the last cluster, two rovers are deployed at the midpoint of the tasks. Meanwhile, four clusters are created in Figure 6.9b. This is because the maximum linear velocity of the rover is slower than that of Figure 6.9a, so that the rovers unloaded in the last cluster move longer to each task location. Therefore, the last cluster is divided into two clusters and each deployment location is found. As there is only one task in the last cluster in Figure 6.9b, the carrier directly moves to the task location. As a result, while the path of the carrier becomes longer, the total time becomes shorter than before the previous last cluster is divided. Table 6.4 shows the number of nodes and edges after the initial roadmap generation and after the last path generation. Note that, the total deployment time is 33.25s for the path in Figure 6.9a and 36.46s for the path in Figure 6.9b.

Table 6.4: Number of nodes and edges in outdoor space without obstacle

Step	Nodes	Edges
Global PRM	300	2,594
after Local PRM ($v_r^{max} = 5.0m/s$)	424	6,857
after Local PRM ($v_r^{max} = 2.0m/s$)	422	6,922

6.5.3 Office area

Third, the proposed algorithm is also simulated for the office area shown in Figure 6.2c. All the parameters used for this simulation are listed in Table 6.5.

In the table, d_{deploy} varies from $3.0m$ to $9.0m$. Table 6.6 shows the number of nodes and edges after the initial roadmap generation and after the last path generation.

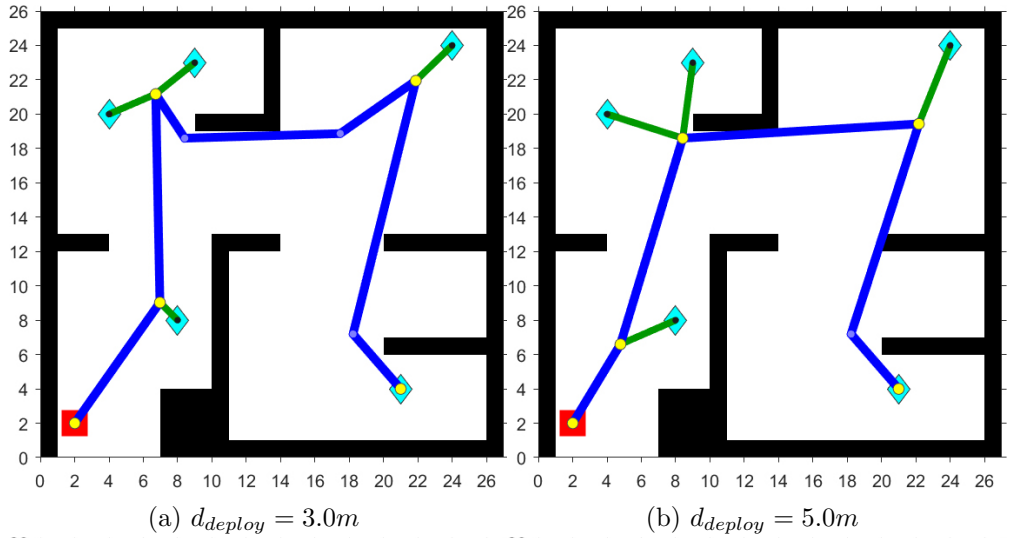
Table 6.5: Initial parameters in office area for five tasks in office area

Specification	carrier	$v_c^{max} = 10.0m/s$, $a_c = 5.0m/s^2$, $t_u = 3.0s$
	rover	$v_r^{max} = 5.0m/s$, $a_r = 5.0m/s^2$, $d_{deploy} = 3.0 - 9.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (2, 2)$
	tasks	$\mathbf{q}_G^1 = (24, 24)$, $\mathbf{q}_G^2 = (9, 23)$, $\mathbf{q}_G^3 = (21, 4)$ $\mathbf{q}_G^4 = (4, 20)$, $\mathbf{q}_G^5 = (8, 8)$

