FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

EFFICIENT MISSION PLANNING FOR ROBOT NETWORKS IN COMMUNICATION CONSTRAINED ENVIRONMENTS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Md Mahbubur Rahman

2017

To: Interim Dean Ranu Jung
College of Engineering and Computing

This dissertation, written by Md Mahbubur Rahman, and entitled Efficient Mission Planning for Robot Networks in Communication Constrained Environments, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Bogdan Carbunar

Ning Xie

Wei Zeng

Ali Mostafavi

Leonardo Bobadilla, Major Professor

Date of Defense: June 06, 2017

The dissertation of Md Mahbubur Rahman is approved.

Interim Dean Ranu Jung
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2017

DEDICATION

To my family for their unconditional love and support.

ACKNOWLEDGMENTS

The research works presented in this dissertation came into a success because of many people who helped me in many ways. First of all, I would like to thank my adviser Dr Leonardo Bobadilla who paved the way for completing the research in time through his continuous support and great advice. He made robotics very interesting to me and we had many fruitful hours of discussions on research topics while eating in restaurants, attending conferences, walking around and riding in car.

My gratitude goes to other committee members, Dr Bogdan, Dr Ali, Dr Ning and Dr Zeng for their time and valuable feedback. I would like to thank my sibling research mates, Sebastian, Tauhid, and Greg. I not only shared a lab with these amazing guys but also discussed many research ideas and they provided me lots of constructive feedback about my presentations. Three other guys Triana, Franklin and Carlos helped me with the experiments and simulation works and I am really proud to be a mentor of these bright students.

I am grateful to Florida International University's Graduate School for supporting me with the Dissertation Year Fellowship award. My special gratitude goes to US Army Research Lab for funding a number of my research projects.

ABSTRACT OF THE DISSERTATION

EFFICIENT MISSION PLANNING FOR ROBOT NETWORKS IN COMMUNICATION CONSTRAINED ENVIRONMENTS

by

Md Mahbubur Rahman

Florida International University, 2017

Miami, Florida

Professor Leonardo Bobadilla, Major Professor

Many robotic systems are remotely operated nowadays that require uninterrupted connection and safe mission planning. Such systems are commonly found in military drones, search and rescue operations, mining robotics, agriculture, and environmental monitoring. Different robotic systems may employ disparate communication modalities such as radio network, visible light communication, satellite, infrared, Wi-Fi. However, in an autonomous mission where the robots are expected to be interconnected, communication constrained environment frequently arises due to the out of range problem or unavailability of signal. Furthermore, several automated projects (building construction, assembly line) do not guarantee uninterrupted communication, and a safe project plan is required that optimizes collision risks, cost and duration. In this thesis, we propose four pronged approaches to alleviate some of these issues: 1) Communication aware world mapping; 2) Communication preserving using the Line-of-Sight (LoS); 3) Communication aware safe planning; and 4) Multi-Objective motion planning for navigation.

First, we focus on developing a communication aware world map that integrates traditional world models with the planning of multi-robot placement. Our proposed communication map selects the optimal placement of a chain of intermediate relay vehicles in order to maximize communication quality to a remote unit. We also

propose an algorithm to build a min-Arborescence tree when there are multiple remote units to be served.

Second, in communication denied environments, we use Line-of-Sight (LoS) to establish communication between mobile robots, control their movements and relay information to other autonomous units. We formulate and study the complexity of a multi-robot relay network positioning problem and propose approximation algorithms that restore visibility based connectivity through the relocation of one or more robots.

Third, we develop a framework to quantify the safety score of a fully automated robotic mission where the coexistence of human and robot may pose a collision risk. A number of alternate mission plans are analyzed using motion planning algorithms to select the safest one.

Finally, an efficient multi-objective optimization based path planning for the robots is developed to deal with several Pareto optimal cost attributes.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Motivation

The connected networks of multiple autonomous robots have an increasing demand for many risky and labor intensive tasks such as military missions, search and rescue operations, construction automation, autonomous mining, health care, and environmental monitoring. Often, these robots are highly mobile and are deployed in groups to remote locations where humans cannot safely venture (e.g. interplanetary space, deep oceans, toxic gas tanks). They are able to perform tasks with levels of precision that are not achievable by human hands, such as micron-level slicing of materials in manufacturing or minimally invasive surgical incising. Even in scenarios where humans have historically been in-the-loop, today aerial, ground and underwater vehicles are able to completely or partially eliminate the need of a physical human presence. Virtual reality has been a promising research topic in recent years and is useful for telepresence or artificially simulating environments that imitate the real world scenario. All of these would not have been made possible without continuous, unprecedented advancement in intelligent robotics research.

Many companies and government research agencies are heavily investing in different robotic research areas, and the technology is rapidly evolving in order to eliminate the need for, or more effectively leverage human effort. Recently, in 2015, DARPA awarded $2M USD to its Grand Robotic Challenge winner KAIST from Korea, $1M to IHMC in Florida, who placed second, and $50K to CHIMP of Carnegie Melon, who held the third place position. All the participant robots were teleoperated from a remote location in both natural and man-made environments [ihm], and were specially designed for disaster response. These humanoid robots (see Figure

1.1(a)) are highly advanced, having the capability of completing challenging tasks, such as search and rescue operations in disaster areas, climbing ladders, walking on rubble and manipulating gas hoses. Similarly, the United States Army is investing

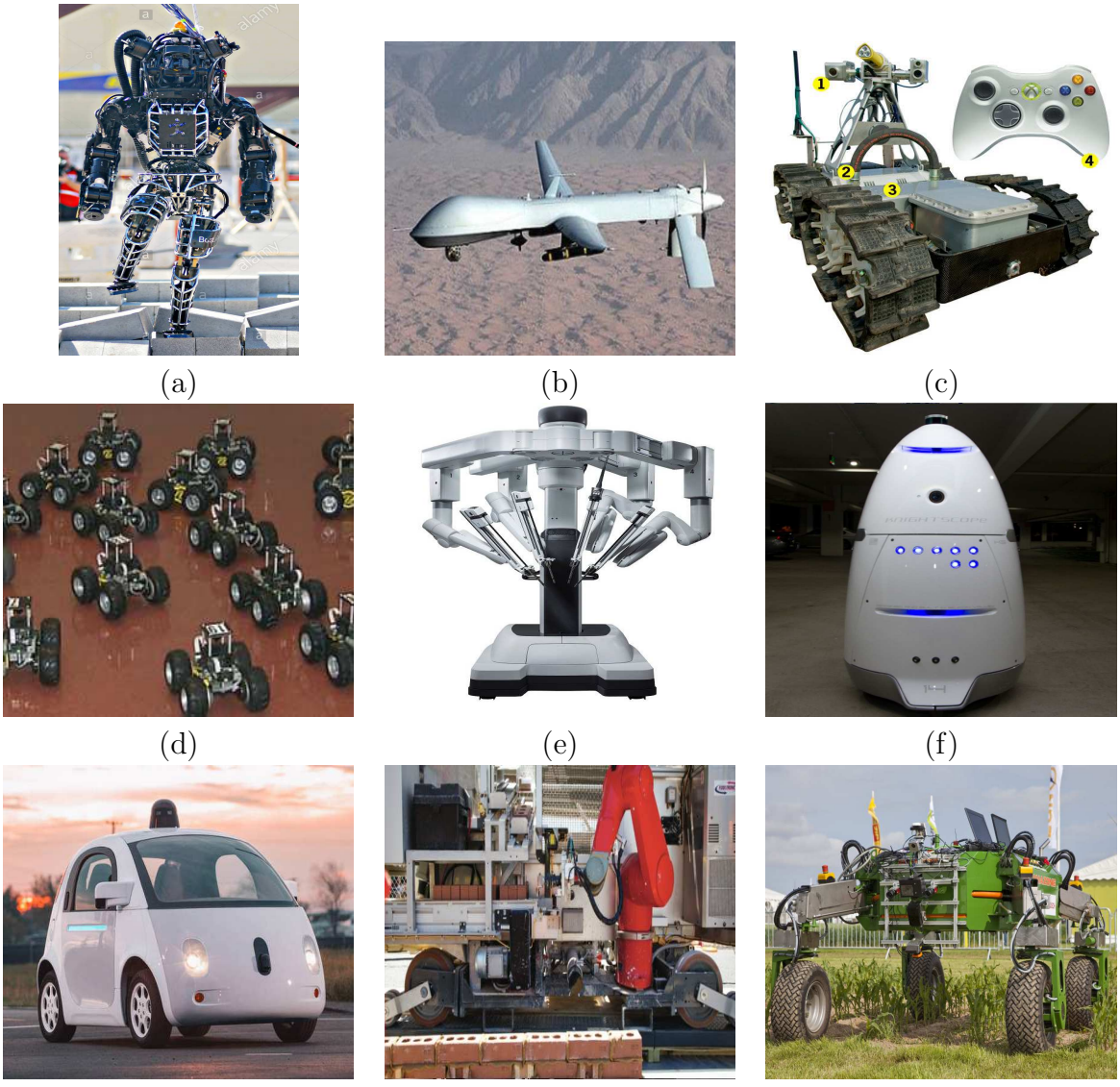


Figure 1.1: (a) IHMC humanoid robot [JSB+15]; (b) US Military drone (UAV) [Zen13]; (c) Sandia Lab's mining drone [ZLLZ08]; (d) Sandia's robot swarm [BHEH02]; (e) da Vinci surgical system developed by Intuitive Surgical [dVS]; (f) K5 security robot [NS09]; (g) Google's waymo self driving car [Rim17]; (h) SAM-100 mason robot [Rob]; (i) BoniRob agricultural robot from Bosch [RBD+09].

heavily on remote controlled unmanned aerial, ground, and maritime vehicles, and has forecasted an increase in spending on research and development of such systems will increase from $6.6 billion in 2013 to $11.4 billion in 2022 [DMC+14]. These systems are operated from a distance that is safe for the operator and therefore reduces fatalities by decreasing the amount of manned missions in adversarial environments.