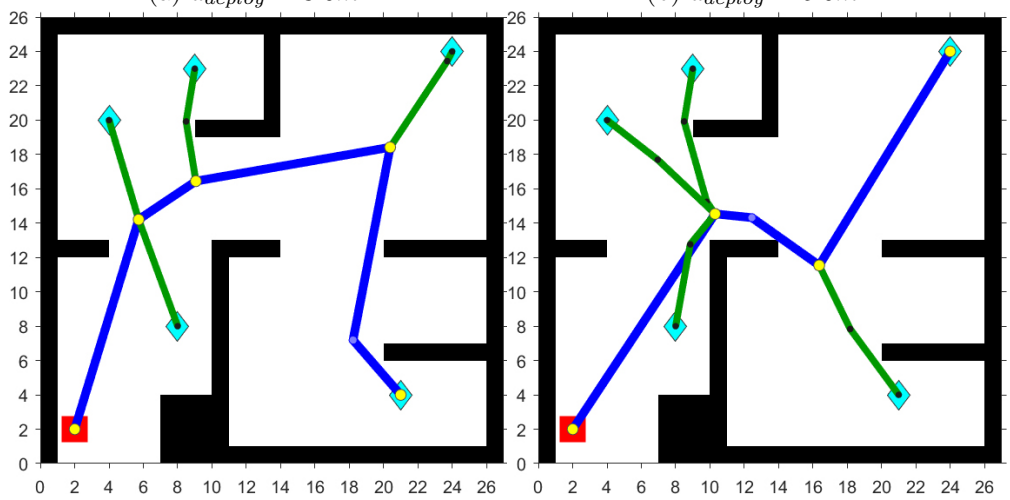
Table 6.6: Number of nodes and edges in office area

Step	Nodes	Edges
Global PRM	190	1,659
after Local PRM ($d_{deploy} = 3.0m$)	240	2,697
after Local PRM ($d_{deploy} = 5.0m$)	254	3,385
after Local PRM ($d_{deploy} = 7.0m$)	258	4,590
after Local PRM ($d_{deploy} = 9.0m$)	239	4,336

Figure 6.10 shows the path planning results to deploy rovers in the office area. All the results in the figure show that the path is well created without colliding with the wall of the office. Due to the maximum travel distance of the rover, the rover is unloaded for only one task at the first deployment location in Figure 6.10a and 6.10b. However, as the maximum travel distance increases, Figure 6.10c contains two tasks in the first cluster and 6.10d contains three tasks in the first cluster. As a result, the four deployment locations in Figure 6.10a decreases to three in Figure 6.10d. In addition, as the maximum travel distance of the rover becomes longer, the carrier unloads the rover farther from



(a) $d_{deploy} = 3.0m$ (b) $d_{deploy} = 5.0m$



(c) $d_{deploy} = 7.0m$ (d) $d_{deploy} = 9.0m$

Figure 6.10: Deployment path planning in office area. As the maximum travel distance of the rover increases, the length of carrier paths and the number of deployment locations decrease. ($v_c^{max} = 10.0m/s$, $a_c = 5.0m/s^2$, $v_r^{max} = 5.0m/s$, $a_r = 5.0m/s^2$, and $t_u = 3.0s$.)

the task. Since the speed of the rover is constant at $5m/s$, it is observed that the time required for all rovers to reach the assigned task location is gradually shortened from 25.42s to 24.25s, 23.56s, and 18.24s respectively.

6.5.4 University research building

Finally, the proposed algorithm is simulated for the university research building shown in Figure 6.2d. All the parameters used for this simulation is listed in Table 6.7 and the number of nodes and edges after the initial roadmap generation and after the last path generation is shown in Table 6.8.

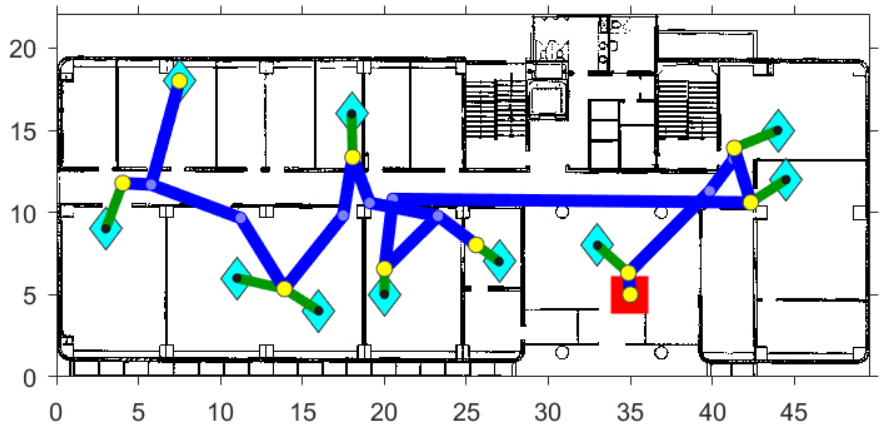
Table 6.7: Initial parameters in university research building

Specification	carrier	$v_c^{max} = 8.0m/s$, $a_c = 4.0m/s^2$, $t_u = 3.0s$
	rover	$v_r^{max} = 3.0m/s$, $a_r = 3.0m/s^2$, $d_{deploy} = 3.0 - 6.0m$
Locations	carrier	$\mathbf{q}_{c_1} = (35, 5)$
	tasks	$\mathbf{q}_G^1 = (3, 9)$, $\mathbf{q}_G^2 = (11, 6)$, $\mathbf{q}_G^3 = (16, 4)$ $\mathbf{q}_G^4 = (20, 5)$, $\mathbf{q}_G^5 = (27, 7)$, $\mathbf{q}_G^6 = (33, 8)$ $\mathbf{q}_G^7 = (44, 15)$, $\mathbf{q}_G^8 = (44.5, 12)$, $\mathbf{q}_G^9 = (7.5, 18)$ $\mathbf{q}_G^{10} = (18, 16)$

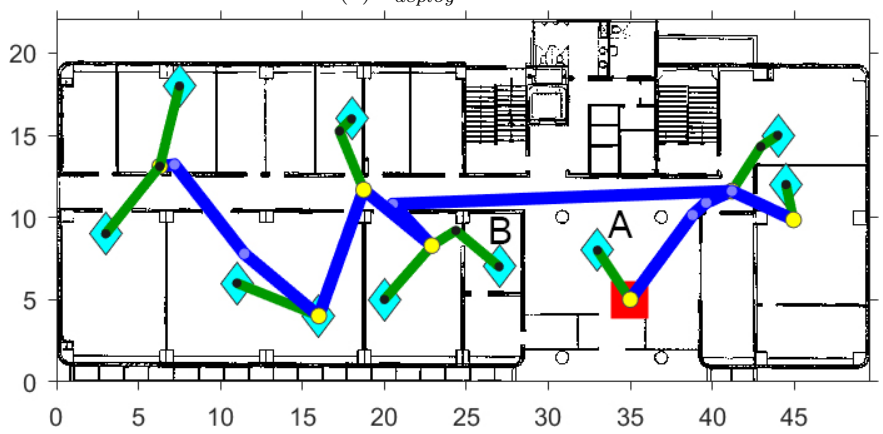
Table 6.8: Number of nodes and edges in university research building

Step	Nodes	Edges
Global PRM	190	1,659
after Local PRM ($d_{deploy} = 3.0m$)	412	5,708
after Local PRM ($d_{deploy} = 6.0m$)	419	5,368

Figure 6.11 shows the path planning results to deploy rovers in university research building which is shown in Figure 6.2d. This environment is more complex than the previous environments. In Figure 6.11a, the carrier visits



(a) $d_{deploy} = 3.0m$



(b) $d_{deploy} = 6.0m$

Figure 6.11: Deployment path planning in university research building. ($v_c^{max} = 8.0m/s$, $a_c = 4.0m/s^2$, $v_r^{max} = 3.0m/s$, $a_r = 3.0m/s^2$, and $t_u = 3.0s$.)

nine deployment locations whereas it visits seven locations in Figure 6.11b. Due to short maximum travel distance of rover in Figure 6.11a, the carrier enters further into the room from the hallway. The time for deployment is 57.86s. On the other hand, the time for deployment in Figure 6.11b is 45.25s. Initially, the task A above the carrier becomes a cluster with the task B in the figure. However, the two tasks cannot be reached from one location as there is

a wall. Therefore, the task B is excluded from the cluster again. As the distance to task A from the carrier's initial location is shorter than d_{deploy}, the carrier immediately deploys a rover at the beginning. The result shows that the carrier deploys the rovers in the corridor and sometimes goes into the room and deploys them depending on the distance constraints and the state of the cluster.

Chapter 7

Conclusion

In this study we introduced the practical path planning problems of a marsupial robot team for efficient deployment and collection. First, the motivation of the study using carrier for efficient operation of MRS was established (see chapter 1). The use of a carrier was suggested to decrease the overall duration for deployment and collection and to overcome the limitations of a rover. For the rigorous study, related studies were reviewed for MRS, path planning problems, and a marsupial robot team (see chapter 2). Through the review, we could confirm that there are very few studies on the path planning of a marsupial robot team, although there are studies related to a marsupial robot team. We defined the problems to be solved for the efficient deployment and collection of the marsupial robot team (see section 4.1, 5.1, and 6.1) and analyzed the computational complexity of the problem (see section 4.2). Since calculating the optimal solution in this setting requires so much computation that it is almost impos-

sible, efficient algorithms were proposed (see section 4.4, 5.4, and 6.4.4). The algorithms are designed considering the maximum traveling distance of a rover, the dynamics of the carrier and rovers, and the dynamicity of the environment. To reduce the computational complexity of the problems, we proposed using a simple clustering method which uses geographical information of the tasks or rovers. By using the clustering method, the entire problem can be divided into several sub-problems. Then we showed the optimal solution can be acquired for each sub-problem. Finally, the entire solution of the path was computed by merging the solutions of the sub-problems.