Sandia National Lab developed their Gemini Robot System as shown in Figure 1.1(c) to explore underground mines and tunnels. They are able to traverse through debris, water, mud, flooding, explosive vapors, poisonous gases, and a variety of other environmental conditions where human teams cannot navigate quickly or safely. These robots are remotely operated and fully equipped with modern cameras and sensors in order to perceive environmental and structural conditions, and serve as two-way communications devices with miners. Sandia has also developed a cooperative squad of robot vehicles (see Figure 1.1(d)) that can be used for fighting forest fires, cleaning up oil spills, delivering and distributing supplies to remote field operations, and conducting military missions. A single operator plans a set of tasks for a squad of robotic vehicles and the coordinated system collectively achieves the goal that would otherwise require many humans to be present in risky environments such as battlefield, nuclear disaster areas.

Nowadays, in the medical sciences, robotic surgery has become very common. Intuitive Surgical's da Vinci system (see Figure 1.1(e)) has already performed successful operations on three million patients, and every 60 seconds someone around the world is receiving minimal invasive surgery from this advanced technology [int]. Robotics research has made significant advancements in security research and medical research. Knightscope has developed their K5 security robots as shown in Figure 1.1(f) that can visualize 360° around it, detect possible threats and report them to

the remote security operation center. In 2016, a number of K5 robots were deployed in the Stanford Shopping Center for a cost of only $7/hour [PPBGC17].

Another very exciting advancement of robotics research is self driving cars, where many companies are aggressively investing to win the race of fully driverless cars (L5 autonomy). Google's autonomous car Waymo (see Figure 1.1(g)) has already driven millions of miles. Intel recently purchased Mobileye and their vision based perception systems, while NVIDIA Corporation is training their cloud based deep neural network to control self driving cars. Most giant car makers, including Honda, Toyota, GM, and Ford are also investing heavily in this area and the fully autonomous cars are predicted to be on the market by 2021 [bmw].

Robotics research advancements have heavily impacted modern agriculture, construction jobs, and manufacturing and assembly, where humans and robots coordinate to complete a bigger task. Therefore, safe and efficient planning for sequences of activities, which at the same time meet the project timeline are required. The SAM-100 robot, shown in Figure 1.1(h), is an automated mason robot developed by Construction-Robotics that is able to lay bricks in construction sites six times faster than humans [sam]. Autonomous dump trucks and cranes that will reduce accidents in construction sites by employing safe motion plans are under heavy research focus. Bosch developed a robotic platform called BoniRob (see Figure 1.1(i)) to be modified for various jobs in agriculture. This system can navigate autonomously along plant rows (e.g. Dams) in the field, carrying the application module (tool) as it moves. Multiple high end sensors such as LiDAR, inertial sensors, wheel odometry and GPS are mounted for row detection and navigation.

1.2 Fundamental Challenges and Key Themes

The success of an autonomous mission using the aforementioned robotic systems depends on two common phenomena, 1) communication, and 2) efficient planning. Most of the systems are remotely controlled and require an uninterrupted control signal from the command center. Additionally, mutual connectivity among the deployed robotic systems is necessary to achieve a goal by a collective effort from a team of robots. Efficient sequence of activities is also required in time critical projects (e.g. automated construction, assembly line) which at the same time ensure a safe collocation of humans and robots. However, traditional communication mediums such as mobile networks, GPS, radio among the robots in a multi-robotic system may not be readily available or may be very primitive, which results in a communication denied environment.

Communication Denied Environment: The conventional communication among the autonomous robots/vehicles and human operators can be interrupted and degraded by many factors, including mission related/random movements, out of range locations, physical obstructions, atmospheric conditions, electromagnetic interference, and adversarial attacks such as jamming and sniffing. One important problem with these systems is that robots cannot be properly controlled in sensor and communication denied environments, a situation that arises frequently in disaster areas, underground exploration (cave or mine), and secure military communication. In such scenarios, a remote robot has a limited communication capability and broad range of communication modalities (satellite, mobile network) are not available. Furthermore, there is a critical safety issue for a robot that is at risk of losing the signals or coverage by other robots in a field full of adversaries.

Relay Robots: To alleviate the difficulties of communication denied scenarios, intermediate communication relays may be established, and these relays can also be mounted on robotic systems. The relay robots cater signals/service to remotely placed units from an operator who stays in a safe location. However, as the number of relays is limited, and the signal degrades, or drops over long distances due to the presence of obstacles and terrain, an optimal placement plan is required to achieve the best communication signal possible. Accordingly, the question of where to place these relays so as to maximize effective communication is one that can be answered through the use of communication aware world mapping.

Communication Aware World Mapping: This tool is proposed in this research in order to deal with the challenge of creating a map of the environment that is directly related to the communication quality. A communication aware world map is a decomposition of the world map based on communication quality that guides the placement procedure of intermediate relay robots in order to maximize the signal strength. The outcome of the process can be either a chain or a spanning tree of relay robots, depending on the number of remote units.

Visibility based Communication and Systems: Visibility is a very important metric for autonomous guarding, patrolling, coverage, and security robots where an area or a target needs to be in the direct Line-of-Sight (LoS) of the robots. Also, in several restricted military missions, two robotic units are only allowed to communicate while they are in the direct LoS of each other. This form of communication is more difficult to intercept or jam, because it requires the attacker to be directly between the sender and receiver. However, mission-critical movements of land forces may naturally cause them to lose LoS with their friendly units. Therefore, we need to solve a number of key challenges to create an efficient relocation method among

the robotic nodes or relays that can re-establish/maintain communications between the units. Some research questions centered on this theme are:

1) Whether a setup of the units and robots form a communication-valid connected network;

2) How the units move and what mobility models they follow;

3) How to relocate a single vehicle in order to recover a unit that went out of sight;

4) How complex the problem is if we select more than one vehicle to relocate;

5) What is the hardness of the problem of replanning the entire setup of the relay vehicles;

6) What is the minimum number of vehicles to maintain visibility with all the units deployed;

7) Can we do a patrolling among the different locations using minimum number of available vehicles in the worst case scenario;

Communication Aware Safe Planning A generic task assigned to an autonomous system is accomplished through a sequence of activities (e.g. Furniture assembly, construction work). Some activities may be performed in parallel while other activities may need to be completed in a sequential manner depending on the precedence constraint. Parallel task requires a number of robots to be engaged at the same time in the system. Moreover, the workers stay in the workplace and therefore we require a safe robotic work schedule that eliminates collisions among the different robotic systems and human workers. For example, in an automated construction project, the human workers and equipment, such as trucks, cranes stay and move together. These jobsites are a source of potential accidents which include a significant loss of lives every year due to struck-by collisions involving moving equipment and workers [OSH]. Recent data shows that the percentage of struck-

by accidents constituted 17.6% of fatalities and serious injuries among construction workers [CPR13]. We found that the real cause of the hazards lies in the planning phase where the workers are not well communicated or informed about potential struck-by risks. Therefore, a communication aware safe planning model is required where alternate construction plans are suggested to the planning managers through computer simulation that calculates their safety, cost, and duration attributes. A key challenge in this research theme is to make an event based simulation framework of the work procedure for the entire project, which enables the project managers to simulate the project in a virtual environment and realize the safety and cost metrics before the actual project takes place.

Communication Aware Movements: The traditional robotic motion planning [LaV06a] algorithms generate a transformation for a robot from one configuration to another by minimizing a cost metric (e.g. distance). However, in a communication denied setup a multi-objective optimal plan is required that will optimize different objectives such as minimizing the traveling duration, enemy exposure, and maximizing communication, profit, and visibility. Objectives can be weighted and converted to a single objective optimization problem, but appropriate weights may not be known a priori. Moreover, the cost functions can be additive, non-additive, cooperative or non-cooperative and therefore significant modifications to the existing motion planning algorithms and the cost functions are required.

1.3 Related Work

Communication map generation: Our first problem of interest of building a communication aware world mapping related to the placements of the relay robots is well motivated by the problems explained in [DD14, DeB10]. According to these

research, the use of autonomous robotic units in military and rescue missions are rapidly increasing to reduce the fatality of human units. These robots are remotely controlled by an operator who stays in a safe region. A solution to this problem was proposed in [BDH$^+$09, BDH$^+$10] where the relays form a chain to provide communication service to a remote unit. A modified Bellman-Ford algorithm [CLRS09] was used on a grid decomposition of the environment to find the shortest sequence of the desired number of grid points, each of which will contain a relay robot. However, one drawback is that the frequent re-computation is required for the same environment when the remote unit moves. We will show that this can be avoided using our proposed communication aware world map.

In the case of multi-unit, multi-relay scenarios the limited branching *Steiner* tree discussed in [WWBB13] can be very helpful in that it will span a minimum spanning tree among the robots. Our proposed methodologies are connected to visibility graph-based [Kir83] planning and art gallery problems [O'R87, O'R04] that guard polygons through visibility. However, we must find a solution using the given number of relays instead of a visibility based shortest path that contains an unrestricted number of intermediate nodes. Two separate groups of researchers presented leader-follower based robot formations in [RS08, RCM04], and [WTM09, BF10, LX05]. In [RS08], the authors used a consensus based scheme while the authors of [WTM09, BF10, LX05] designed a dynamic controller. None of them considered obstacles and therefore no motion based optimality was guaranteed.

Robot Placement in LoS network: A visibility based robot network setup is commonly used in modern military missions for unit formation, area coverage or security systems [WTK11, PTDM12]. Here, in this part of our research, we are mostly interested in monitoring a number of human units or landmarks by a given number of autonomous vehicles through direct Line-of-Sight (LoS). This class of

problems are well motivated in [MVSW12] where multiple vehicles were used for area coverages. Also, the usefulness of connected network in the modern military expedition was discussed briefly in recent researches [WTK11, PTDM12].