Based on several scenarios, the feasibility of the proposed algorithms was shown by the simulations (see section 4.5, 5.5, and 6.5). For the deployment of the rovers, the deployment for two tasks was demonstrated, then the deployment for 15 tasks was also demonstrated varying the dynamics of the carrier and the rovers. For the collection of the rovers, the 15 rovers, which have been previously deployed, were collected varying their remaining energy. The simulation results imply as follows:

1) Increase of loading/unloading time may cause increase of the number of clusters;

2) Increase of the carrier's speed may result in increase of the number of clusters, and the length of the carrier's travel distance;

3) Increase of the rover's speed may shorten the length of the carrier's travel distance.

Both the deployment and collection simulations were shown in 3D space. It

was also shown that the proposed algorithms are applicable in a dynamic environment where the tasks disappear or are created and the rovers disappear. However, dynamic environments in robotics usually involve different sources of uncertainty. Therefore more uncertain factors should be investigated. Finally, the performance evaluation showed that the near real-time rover can be implemented for large-size fleets of rovers or tasks by using the proposed algorithms (see section 4.6 and 5.6). The efficiency of the generated path was compared with the near optimal solution of the TSP, greedy two-opt.

In addition, we suggested a novel method for a marsupial robot team deployment using a graph (see chapter 6). The overall framework for this was proposed first (see section 6.2). We used PRM to create roadmap, and created a more efficient roadmap by dividing the global PRM and the local PRM (see section 6.3). Next, the given task locations were clustered based on the path length from the carrier's position. Then we use the generated roadmap to search A* algorithm and find the optimal deployment node satisfying the distance constraint. The simulation results (see section 6.5) show that this method can be used for exploration and urban search and rescue. To improve the performance of this method, we can consider using PRM variants such as visibility-PRM [83]. In addition, a more fundamental improvement can be achieved by using dynamic programming techniques to calculate the more efficient overall path. However this will require more computation resources. To use this study in real-world environments, it is necessary to extend the problem to non-holonomic robots and also to create a curved path.

The room for improvement is as follows.

- The transmission problem is not considered in this study: we only assumed that each rover can communicate with the carrier via a wireless network. However, this assumption limits the work's significance. First, we need to determine the kind of network topology. Moreover, the carrier may have to deploy the network sensors according to the size of the space. Second, the energy required for transmissions should be considered for yielding equations.

- Using two or more carrier can maximize the advantages of the proposed collaborative system. To realize this, another task allocation method should be investigated.

- This study assumes that each rover can perform only one task. However, some of rovers may move shorter than other rovers and leave enough energy to move and perform another tasks. This may make the analysis more intricate and complex.

- Different task durations may entail the use of temporal windows.

- The collection algorithm has an implicit assumption that all rovers will be collected after they have finished task. However, since the end of the operation can be estimated, expansions that create more efficient paths can be considered by starting the carrier earlier.

- The proposed method is used in a known environment, however, some tasks such as exploration should assume an unknown environment.

Summarizing the above, we plan to extend this research in four aspects for future work.

1) Using more than one carrier for efficient deployment and collection.

2) Conducting experiments with real robots by implementing the system. This work includes the development of communication protocol between robots, positioning system, and etc.

3) Deployment and collection of a marsupial robot team in various roadmaps.

4) Expansion into path planning for long-term deployment.

Bibliography

[1] N. Agmon, C.-L. Fok, Y. Emaliah, P. Stone, C. Julien, and S. Vishwanath, "On coordination in practical multi-robot patrol," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 650–656.

[2] A. Amanatiadis, C. Henschel, B. Birkicht, B. Andel, K. Charalampous, I. Kostavelis, R. May, and A. Gasteratos, "Avert: An autonomous multi-robot system for vehicle extraction and transportation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1662–1669.

[3] N. M. Amato, O. B. Bayazit, and L. K. Dale, "Obprm: An obstacle-based prm for 3d workspaces," 1998.

[4] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. (2006) Concorde tsp solver. [Online]. Available: http://www.math.uwaterloo.ca/tsp/concorde/

[5] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in multi-robot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 655–661, 2002.

[6] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[7] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica*, vol. 31, no. 03, pp. 345–359, 2013.

[8] U. Baroudi, G. Sallam, M. Al-Shaboti, and M. Younis, "Gps-free robots deployment technique for rescue operation based on landmark's criticality," in *Proceedings of International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2015, pp. 367–372.

[9] B. R. W. Beard, T. W. McLain, D. B. Nelson, D. Kingston, and D. Johanson, "Decentralized cooperative aerial surveillance using fixed-wing miniature uavs," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.