Closely related to our work is presented in [OOD12], where the authors proposed a solution to visit all the visibility polygons using a single vehicle. This solution is based on an artificial genetic algorithm and we found that the optimality cannot be easily guaranteed due to random mutation of two different paths. In fact, the solution is not guaranteed to visit a minimal number of regions using an optimal traveling route. In another stream of research, some attempts have been made to maintain visibility to a single static landmark [BMCH07, MMCH05]. Also, the idea of a powerful servicing vehicle serving light mobile units has been explored in *data muling* and *data ferrying* [MAZ+15, BTI11, DCIVR06, TILT09]). These schemes differ slightly from our problem because of focusing more on proximity rather than Line of Sight based communication such as Free-Space Optical Communications (FSOC) [JDH+06].

The analysis of the problems computational complexity about robot patrolling has similarities to the well-known TSP [DM01] problem. We could also relate the problem to the solutions of the set-cover [Mit00] problem, where a set of regions can cover all the units. In computational geometry, the *Watchman Route* problem [Mit13] has a strong connection to our ideas. However, the solutions of traditional watchman route problem do not consider the differential constraints of the robots nor the visibility metrics of the solution path.

Communication aware safe planning: Communication aware safe planning research enables us to create a safe plan for an automated project, especially where the human workers and machines coexist in the same workspace. As a generic project we select the construction planning where the human workers and equipment, such

as trucks and cranes stay and move together. A safety oriented project plan can be achieved through simulating different options of action sequences and selecting the optimal one based on project related metric. Many existing research works focus on construction project simulation such as [CT13, AH11, KM01]. However, most of these tools only provide graphical modeling in computer aided systems and are unable to quantify the safety aspect of a construction plan. Also the existing literature cannot answer about an alternate project plan in case the selected plan is not safe.

This research has commonalities to approaches that use Linear Temporal Logic [BKV10], STRIPS-like representations [GNT04] that connect with motion planning algorithms [CA09]. These systems converts a high level plan to low level trajectories. However, we found that the *Activity Graphs* blended with the Discrete Event Simulations (DEVS) [Zei84] models are more efficient than other methods, which generate a number of alternate plans and simulate them in detail using low level motion planning methods. In some research works [ZAH10] and [ZHB11], traditional motion planning algorithms such as Rapidly Exploring Random Tree (RRT) was used for re-planning of crane motion in real time. However, these tools [ZAH10, ZHB11, KL90] are not intended to capture the whole project and fail during detailed low level simulation.

Communication Aware Multi-Objective Motion Planning: As described earlier, we may need to optimize more than one objective during path planning for a robot compared to the traditional robotic systems where a single objective is optimized. A common approach to this problems is found in literature based on scalarization of objectives, where the objectives are weighted and added to form a single scalar value [Tar07a]. However, appropriate weights are difficult to compute and lots of tuning is needed before achieving an acceptable value.

We propose a solution by modifying the RRT* [KWP+11b, KF11] algorithm which is an optimized version of the standard Rapidly Exploring Random Tree (RRT) [LK99] that generates a single path optimizing all of the objectives. We incorporate the multi-criteria optimization problem by normalizing the objectives using the *Utopian* optimal vector [ZL07] during the RRT* tree expansion process. One stream of research work also uses RRT* in [YGS15] in order to adopt multiple criteria during expansion. However, they generate a number of Pareto optimal paths and do not suggest a way to select a single one of them.

Another stream of research [Fuj96] prioritizes one objective over other and the resulting path is naturally biased towards the high priority objectives. In the sampling based pursuit evasion scheme [KF10a], multiple RRT* [KF11] trees were used, one for each unit, and the evader's tree was expanded in a restricted way to avoid pursuers. We also extend this idea and apply to our modified RRT* tree algorithm to avoid enemy units while maximizing communication and minimizing path length.

1.4 Thesis Organization and Contribution

The thesis consists of six chapters that solve the different robotic problems in communication denied environments.

Chapter 2: In chapter 2, we investigate a remote controlled mission where a base operator controls a number of remotely placed units through several intermediate relay robot vehicles. We develop algorithmic solutions for estimating the best locations to place the available relay robots in order to maximize the overall communication quality. Initially we decompose the world model into a grid as we found that the solution on a continuous plane is NP-Hard. Here, two major problems were considered: 1) A chain formation of communication relays building a signal

link from the base station to remote robot; 2) A tree formation that spans over multiple remote robots, relays and base station operator. For the chain formations, a re-usable data structure based on a layered graph is computed that contains the positions of the intermediate relays, and the initial node at level 0 of this structure is the base operator's position. We propose a modified breadth-first search algorithm and apply it on the layered graph to estimate the communication map. This map is used to extract the optimal positions of the relay robots on a communication chain depending on the position of the single remote unit throughout the mission. We show that our solution is able to reduce significant computation time through the elimination of the frequent re-computation of the entire plan each time the remote unit moves to a new position.

In the cases of serving multiple remote units, a limited branching Steiner tree [WWBB13] is computed that essentially optimizes the communication cost. This solution is achieved by building a number of alternate min-arborescence trees [GGST86] and selecting the one that yields the optimal communication cost.

Chapter 3: This chapter contains the problems and sub-problems of visibility based relay network communication systems. We propose motion planning solutions to recover a LoS based network through re-planning and relocating the robot vehicles. Two categories of robots are used here, mobile units and autonomous vehicles, where the former moves freely and independently in the environment. Consequently, the autonomous vehicles chase the units in order to repair any visibility based disconnection. Therefore, we first need to identify any disconnection resulting from the motion of the units. Two algebraic graph theory based algorithms, centralize and distributed, have been proposed and either of these is effectively triggered by any movement in the system to check the system status. The proposed centralized

algorithm uses algebraic graph theoretical methods while the distributed algorithm depends on a message passing protocol.

Afterwards, we propose techniques that can recover visibility based connectivity by relocating a single vehicle based on optimal motion cost. Oftentimes, a single vehicle is not sufficient to reconnect the relay network as this may disrupt the remaining connected part. We therefore extend the solution to relocate multiple vehicles in situations where a single vehicle is unable to repair disconnections. However, the exact solution of calculating new positions for the relays is proven to be NP-hard and an approximated heuristic procedure has been proposed to calculate a possible sub-optimal solution. Additionally, a patrolling scenario may be required in the cases of an insufficient number of vehicles to visit the newly calculated polygons. Further optimization has been achieved in terms of motion cost by utilizing the graph theoretical methods.

Chapter 4: Next, in Chapter 4, we focus on quantifying the safety score for a fully communication aware safe robotic project plan and analyze the alternate plans to select the safest one. We define a project plan as *safe* if, 1) there is no or a minimal chance of collision among the moving robots, and 2) the moving robots avoid the solid obstacles. An automated building construction project has been selected for safety analysis purposes as these environments are naturally very complex and contain lots of motions and obstacles. Our proposed solution aids the project managers to plan/re-plan a sequence of project related activities in order to reduce the chance of fatalities and injuries during different phases of a project. In addition to collision avoidance, the plans for the robotic project are required to optimize multiple objectives, which leads to Pareto optimality [War87] if there is not a single best plan that minimizes all the objectives, such as cost, duration or safety.

Chapter 5: This chapter presents the multi-objective optimal RRT* [KF11] motion planning algorithm for calculating a Pareto optimal path that best optimizes all of the different objectives. Accordingly, we must guarantee that the selected solution path is in the Pareto optimal set [War87], which is a set of non dominating solutions (i.e. no solution in this set is better that the other members). We mainly modify the tree expansion steps of the well known RRT* sampling based motion planning algorithm so that the RRT* tree expands through satisfying multiple objectives. This has been achieved through the incorporation of a cost vector in place of a single cost function and normalizing the elements of the vector during the tree update process. We also provide a solution for multiple robots that may be cooperative and non-cooperative. In such cases, separate cost functions are designed along with the multi-objective cost vectors that either attract or repel the multiple tree nodes (for multiple robots) during the tree expansion phase.

Finally, We evaluate our theories through extensive experiments on realistic computer simulation models (python, MSL Library and Gazebo simulator [KH04]) and outdoor hardware deployment. We also design inexpensive robot vehicles equipped with a number of sensors (GPS, Lidar, ZigBee communication antennas) and onboard computation modules (Raspberry Pi, Arduino). The test-bed is generic and re-programmable for the purposes of adapting them easily in the future robotic experiments.

CHAPTER 2

COMMUNICATION AWARE MAPPING

In this Chapter, we present our relay based communication framework with the modeling of a communication-based world model that we define as communication map. Our proposed communication map is a graph-based data structure that effectively encodes the positions of relay robots depending on the 1) number of remote units to be served and 2) number of available relays. The sole purpose of this work is to maximize communication quality of distant units controlled from a safe base station.

2.1 Relay Based Communication

Relay-based communication has practical applications in scenarios where traditional communication systems are compromised or broken. Such scenarios can be found in disaster areas, military operations, nuclear waste monitoring, underwater exploration, or forest areas where either the traditional communication is absent or a manned mission is not safe. In these communication-constrained environments, one or more unmanned units can be used to collect data, monitor activity, or take other actions. These robots are remotely controlled by an operator who stays in a safe region. However, due to obstacles and terrain the signal degrades or drops over long distances and we need to deploy intermediate relay robots in between the operator and the remote units in order to maximize communication quality. Example scenarios for this problem are shown in Figure 2.1(a), where we need to build a relay chain to serve a single remote unit, and in Figure 2.1(b), where we need to construct a spanning tree to serve three remote units. As the number of relays is limited, an optimal placement plan is required to achieve the best communication signal possible to the remote units using the available relays.

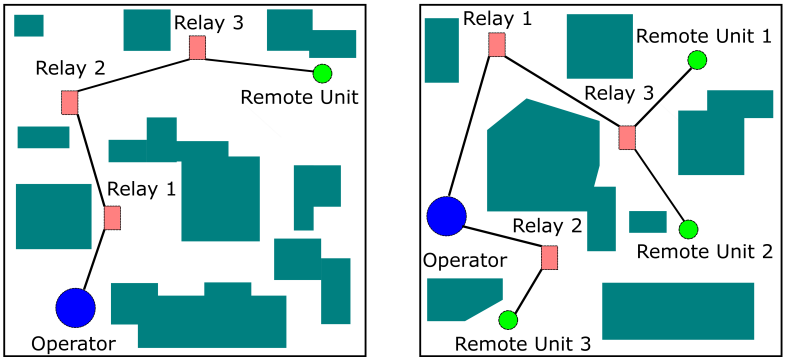


Figure 2.1: (a) A chain consisting of three robots that relay communication from an operator to a remote unit; (b) A minimum spanning tree incorporating three relays, optimizing communication from an operator to three units.

2.2 Related Work

The robotic relay placement problem has been studied in the literature with a focus on controlling the robots and the formation of a relay chain. The best known solution for our first problem about the robotic relay chain formation (Figure 2.1(a)) was proposed in [BDH$^+$09, BDH$^+$10]. Two different algorithms, a modified Bellman-Ford algorithm [CLRS09] and a dual ascent algorithm, were used on a grid to find the shortest sequence of grid points for placing the given number of robot relays. Although their solution is able to form a relay chain, frequent re-computation of the chain is required each time either the unit moves to a new location or the number of relays changes. In contrast, we develop a reusable data structure as a static placement map that is computed once and used to extract the new locations of the available relays when the unit moves throughout the mission. Thus, our solution eliminates significant re-computation and re-planning time in a mobile robotic system.

Our second problem, multi-unit multi-relay tree formation, is connected to the limited branching *Steiner* tree discussed in [WWBB13]. Although the general prob-

lem is known to be NP-Hard, the authors proved that a polynomial time algorithm can compute a tree for a fixed number of branching and terminal nodes. We adapted the ideas for our problem and illustrated them in an algorithmic way that was missing in [WWBB13]. Another relay formation solution using the Markov chain is proposed in [KMadH11], where the relays move based on the inputs from their neighbors. However, obstacles were not considered, and the robots did not form other topologies beyond a chain.

This research is closely related to wireless sensor networks, mesh networks, and multi-hop dynamic wireless networks. A summary of notable works can be found in the survey, [YA01]. However, most of the solutions are related to area coverages for which static relay nodes are used. In contrast, we use robotic relay nodes that are capable of adjusting their locations through movement to maintain mutual connectivity. Therefore, a better communication quality can be achieved with fewer nodes compared to area-based sensor mesh networks.

Our ideas are naturally connected to visibility graph-based [Kir83] planning and art gallery problems [O'R87, O'R04] that guard polygons through visibility. However, the solution is a minimum number of nodes required to observe the whole galley, which is not applicable in our problem where we need to achieve the best communication using the given number of nodes. Similarly, visibility graph based approaches focus on finding shortest visibility paths and cannot limit the number of intermediate nodes.

In [LOC16], a particle swarm optimization is used on an initially connected network to change the travel direction of the relays. This method, however, cannot repair a disconnection, nor initialize an entire setup. Their approach also needs fine tuning of different weights which may introduce additional complexity.

In [OZLL14], the authors proposed a relay UAV motion relative to an access point based on the perceived signal to noise ratio (SNR). This solution works on a single relay and single unit problem, and cannot be extended to multi-relay coverage problems. Similarly, an energy-minimization solution is proposed in [CKS14], where a static operator can communicate with a static unit through a UAV. However, no algorithmic solution is provided and the model cannot be adapted for moving targets.

Our work also has similarities to the leader-follower robot formation where a number of robots position themselves according to the policy distributed by their leader. A number of related solutions are presented in [RS08, RCM04], and [LX05]. Although a consensus-based control algorithm is provided in [RS08] and a dynamic controller was designed in [LX05], no obstacles are considered in either work. A visual odometry is used in [RCM04] to keep the leader in sight, but the calculated trajectories and positions do not guarantee any optimality.

Also closely related to our work are the ideas described in [BF10, WTM09], where a number of relay routers adapt their locations based on that of a moving unit. Although an initial implementation was provided for motion tracking, the optimality of the relay placement is not guaranteed. Furthermore, the methodology was not implemented for obstacle avoidance during the relay robot motion, which will make the problem significantly harder.

2.3 Mathematical Formulation

We will consider a two-dimensional *environment* $\mathcal{W} = \mathbb{R}^2$ that is filled with polygonal *obstacles* $\mathcal{O}$ as illustrated in Figure 2.1. In this environment, there is a set of m *relay vehicles* $A_1, A_2, \ldots, A_m$ and p remote units $B_1, B_2, \ldots, B_p$ that need to be connected to a static *operator* S. We define the collision-free space as $\mathcal{W}' = \mathcal{W} \setminus \mathcal{O}$

where the units and relays present in the world can move freely. The remote units are modeled as point robots and a unit B_j has configuration space $\mathcal{B}_j$, where a particular configuration $r_j \in \mathcal{B}_j$ is defined as $r_j = (x, y) \in \mathcal{W}'$. Similarly, the configuration space for the operator S is defined as $\mathcal{S}$, where an operator's position $s \in \mathcal{S}$ is denoted by $s = (x, y)$. The relay vehicles are modeled as car-like robots and incorporate differential constraints on their movements. A particular vehicle A_i has a configuration space $\mathcal{C}_i$ and the positions $q_i \in \mathcal{C}_i$ are defined as $q_i = (x, y, \theta) \in \mathcal{W}' \times [0, 2\pi)$ [LaV06b]. Vehicle dynamics for A_i are defined as $\dot{x}_i = u_s^i \cos\theta$, $\dot{y}_i = u_s^i \sin\theta$, and $\dot{\theta}_i = \frac{u_s^i}{L^i} \tan u_\phi^i$; [BL89b], where u_s^i is the forward speed and u_ϕ^i is the steering angle of the vehicle. In most parts of the chapter (except motion planning), we will consider $q_i = (x, y)$ as our approach calculates the positions of relays rather than planning their trajectories. We define the entire system state space to be $X = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_m \times \mathcal{B}_1 \times \mathcal{B}_2 \times \cdots \times \mathcal{B}_p$. Let $X_{obs} = \{x \in X : x \cap O \neq \emptyset \text{ where } O \in \mathcal{O}\}$ be the obstacle state space. The collision-free state space is then $X_{free} = X \setminus X_{obs}$.

2.3.1 Communication Quality

As the communication to the units must be established through the relay vehicles to and from the operator, their placements will affect the communication quality. The communication quality can be interrupted or degraded due to: a) the distance between two components, and b) the presence of obstacles that directly affect the communication quality [BMPC08]. For any two points on the plane $\rho_1, \rho_2 \in \mathcal{W}'$ that have free Line of Sight (LoS), the path loss [BMPC08, SAZ08] is proportional to the quadratic distance, $d^2(\rho_1, \rho_2)$, and we define the free space path loss function $f_F : \mathcal{W}' \times \mathcal{W}' \to \mathbb{R}^{\geq 0}$ as:

$$f_F(\rho_1, \rho_2) = \begin{cases} \alpha d^2(\rho_1, \rho_2) & \text{if } d(\rho_1, \rho_2) < d^{th} \\ \infty & \text{otherwise} \end{cases} \tag{2.1}$$

Here α is the loss coefficient and d^{th} is the distance threshold beyond which no communication can be established. Let the path loss in the presence of obstacles and terrain be $f_{\mathcal{O}}(\rho_1, \rho_2, \mathcal{O})$, which includes the costs resulting from diffraction (f_{DF}), fading (f_{FA}), and/or multipath propagation [SAZ08].

$$f_{\mathcal{O}}(\rho_1, \rho_2, \mathcal{O}) = \begin{cases} 0 & \text{if } \overline{\rho_1\rho_2} \text{ has LoS} \\ f_{DF}(\mathcal{O}) + f_{FA}(\mathcal{O}) & \text{otherwise} \end{cases} \tag{2.2}$$

Diffraction loss of a signal results from an obstacle in between the transmitter and receiver that scatters the signal by the edges of the obstacle, and *Fading* occurs when the obstacles reflect the signal, causing multiple routes of reception [SAZ08]. Here, for simulation purposes, we use a simple weighted obstacle crossing based on the amount of intersection of the line segment $\overline{\rho_1\rho_2}$ with the obstacles and $f_{\mathcal{O}}(\rho_1, \rho_2, \mathcal{O}) = \gamma \cdot |\overline{\rho_1\rho_2} \cap \mathcal{O}|$, where γ is a weighting coefficient.

Finally, the total communication cost f_C between ρ_1 and ρ_2 is defined as:

$$f_C(\rho_1, \rho_2) = f_F(\rho_1, \rho_2) + f_{\mathcal{O}}(\rho_1, \rho_2, \mathcal{O}) \tag{2.3}$$

2.3.2 Relay Placement Problems

Our first problem of interest is to develop a solution for the relay placement problem involving an operator, a number of relay robots, and a remote unit. Given the operator's position s and a remote unit's position r, we need to calculate a set of relay robots' positions $q_1, q_2, \ldots, q_m$ such that they form a communication chain. The operator and the remote unit are the two endpoints to complete the chain and the communication cost is,

$$f_C^L = f_C(s, q_1) + \sum_{1 \leq i < m} f_C(q_i, q_{i+1}) + \cdots + f_C(q_m, r) \tag{2.4}$$

We are required to solve the problem of creating a reusable placement map that gives the best placements for a given number of relay vehicles and different positions of a remote unit. Therefore, we define a communication map corresponding to the static operator s as $M_c^s : \mathcal{B} \to \mathcal{C}^n$. Computation of a chain formation of multiple relay robots describes our first problem, and we define the MULTI-RELAY CHAIN problem as:

Problem 1: MULTI-RELAY CHAIN - Finding Optimal Positioning of a Set of Relay Robots on a Chain.