[10] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs," in *Cooperative Control: Models, Applications and Algorithms*. Springer, 2003, pp. 23–41.

[11] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[12] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2001, pp. 271–276.

[13] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 1957–1962.

[14] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.

[15] bluAir. (2015) Bluetooth Range: 100m, 1km, or 10km? [Online]. Available: http://www.bluair.pl/bluetooth-range

[16] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2002, pp. 2383–2388.

[17] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 476–481.

[18] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[19] A. Campbell and A. S. Wu, "Multi-agent role allocation: issues, approaches, and multiple perspectives," *Autonomous agents and multi-agent systems*, vol. 22, no. 2, pp. 317–355, 2011.

[20] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous Robots*, vol. 25, no. 3, pp. 305–316, 2008.

[21] S. Carpin, T. H. Chung, and B. M. Sadler, "Theoretical foundations of high-speed robot team deployment," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2033–2040.

[22] T.-C. Chen, T.-S. Chen, and P.-W. Wu, "On data collection using mobile robot in wireless sensor networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 6, pp. 1213–1224, 2011.

[23] P. Cheng, J. Fink, V. Kumar, and J.-S. Pang, "Cooperative towing with multiple robots," *Journal of mechanisms and robotics*, vol. 1, no. 1, p. 011008, 2009.

[24] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[25] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[26] M. S. Couceiro, C. M. Figueiredo, D. Portugal, R. P. Rocha, and N. M. Ferreira, "Initial deployment of a robotic team-a hierarchical approach

under communication constraints verified on low-cost platforms," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4614–4619.

[27] F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings, "Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 469–476.

[28] A. Drenner, M. Janssen, A. Kottas, A. Kossett, C. Carlson, R. Lloyd, and N. Papanikolopoulos, "Coordination and longevity in multi-robot teams involving miniature robots," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 263–284, 2013.

[29] M. et al. (2013) Mars 2020 Rover. [Online]. Available: http://mars.jpl.nasa.gov/mars2020

[30] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[31] G. Gan, C. Ma, and J. Wu, *Data clustering: theory, algorithms, and applications*. Siam, 2007, vol. 20.

[32] V. Gazi and K. M. Passino, "Stability analysis of social foraging swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 539–557, 2004.

[33] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[34] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[35] N. Ghods, P. Frihauf, and M. Krstic, "Multi-agent deployment in the plane using stochastic extremum seeking," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5505–5510.

[36] A. Gil, Ó. Reinoso, M. Ballesta, and M. Juliá, "Multi-robot visual slam using a rao-blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68–80, 2010.

[37] J. Goerner, N. Chakraborty, and K. Sycara, "Energy efficient data collection with mobile robots in heterogeneous sensor networks," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2527–2533.

[38] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information processing letters*, vol. 1, no. 4, pp. 132–133, 1972.

[39] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.

[40] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[41] J. A. Hartigan, "Statistical theory in clustering," *Journal of classification*, vol. 2, no. 1, pp. 63–76, 1985.

[42] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[43] G. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. S. Sukhatme, M. Stojanovic, H. Singh, F. Hover *et al.*, "Underwater data collection using robotic sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 899–911, 2012.

[44] H. Hourani, P. Wolters, E. Hauck, and S. Jeschke, "A marsupial relationship in robotics: a survey," in *Proceedings of International Conference on Intelligent Robotics and Applications*. Springer, 2011, pp. 335–345.

[45] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, no. 12, pp. 1243–1256, 2006.

[46] P. D. Hung, M.-T. Pham, T. Q. Vinh, and T. D. Ngo, "Self-deployment strategy for a swarm of robots with global network preservation to assist rescuers in hazardous environments," pp. 2655–2660, 2014.

[47] A. Ismail, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, no. 4, pp. 341–344, 2008.

[48] W. Jacak, "Intelligent robotic systems," *Intelligent Robotic Systems: Design, Planning, and Control*, pp. 9–19, 2002.

[49] W. Jatmiko, K. Sekiyama, and T. Fukuda, "A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement," *IEEE Computational Intelligence Magazine*, vol. 2, no. 2, pp. 37–51, 2007.

[50] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[51] E. Kadioglu and N. Papanikolopoulos, "A method for transporting a team of miniature robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 2297–2302.

[52] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[53] B.-I. Kim, J.-I. Shim, and M. Zhang, "Comparison of tsp algorithms," *Project for Models in Facilities Planning and Materials Handling*, 1998.

[54] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[55] E. Lalish, K. A. Morgansen, and T. Tsukamaki, "Decentralized reactive collision avoidance for multiple unicycle-type vehicles," in *Proceedings of American Control Conference (ACC)*, 2008, pp. 5055–5061.