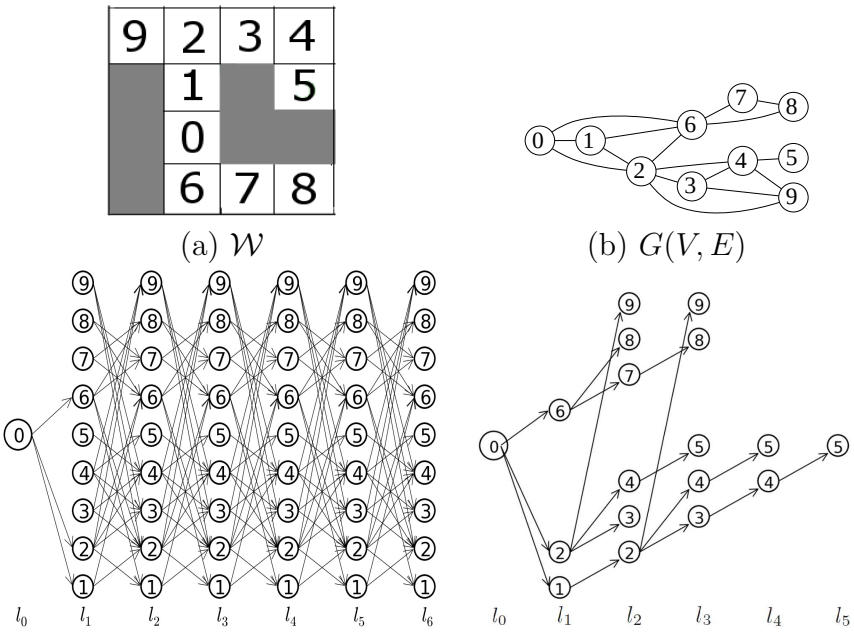
Given the fixed positions r and s corresponding to a unit B and an operator S, find m points $q_1, \ldots, q_m$ corresponding to the m relay vehicles $A_1, A_2, \ldots, A_m$ in the free space that form an $m+1$-link m hop path to connect s to r and minimize f_C^L.

We extend the multiple-relay single-unit problem to a multiple-relay multiple-unit problem. Consequently, we have p unit positions $r_1, \ldots, r_p$ that must be connected to s through m relays. Therefore we define our second problem as a MULTI-RELAY MULTI-UNIT problem:

Problem 2: MULTI-RELAY MULTI-UNIT - Finding Optimal Positioning of a Set of Relay Robots That Serve a Number of Remote Units.

Given a set of fixed positions $r_1, \ldots, r_p$ of p units and the position s of one operator, compute the optimal positions $q_1, q_2, \ldots, q_m$ of m relay robots on the plane that form a connected component among the operator, relays and units while the term $\sum_{1 \leq i \leq p} \min_{1 \leq j \leq m} f_C(r_i, q_j) + \min_{1 \leq j \leq m} f_C(s, q_j)$ is minimized.

In this case, the optimal solution is a tree $T = (V, E)$ that spans over the operator, p remote unit positions, and m available relay positions. Accordingly, the



(a) $\mathcal{W}$ (b) $G(V, E)$

(c) $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (d) M_c^0

Figure 2.2: (a) A sample environment with obstacles decomposed into a grid; (b) Connected communication graph G with the weights in f_C; (c) Directed layered graph $\mathcal{G}$ generated from G and (d) Communication map M_c^0 as a form of a shortest path tree excluding irrelevant nodes of $\mathcal{G}$.

communication cost of this multi-unit system is defined as:

$$f_C^T(T) = \sum_{(u,v)\in E} f_C(u, v) \tag{2.5}$$

We need to compute a tree T that minimizes the communication cost $f_C^T(T)$.

2.4 Single Unit Multiple Relay Placement

A MULTI-RELAY CHAIN problem is shown in Figure 2.1(a) where we want to form a relay chain between the operator and the remote unit. However, the problem becomes NP-Hard on a plane filled with obstacles as stated below.

Proposition 2.4.1 *The MULTI-RELAY CHAIN problem in a polygon with holes is NP-Hard.*

Proof. (Sketch) Our problem is similar to the shortest $m+1$-link paths in polygons with holes as discussed in [AMP91, MPA92]. The bi-criteria shortest path decision problem was proven to be NP-Complete [AMP91] when we need to decide if a path with $m+1$ links is the shortest. Therefore the optimization version of calculating the shortest $m+1$ link path (our MULTI-RELAY CHAIN problem) is generally NP-Hard. Although we use communication cost metric f_C, it depends on the distance metric d and does not reduce the hardness of the problem. □

Therefore, we employ discretization as shown in Figure 2.2(a)-(b) instead of solving the problem in the continuous plane. We convert the world $\mathcal{W}' = \mathcal{W} \setminus \mathcal{O}$ into a grid (such as a *Sukharev* grid [YL04]) with n grid points $\Omega = \{g_1, g_2, \ldots, g_n\}$. An example environment grid Ω is shown in Figure 2.2(a) where the operator S stays in cell 0. A graph representation $G(V, E)$ of Ω, based on communication cost f_C, is drawn in Figure 2.2(b) where the node set V is composed of all the grid points that are not inside the obstacles $\mathcal{O}$, and is defined as $V = \{v_i | v_i \equiv g_i \in \Omega \text{ and } g_i \notin \mathcal{O}\}$. Here, a node $v_i \in V$ is equivalent to a grid point $g_i \in \Omega$, but contains additional attributes such as identifier, cost, neighbors, and parent. Each node $v \in V$ has a unique identifier $v.id$ that is used to identify the node. The set of undirected edges E is defined as $E = \{(u, v) : f_C(u, v) < \infty\}$. Here, the communication between two grid points is blocked by the obstacles that we enforce for demonstration purposes. However, we will show other general cases in the experimental section where the signal is allowed to penetrate the obstacles.

Next, we compute the communication map M_c^s using Algorithm 1. A layered directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $m+2$ levels $l_0, l_1, \ldots, l_{m+1}$ for m available relay robots is computed (see Figure 2.2(c)) based on the original graph G. Level l_0 contains only one node $v_s^0 \equiv v_s \in V$ corresponding to the static operator's position s which also represents the root of the tree. Each of the subsequent layers l_i, where $1 \leq i \leq m+1$,

will copy all the nodes $V \setminus v_s$ of the original graph G. This means a particular layer l_i contains the nodes $\mathcal{V}_i = \{v_1^i, v_2^i, \ldots, v_{|V|}^i\}$ and, for a node $v_k^i \in V_i$, the identifier $v_k^i.id = v_k.id$, where $v_k \in V$ is the corresponding original node in G. Additionally, the nodes at different layers with the same index have the same identifier, which means $v_k^1.id = v_k^2.id = \cdots = v_k^{m+1}.id$ (see Figure 2.2(c)). Finally, the node set $\mathcal{V}$ for the graph $\mathcal{G}$ is defined as,

$$\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_{m+1} \tag{2.6}$$

which contains $O((m+1) \cdot |V|)$ nodes. A directed edge $(u, v) \in \mathcal{E}$ is allowed only between the nodes of any two consecutive layers l_i and l_{i+1} (lexicographic order) if and only if $(u', v') \in E \Leftrightarrow f_C(u, v) < \infty$ where $u.id = u'.id$ and $v.id = v'.id$ for $u', v' \in G.V$:

$$\mathcal{E} = \{(u, v) : u \in l_i, v \in l_{i+1} \text{ and } f_C(u, v) < \infty \ ; \ 0 \leq i \leq m\} \tag{2.7}$$

Once the layered graph $\mathcal{G}$ is constructed, we compute a modified shortest path tree that results in our communication map M_c^s. The resulting tree is constructed by exploring $\mathcal{G}$ layer-wise in a lexicographic order while removing the unnecessary nodes that have already attained optimality. Therefore, we modify the breadth first graph search (BFS) [CLRS09] algorithm to explore layer by layer and compute the shortest chain from the root v_s to each of the nodes. Line 3 of Algorithm 1 initializes the exploration by enqueuing v_s into a queue Q. In order to compute the shortest path tree, we introduce a hash table h of length $|V|$ that uses $v.id$ as the keys and is initialized to ∞ (line 4). We defined earlier that a particular node $v \in \mathcal{V}$ has the same key $v.id$ in all the layers of $\mathcal{G}$ where its instances appear (see the numbering in Figure 2.2(c)). Therefore, h is used to keep track of the lowest cost of each node $v \in V$ of the original graph G as we explore throughout the levels of $\mathcal{G}$.