[56] J. C. Las Fargeas, P. T. Kabamba, and A. R. Girard, "Path planning for information acquisition and evasion using marsupial vehicles," in *Proceedings of American Control Conference (ACC)*, 2015, pp. 3734–3739.

[57] S. M. LaValle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79–89, 2011.

[58] S. M. LaValle, S. Hutchinson *et al.*, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[59] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[60] H. Lee, J. Jeon, and B. Lee, "An efficient cooperative deployment of robots for multiple tasks," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5419–5425.

[61] H. Lee and B. Lee, "Energy constrained collection of multiple robots in 3d space," *Electronics Letters*, vol. 53, no. 4, pp. 231–233, 2017.

[62] H. Lee, H. Yoo, and B. Lee, "Deployment method of uavs with energy constraint for multiple tasks," *Electronics Letters*, vol. 51, no. 21, pp. 1650–1652, 2015.

[63] H. Lee, J. Oh, J. Jeon, and B. Lee, "Efficient deployment of energy-constrained unmanned aerial vehicles in 3-dimensional space," in *Proceedings of International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2016, pp. 446–451.

[64] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[65] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[66] S. Lin, "Computer solutions of the traveling salesman problem," *The Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269, 1965.

[67] K. D. Listmann, V. Willert *et al.*, "Discoverage: A new paradigm for multi-robot exploration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 929–934.

[68] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[69] A. Macwan, G. Nejat, and B. Benhabib, "Optimal deployment of robotic teams for autonomous wilderness search and rescue," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4544–4549.

[70] E. Masehian and D. Sedighizadeh, "An improved particle swarm optimization method for motion planning of multiple robots," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 175–188.

[71] S. A. Masoud and A. A. Masoud, "Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 32, no. 6, pp. 705–723, 2002.

[72] M. J. Matarić, M. Nilsson, and K. T. Simsarin, "Cooperative multi-robot box-pushing," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1995, pp. 556–561.

[73] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[74] M. Matusiak, J. Paanajärvi, P. Appelqvist, M. Elomaa, M. Vainio, T. Ylikorpi, and A. Halme, "A novel marsupial robot society: towards long-term autonomy," in *Distributed Autonomous Robotic Systems 8*. Springer, 2009, pp. 523–532.

[75] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[76] H. J. Min and N. Papanikolopoulos, "Vision-based effective dispersion of miniature robots by using local sensing," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, pp. 73 321V–73 321V.

[77] J. Minguez, L. Montano, and O. Khatib, "Reactive collision avoidance for navigation with dynamic constraints," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2002, pp. 588–594.

[78] B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire, "A communication-free behavior for docking mobile robots," in *Distributed autonomous robotic systems 4*. Springer, 2000, pp. 357–367.

[79] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, 2002, pp. 593–598.

[80] R. R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue," *IEEE Intelligent Systems and their applications*, vol. 15, no. 2, pp. 14–19, 2000.

[81] R. R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, and L. Pollock, "Marsupial-like mobile robot societies," in *Proceedings of Annual conference on Autonomous Agents*, 1999, pp. 364–365.

[82] S. S. Nestinger and H. H. Cheng, "Mobile-r: A reconfigurable cooperative control platform for rapid deployment of multi-robot systems," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 52–57.

[83] C. Nissoux, T. Siméon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1999, pp. 1316–1321.

[84] L. M. Noz-Gómez, M. Alencastre-Miranda, R. Lopez-Padilla, and R. Murrieta-Cid, "Exploration and map-building under uncertainty with multiple heterogeneous robots," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2295–2301.

[85] J. H. Oh, S.-H. Lee, and B. H. Lee, "Accurate visual loop-closure detection using bag-of-words for multiple robots," *Journal of Automation and Control Engineering*, vol. 3, no. 5, 2015.

[86] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[87] 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.

[88] L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, 1997, pp. 2082–2089.

[89] M. Peasgood, C. M. Clark, and J. McPhee, "A complete and scalable strategy for coordinating multiple robots within roadmaps," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 283–292, 2008.

[90] Y. Pei and M. W. Mutka, "Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1551–1556.

[91] R. Regele and P. Levi, "Cooperative multi-robot path planning by heuristic priority adjustment," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 5954–5959.

[92] I. Rekleitis, R. Sim, G. Dudek, and E. Milios, "Collaborative exploration for map construction," in *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2001, pp. 296–301.

[93] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 324–333, 2008.

[94] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos, "Multi-robot three-dimensional coverage of unknown areas," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 738–752, 2012.