Algorithm 1 multiRelaySingleUnit($G(V, E)$)

```
1: G(V, E) = calculateGraph(G)
2: v_s.cost = 0, and v.parent = NULL; ∀v ∈ V
3: Enqueue(Q, v_s)
4: h[v.id] = ∞; ∀v ∈ G.V
5: while Q ≠ ∅ do
6:    u = Dequeue(Q)
7:    for v ∈ u.Neighbors do
8:       if u.cost + f_C(u, v) < h[v.id] then
9:          v.parent = u
10:         v.cost = u.cost + f_C(u, v)
11:         h[v.id] = v.cost
12:         Enqueue(Q, v)
13:      end if
14:   end for
15: end while
```

Although the identifiers ($v.id$) of a node's replicas across all layers are identical, their cost attributes $v.cost$ differ at different layers. Initially, the cost of the root node $v_s.cost = 0$ and the parents of all the nodes are set to $NULL$ (line 2), as many nodes have multiple incoming edges. Our target is to select one incoming edge per node in order to choose a parent. We dequeue a node u from Q (line 6) and check to see if setting it as the parent of its neighbors in the next layer will reduce their costs. Accordingly, in lines 7-14 we select node $u \in \mathcal{V}$ as the parent of a node $v \in \mathcal{V}$ if the condition $u.cost + f_C(u, v) < h[v.id]$ is satisfied. Otherwise, $u.cost + f_C(u, v) \geq h[v.id]$ indicates that we already have achieved the optimal cost in one of the prior layers, including the current layer, with a better parent than u. For example, in Figure 2.2(d) node 2 achieves the optimal cost $h[v_2.id] = 2$ (using (2.4)) at layer l_2 through the node 1 of layer l_1. During the evaluation of node 2's replica in layer l_3, we do not find any node u that satisfies $u.cost + f_C(u, v_2) < h[v_2.id]$, thus it is excluded from the tree, having no incoming edge. Finally, we achieve a communication map M_c^s, as shown in Figure 2.2(d), after traversing all the nodes.

Chain Extraction: Given the position of a mobile unit r, and a number of relay robots m, we search for $v \in \mathcal{V}$ s.t. $v.x = r.x$ and $v.y = r.y$ in the $(m+1)$-th layer of the communication map M_c^s. If such a node is found we backtrack recursively using its parent pointer until the root v_s is reached at layer l_0. The nodes found along this traversal are the positions of the intermediate relays. However, if r is not found in layer $(m+1)$, we search for it in layer m, then layer $(m-1)$ and so on, until we find r or reach layer l_1. If r is found in a lower layer $l_{m'}$ where $m' \leq m$, we can achieve a minimum cost using $m'-1$ relay nodes. On the other hand, if we reach layer l_1 in this process, then the position r cannot be served by m relay robots. The grid points that cannot be served by m relays compose the *shadow region* $\Phi^m \subset \Omega$ of $\mathcal{W}$:

$$\Phi^m = \{g \in \Omega | g \equiv v \notin l_i \text{ where } 1 \leq i \leq m+1\} \tag{2.8}$$

In Figure 2.2(a), grid point 5 cannot be served by $m = 1$ relay and therefore does not appear in the layers l_1 and l_2 of M_c^s in Figure 2.2(d).

The above chain extraction procedure implies that M_c^s only needs to be constructed once for an environment $\mathcal{W}$ if the operator does not change its position s. Then, the positions of any number of relays can be extracted to serve a unit located anywhere on the grid.

Algorithm analysis: The running time of Algorithm 1 is $O(\mathcal{V}+\mathcal{E})$ as every node and edge is visited once [CLRS09]. However, the input is a graph G of n nodes from which we computed $\mathcal{G}$ with $(m+1)(n-1)+1$ nodes for $m+2$ layers. In the worst case, where every node can communicate to all other nodes, the total number of edges is at most $|\mathcal{E}|$ = (number of edges in $m+1$ layers) + (number of edges in layer l_0) $= m(n-1)(n-2)+n-1 = O(mn^2)$, which is also the running time of Algorithm 1.

2.5 Multiple Unit Multiple Relay Placement

According to the definition of the MULTI-RELAY MULTI-UNIT problem (Problem 2), there are m relays available for serving p mobile units that are located at $r_1, r_2, \ldots, r_p$. We need to compute the optimal locations $q_1, q_2, \ldots, q_m$ that will connect the operator position s to the units. However, the general problem on a plane becomes NP-Hard.

Proposition 2.5.1 *The MULTI-RELAY MULTI-UNIT SERVING problem in a polygon with holes is NP-Hard.*

Proof. (Sketch) A Euclidean m-median problem is to find a set of m points on a plane to serve p fixed nodes so as to minimize $\sum_{1 \leq i \leq p} \min_{1 \leq j \leq m} d(r_i, q_j)$. This is shown as NP-Hard in [MS84] and [FMW00] for polygons with holes. Our MULTI-RELAY MULTI-UNIT problem is similar except that the $m + p + 1$ points need to form a connected component, and therefore cannot be relaxed to an easier version. Thus, according to the technique of proof by restriction [GJ79], MULTI-RELAY MULTI-UNIT contains the Euclidean m-median problem and is therefore NP-Hard. □

Consequently, we use the same discretization method of relay chain placement similar to that shown in Figure 2.2(a). We need to compute a minimum spanning sub-tree of $G(V, E)$ (Figure 2.2(b)) that spans over all the p unit locations, m relays, and the operator such that the units become the leaf nodes while all the relays become the internal nodes. The problem of interest has commonalities to the limited branching *Steiner* tree discussed in [WWBB13] where the authors prove that a polynomial time algorithm exists for a fixed number of branching and terminal nodes (m intermediate and p terminals in our case). However, we must prevent the remote units from branching and must make the operator the root.

Algorithm 2 computes the solution for the optimal multi-relay positioning for multiple units. Let the set of $p+1$ fixed nodes be $V_T = \{v_s\} \cup V_B$, where $v_s \in V$ is the operator node and $V_B \subset V$ is the set of nodes corresponding to the remote units. As we have n nodes in $G(V, E)$ (from n grid points), including the $p+1$ fixed nodes, we have to select m relay locations from the remaining $n-p-1$ nodes. Therefore, we define $\vartheta_m \subset \mathcal{P}(V \setminus V_T)$ as the set of all possible sets of nodes with exactly m members. Here, $\mathcal{P}(V \setminus V_T)$ is the power set of the remaining nodes other than the fixed nodes. Accordingly, ϑ_m has $\binom{n-p-1}{m}$ members that are used to enumerate $\binom{n-p-1}{m}$ possible graphs, each of which has exactly m relays, p units, and one operator.

Algorithm 2 multiRelayMultiUnit($G(V, E)$)

1: $V_T = \{v_s\} \cup V_B$
2: $\vartheta_m = \{\nu \in \mathcal{P}(V \setminus V_T) : |\vartheta| = m\}$
3: **for** $\nu_i \in \vartheta_m$ **do**
4: $\quad V_i = \nu_i \cup V_T$
5: $\quad G_i = computeDiGraph(V_i)$
6: $\quad$ **if** $G_i.connected()$ **then**
7: $\quad\quad T_i = minArborescence(G_i)$
8: $\quad\quad \mathcal{T}.add(T_i)$
9: $\quad$ **end if**
10: **end for**
11: *return* failure if $\mathcal{T} = Null$
12: *return* $\underset{T_i \in \mathcal{T}}{\text{argmin}} \, [f_C^T(T_i)]$

From lines 3-10 of Algorithm 2, we compute a set of $\binom{n-p-1}{m}$ spanning trees, $\mathcal{T}$, and select the optimal one. For each set $\nu_i \in \vartheta_m$ of m nodes, we construct a directed sub-graph $G_i(V_i, E_i)$ from the undirected graph $G(V, E)$, where $V_i \subset V$ and $V_i = \nu_i \cup V_T$ (in total, $m+p+1$ nodes). For each undirected edge $(u, v) \in E$, if $u, v \notin V_B$, then the edge is replaced with two directed edges. Otherwise, if $u \in V_B$, then the edge is replaced with only one directed edge from v to u, or vice versa (see

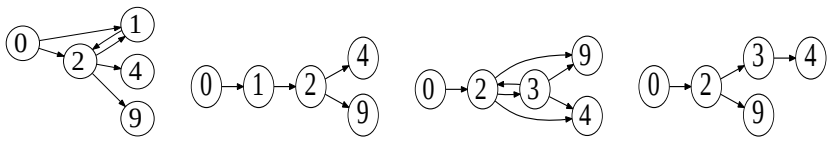


Figure 2.3: The operator is at cell 0 and two units ($p = 2$) are placed at cells 4 and 9 that need to be served by $m = 2$ available relays: (a) A sub-graph G_1 constructed with $\nu_1 = \{v_1, v_2\}$; (b) Resulting min-arborescence tree T_1 of G_1; (c) Another candidate sub-graph G_2 with $\nu_1 = \{v_2, v_3\}$; and (d) Candidate tree T_2

Figure 2.3(a) and (c)). Also, the operator node (root) v_s has no incoming edges.

$$E_i = \{(u, v) \in E : u \notin V_B \text{ and } v \neq v_s \text{ where } u, v \in V_i\} \tag{2.9}$$

Once we construct a graph G_i, we check its connectivity and exclude it from further computation if it is not connected. Otherwise, on the graph G_i that has exactly m relays, p units, and one operator, we compute the minimum spanning tree T_i which is generally called the min-*arborescence* tree [GGST86] for directed graphs. We apply Tarjan's algorithm [GGST86] to get a minimum arborescence tree T_i (see Figures 2.3(b) and (d)). Finally, we choose the tree that yields the minimum cost:

$$\underset{T_i \in \mathcal{T}}{\operatorname{argmin}} f_C^T(T_i) \text{ where } f_C^T(T_i) = \sum_{(u,v) \in T_i} f_c(u, v) \tag{2.10}$$

Algorithm analysis: The running time of Algorithm 2 depends on lines 3-10. The *loop* of line 3 runs $\binom{n-p-1}{m}$ times, which can be simplified as $\frac{(n-p-1)^{\underline{m}}}{m!} = \frac{(n-p-1)(n-p-2)...(n-p-1-m)}{m(m-1)...2 \cdot 1} = O(n^m)$ for a constant m. As the Tarjan's algorithms runs in $O(E + V \log V)$ [GGST86], in the worst case it's running time is $O(E) = O(m + p + m(m-1) + mp) = O(m^2 + mp + p)$ (by sub-graph construction as shown in Figures 2.3(a) and (c)). Therefore, the running time of Algorithm 2 is

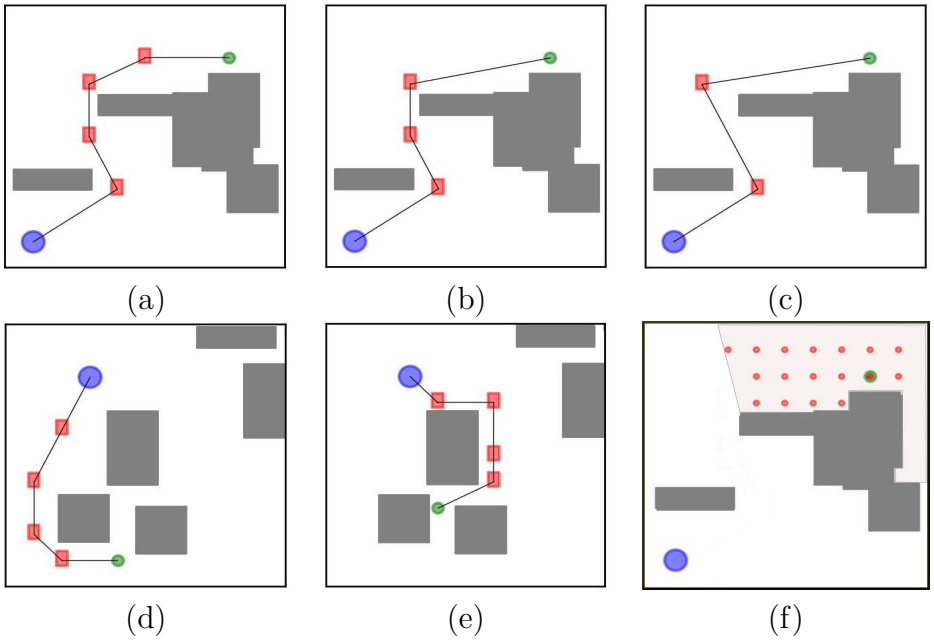


Figure 2.4: Multi relay chain simulation: (a) Four relays forming a chain; (b) and (c) Number of relays are reduced to three and two, respectively; (d) and (e), the remote unit relocates to a new position, triggering the reformation of the relays; (f) Shadow region Φ^1 for one relay (using (2.8)).

$O(n^m(m^2 + mp + p))$. Generally, for a robotic mission, the given number of relays, m is fixed which makes the running time polynomial.

Extension to Aerial Systems: Algorithms 1 and 2 can easily be extended to a 3D environment. In such cases, the environment would be in the form of a cuboid that can be decomposed into a 3D grid onto which our algorithms can be applied.

2.6 Experimental Results

We tested our proposed methodologies through software simulations and hardware experiments. A detailed analysis will be presented in this section.

2.6.1 Software Simulation

Multi-Relay Chain: We have implemented Algorithm 1 on several randomly generated environments shown in Figures 2.4 and 2.5. Once we generate the communication map M_c^s, we extract the chain based on the number of available relay robots. Figure 2.4(a) is a solution to a visibility based system when we have 4 intermediate relays and no connection is allowed through the obstacles $\mathcal{O}$. In this case we search and find the unit node in layer l_5 of the communication map M_c^s and backtrack until we reach the operator node in layer l_0. Consequently, the four intermediate relay positions are extracted from the layers $l_4,\ l_3,\ l_2$ and l_1 of M_c^s. Next, we reduce the number of relays to 3 and then 2, and the solutions extracted from the same M_c^s are shown in Figure 2.4(b) and (c), respectively. In all of the cases, our algorithm extracted solutions form the same map M_c^s and are able to minimize the distances of the successive nodes in the chains.

Figures 2.4(d) and (e) demonstrate the reusability of our communication map M_c^s when the unit moves from place to place. Here, we have four relays and we are able to find the unit node in layer l_5 of M_c^s for both the cases. Finally, Figure 2.4(f) shows a case where a single relay cannot serve the unit which stays inside the shadow region Φ^1 (as per (2.8)).

In our next case study, we allow the signal to be penetrated through the obstacles $\mathcal{O}$ (like radio waves). The path loss is therefore impacted by fading (f_{FA}) and diffraction (f_{DF}) effects (as per (2.2)). Accordingly, Figures 2.5(a), (b) and (c) show the optimal relay placements for four, three, and one available relays, respectively. In all three cases, the algorithm found the minimal obstacle intersections to minimize the communication cost f_C^L. In Figures 2.5(d), (e) and (f), we allow the operator to stay at a safe place inside a building, and compute the solutions for one, two and

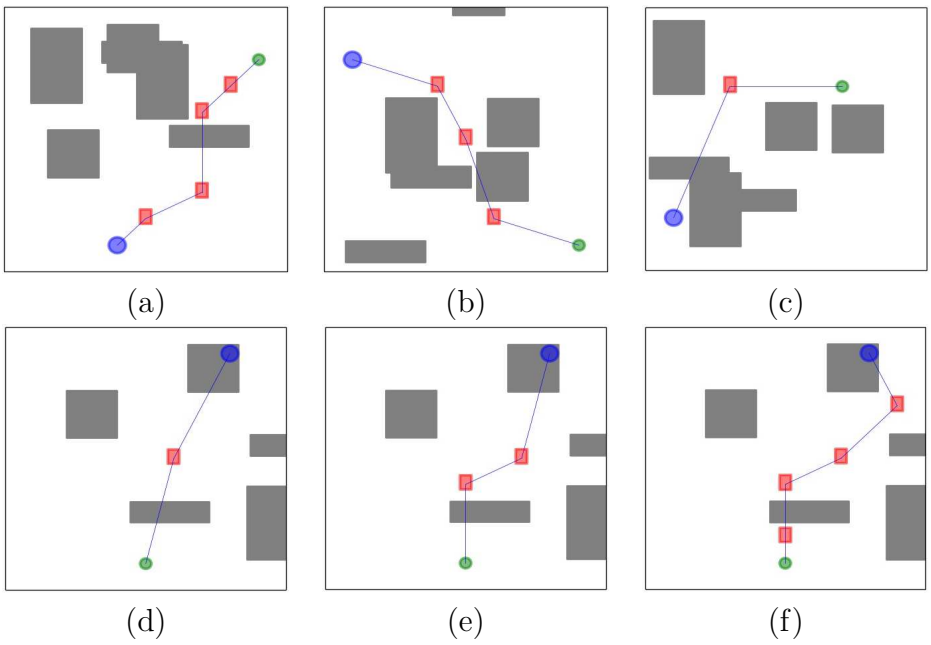


Figure 2.5: Communication can now be established through the obstacles with extra costs according to (2.2): (a) Four relays, (b) three relays and (c) one relay connecting the unit to the operator. (d) The operator stays inside a building and one relay is available; (e) and (f), Obstacle crossings have decreased as the number of relays has been increased to two and four, respectively.

four available relays. We see that the obstacle intersections are decreasing with the increasing number of relays, and the signal gets stronger, which is expected.

Multi-Relay Multi-Unit Tree: Next we simulate the cases involving multiple units that are collectively served by multiple relays. Six sample min-arborescence trees are shown in Figure 2.6 as generated by Algorithm 2. The operator does not directly serve the units, which means the units receive their service from one of the relays. Figures 2.6(a) and (b) demonstrate the cases of two vehicles A_1, and A_2 relaying communication to four units B_1, B_2, B_3 and B_4. Next, we increase the number of relays to three and four; the outputs are shown in Figures 2.6(c), (d) and Figures 2.6(e), (f), respectively.

2.6.2 Hardware Experiment

To demonstrate our chained-relay, and multi-relay multi-unit solutions, we use robots based on the open source SERB robot [BMG+12] in a hardware/software test-bed. The experiments are performed on an indoor small prototype test-bed which is considered as a miniature of the larger real world systems. The vehicles are equipped with a battery, two servo motors and an Arduino Uno that operates the motors and sensors. We added line detection sensors for use in navigation along the environment's grid lines.

The test-bed's software components are distributed between a C++ program that runs on the Arduino to operate the vehicles, and a Python program that runs on a separate controller device to implement the two presented algorithms. The centralized communication between the controller and the robots is implemented using XBee DigiMesh 2.4GHz wireless radio transceivers.

Motion Planning: We used the A^* search algorithm [LaV06b] to generate trajectories that avoid all obstacles. During one robot's trajectory generation, all other robots, the operator, and the units are considered obstacles, in addition to static obstacles $\mathcal{O}$. The coordination method [LH98b] also works well for multi robot path planning. In cases where a car-like robot is used, the optimal RRT* [KF11] path planning in $3m$ dimensional space ((x, y, θ) for each relay) for m robots can be implemented. However, the relocation cost (e.g. fuel consumption, navigation costs for large military vehicles) may outweigh the new communication cost, if the communication quality improvement is marginal. Therefore, the operator will decide either to move the relays or to stay with the current setup considering the communication cost calculated by our system (f_C^L or f_C^T) and the traveling cost computed by a selected motion planner.