[95] G. G. Rigatos, "Distributed gradient and particle swarm optimization for multi-robot motion planning," *Robotica*, vol. 26, no. 03, pp. 357–370, 2008.

[96] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automations*, vol. 8, no. 5, pp. 501–518, 1992.

[97] C. Rossi, L. Aldama, and A. Barrientos, "Simultaneous task subdivision and allocation for teams of heterogeneous robots," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 946–951.

[98] C. Rossi, L. Aldama, A. Barrientos, A. Valero, and C. Cruz, "Negotiation of target points for teams of heterogeneous robots: an application to exploration," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 5868–5873.

[99] P. E. Rybski, N. P. Papanikolopoulos, S. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, M. D. Erickson *et al.*, "Enlisting rangers and scouts for reconnaissance and surveillance," *Robotics & Automation Magazine, IEEE*, vol. 7, no. 4, pp. 14–24, 2000.

[100] T. Sahin and E. Zergeroglu, "A computationally efficient path planner for a collection of wheeled mobile robots with limited sensing zones," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 1074–1079.

[101] M. Saska, T. Krajník, and L. Pfeucil, "Cooperative μuav-ugv autonomous indoor surveillance," in *Proceedings of International Multi-Conference on Systems, Signals and Devices (SSD)*, 2012, pp. 1–6.

[102] C. Satish, "Inter-vehicular communication for collision avoidance using wi-fi direct," 2014.

[103] I. Satoh, "Coordination and deployment of mobile agents on dependable systems," in *Proceedings of International Conference on Dependability (DEPEND)*, 2010, pp. 139–145.

[104] E. Schikuta, "Grid-clustering: An efficient hierarchical clustering method for very large data sets," in *Proceedings of IEEE International Conference on Pattern Recognition*, vol. 2, 1996, pp. 101–105.

[105] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 811–824, 2007.

[106] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[107] C. Stachniss, *Robotic mapping and exploration*. Springer, 2009, vol. 55.

[108] Y. Stergiopoulos and A. Tzes, "Autonomous deployment of heterogeneous mobile agents with arbitrarily anisotropic sensing patterns," in *Proceedings of IEEE Mediterranean Conference on Control & Automation (MED)*, 2012, pp. 1585–1590.

[109] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 1989.

[110] J. Tran, A. Ferworn, M. Gerdzhev, and D. Ostrom, "Canine assisted robot deployment for urban search and rescue," in *Proceedings of IEEE International Workshop on Safety Security and Rescue Robotics (SSRR)*, 2010, pp. 1–6.

[111] Y. Uchimura, T. Imaizumi, and H. Murakami, "Mobile robot deployment based on voronoi diagram," in *Proceedings of International Symposium on Access Spaces (ISAS)*, 2011, pp. 71–76.

[112] J. van Den Berg, J. Snoeyink, M. C. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans." in *Robotics: Science and systems*, vol. 2, no. 2.5, 2009, pp. 2–3.

[113] J. Wang, C. Smith, G. Staskevich, and B. Abbe, "A distributed deployment algorithm for mobile robotic agents with limited sensing/communication ranges," in *Proceedings of IEEE International Conference on Electro/Information Technology (EIT)*, 2015, pp. 530–535.

[114] N. Wang, A. Liang, and H. Guan, "A multi-robot self-deployment method based on particle swarm optimization," in *Proceedings of IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, 2010, pp. 152–156.

[115] Y.-H. Wang, C.-H. Tsai, and Y.-H. Wu, "Robot-based deployment mechanism for wireless sensor networks in unknown region," in *Proceedings of International Joint Conference on Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA)*, 2013, pp. 143–149.

[116] Z. Wang, X. Zhao, and X. Qian, "Carrier-based sensor deployment by a mobile robot for wireless sensor networks," in *Proceedings of International Conference on Control Automation Robotics & Vision (ICARCV)*, 2012, pp. 1663–1668.

[117] L. Wilkinson, "Exact and approximate area-proportional circular venn and euler diagrams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 321–331, 2012.

[118] K. M. Wurm, C. Dornhege, B. Nebel, W. Burgard, and C. Stachniss, "Coordinating heterogeneous teams of robots using temporal symbolic planning," *Autonomous Robots*, vol. 34, no. 4, pp. 277–294, 2013.

[119] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 1160–1165.

[120] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, p. 399, 2013.

[121] C.-w. Zheng, M.-y. Ding, and C.-P. Zhou, "Cooperative path planning for multiple air vehicles using a co-evolutionary algorithm," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 1, 2002, pp. 219–224.

초록

본 논문은 다중 로봇 시스템에서 에너지 제약을 갖고 있는 이동 로봇을 효율적으로 배치하고 회수하는 문제를 다룬다. 기존에도 다중 로봇 시스템을 위한 이동 로봇들의 작업 할당 및 경로 계획 등의 연구는 활발히 이루어졌다. 이러한 연구들은 일반적으로 이동 로봇들이 이미 여러 장소에 흩어진 상황에서 최적의 해를 구하는 방향으로 주로 진행되어왔다. 한 편 초기 상태의 한 장소에서 시작하여 이동 로봇들을 효율적으로 배치하는 연구는 아직 부족한 실정이다. 로봇을 이용해 수행하고자 하는 작업에 따라 이동 로봇을 빠르게 배치하는 문제는 매우 중요해질 수 있다. 또한 이동 로봇들을 회수하는 문제 또한 기존에 거의 다루어지지 않았다. 실제 임무 수행을 마친 로봇들을 회수하는 것은 재사용 등의 측면에서 꼭 필요한 연구 분야이다. 이동 로봇은 그 특성상 에너지의 제약을 가지고 있다. 본 논문에서는 배치 및 회수의 두 문제를 에너지 제약이라는 가정 하에 다룬다. 즉, 에너지 제약을 갖는 이동 로봇의 배치 문제는 모든 이동 로봇들이 주어진 작업 위치에 도달하기까지의 소요 시간을 최소화하는 것이고, 에너지 제약을 갖는 이동 로봇의 회수 문제는 모든 이동 로봇들이 작업을 마친 위치로부터 복귀하여 회수되는데 필요한 소요 시간을 최소화하는 것이다. 본 논문에서는 이동 로봇의 에너지 제약을 고려함과 동시에 보다 효율적인 배치 및 회수를 위해 이동 로봇보다 상대적으로 크며 속도도 빠른 수송 로봇이 포함된 marsupial 로봇 팀의 활용을 제안한다. 여기서 수송 로봇은 여러 대의 이동 로봇을 싣고 수송할 수 있으며, 정해진 위치에서 이동 로봇을 배치하고 다시 회수하는 역할을 담당한다. 이처럼 수송 로봇을 이용해 이동 로봇을 배치하는 문제는 몇몇 연구에서 다루어지기도 했다. 그러나 이 연구

들은 주로 기계적인 메커니즘 등의 기능 위주로 연구가 진행되었고, 효율적인 경로 생성을 위한 연구는 미흡한 실정이다.

본 논문에서 이용되는 수송 로봇은 주어진 목표를 달성하는데 충분한 에너지를 갖고 있다고 가정한다. 그런데 이러한 환경에서 최적의 배치 지점 및 회수 지점을 결정하기 위해서는 NP-Hard 문제로 알려진 외판원 문제보다도 더 많은 계산을 필요로 하는 점이 문제이다. 이러한 계산 요구 사항을 줄이기 위해서는 목표 작업 위치 또는 회수하고자 하는 이동 로봇의 위치와 이동 로봇의 가용한 에너지 등을 고려하여 인접한 대상들을 클러스터로 무리지을 필요가 있다. 클러스터를 나누는 방법은 모든 경우를 고려할 경우 계산의 복잡도가 증가하여 단순한 알고리즘을 이용해 나눈다. 그 다음으로는 클러스터들과 수송 로봇 간의 전후 위치를 고려하여 클러스터의 순서를 정할 수 있고, 최종적으로 각 클러스터에 해당하는 효율적인 배치 및 회수 지점을 계산할 수 있게 된다. 결국 수송 로봇은 계산된 지점들을 순차적으로 방문하게 되며, 이동 로봇의 효율적인 배치 및 회수를 달성할 수 있다. 또한 본 논문에서는 로드맵을 생성하고 이를 이용해 marsupial 로봇팀의 배치를 효율적으로 달성하는 방법도 제안한다. 이 방법을 이용하면 고정된 장애물을 자연스럽게 회피할 수 있는 경로를 얻어낼 수 있다. 제안된 방법은 최적의 해를 보장할 수는 없지만 빠른 시간 안에 적은 양의 계산으로 효율적인 경로를 생성할 수 있다는 장점이 있다. 또한 이 방법은 여러 대의 이동 로봇 및 작업뿐만 아니라 3차원 공간과 동적인 환경도 고려하여 고안되었다. 본 논문에서 제안된 방법은 로봇들의 동역학을 고려하여 설계된 시뮬레이션 프로그램을 통해 검증되었으며, 같은 문제에 대한 외판원 문제의 해와 비교하여 더 효율적인 경로를 생성함을 확인할 수 있었다.

주요어: 다중 로봇 시스템, 다중 로봇 경로계획, marsupial 로봇, 배치, 회수, 에너지 제약

학번: 2012-30227