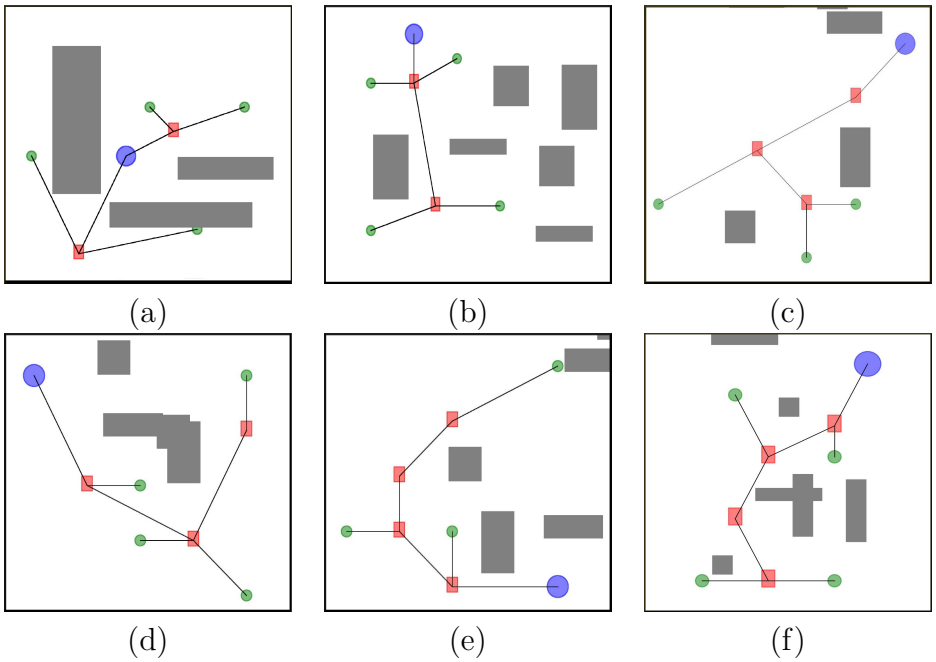


Figure 2.6: Multi-Relay Multi-Unit simulations. (a) and (b) show min-arborescence tree for two relays serving four units; (c) shows three relays serving three units and (d) is a case of three relays connecting four units; (e) and (f) are min-arborescence tree for four relays connecting the units to the operator.

MULTI-RELAY CHAIN: In Figure 2.7 we demonstrate the relay chain formation using two available relay robots A_1 and A_2. The unit B is located at the top-left corner while the operator S stays near the red obstacle (Figure 2.7(a)). The two marked locations are the desired positions for the relays that were calculated using Algorithm 1. In Figure 2.7(b), robot A_1 starts moving following the path generated by the A^* algorithm, avoiding all the obstacles and other robots. A_1 reaches its destination and A_2 prepares to move as shown in Figure 2.7(c). Finally A_2 reaches its goal location as shown in Figure 2.7(d), establishing a relay chain (yellow dotted lines).

MULTI-RELAY MULTI-UNIT: An example environment with two relays (placed at top right) and two remote units is shown in Figure 2.8. The relays need to

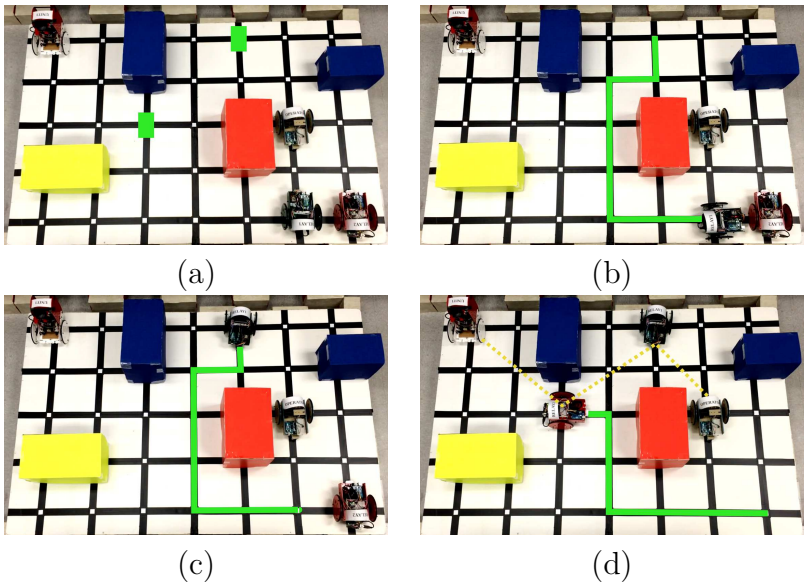


Figure 2.7: Multi-relay chain experiment: (a) A_1 and A_2 need to be on the two marked positions generated by Algorithm 1; (b) A_1's path generated by the A^* algorithm; (c) A_1 reaches its destination and A_2 prepares to move; (d) A relay chain is established: $S \rightarrow A_1 \rightarrow A_2 \rightarrow B$.

move to the marked locations (Figure 2.8(a)) that are the part of a min-arborescence tree generated by Algorithm 2. Vehicle A_2 moves along its path as shown in Figure 2.8(b), and reaches its destination in Figure 2.8(c), in which A_1 also starts moving. Finally, in 2.8(d), both vehicles A_1 and A_2 have reached their destinations and an optimal communication tree (green lines) is established.

2.6.3 Numerical Analysis

In Table 2.1, we compare our relay chain model (Algorithm 1) with a closely related solution from [BDH$^+$10] in terms of the increasing number of nodes. For each model, the left column has two components: 1) the time to build the graph + 2) the time to compute the underlying data structure (M_c^s in our case), and the right column shows the time to recompute solutions in response to the changes either in the

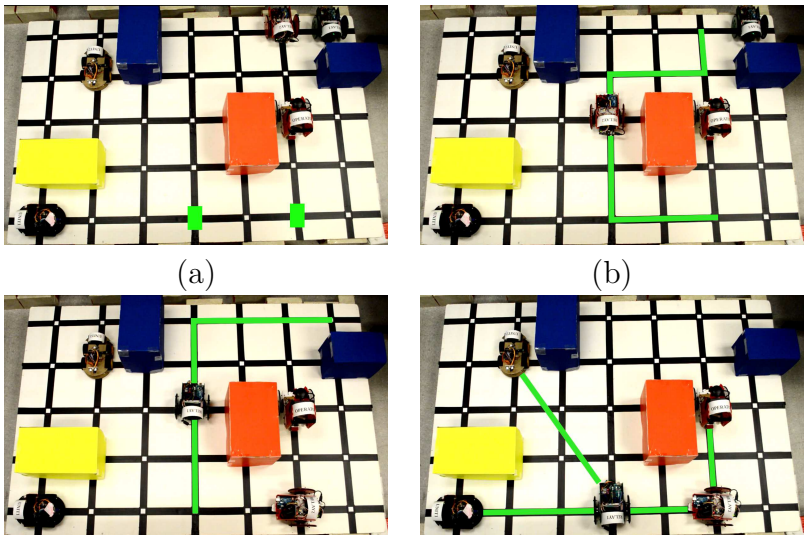
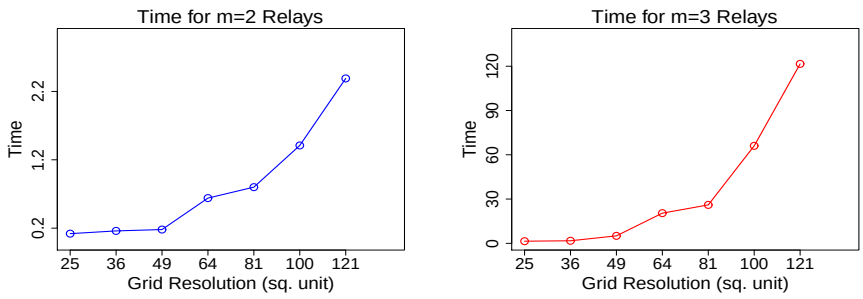


Figure 2.8: Multi-unit multi-relay experiment: (a) A_1 and A_2 need to be on the two marked positions generated by Algorithm 2; (b) A_2 is moving along its path as generated by the A^* algorithm; (c) A_2 reaches its destination and A_1 is moving along its path; and (d) a min-arborescence tree has been formed with the edges $E = \{(v_s, v_2), (v_2, v_1), (v_1, r_1), (v_1, r2)\}$.

number of relays or the location of the unit. Although our graph-building phase takes longer than that of [BDH$^+$10] due to the construction of the layered graph $\mathcal{G}$, computation of reusable map M_c^s is commonly faster for smaller environments as we use a modified BFS algorithm on $\mathcal{G}$ (which is a tree), compared to a modified Bellman-Ford algorithm [CLRS09] used on G according to [BDH$^+$10]. Then, we achieve significant improvements in the subsequent computations than [BDH$^+$10], as we only need to extract a chain of relays from M_c^s instead of recomputing the entire data structure.

We plot the running time of our min-arborescence tree computation (Algorithm 2) with the increasingly large environments in Figure 2.9. The curves correspond to polynomial running times in terms of a fixed number of relays m.



(a) (b)

Figure 2.9: Running time plotted against environment size for (a) two available relays ($m = 2$) and (b) three available relays ($m = 3$).

Table 2.1: Analysis of Running Time (in seconds)

Nodes	Our Method		Burdakov et el.	
	Building $\mathcal{G}$+ M_c^s computation	Subsequent Runs	Building G + k-hop BF	Subsequent Runs
361	6.70+0.438	0.0052	2.39+1.23	1.05
400	8.15+0.58	0.0067	2.66+1.62	1.50
625	23.82+2.41	0.0081	6.54+4.12	4.06
729	35.57+3.70	0.0079	11.39+7.78	7.23
900	49.86+8.01	0.0095	13.52+8.93	8.85
1089	85.01+14.07	0.012	23.03+14.51	14.87

2.7 Discussion and Extension

We have studied the complexities of optimal relay placement problems in an environment filled with obstacles, and proposed solutions that are capable of dealing with most variations of the problems. In the case where we have multiple relay robots, we build a static map which is a reusable data structure computed from a layered graph using the modified breadth-first search algorithm. Thus a chain formation can be obtained for m available relays and a single unit in different positions. This eliminates a significant amount of re-computation in scenarios where the unit relocates, the number of relay changes in the same environment. We also developed a solution for optimal placement of multiple relays in order to serve multiple units.

The solution is a tree and we generate a number of alternate min-*arborescence* trees from which we select the optimal one in terms of communication cost.

One immediate extension of our work is to test the solutions for different communication modalities and perform a benchmark analysis. We are aware of the running time of the multi-unit problem where all possible combinations of candidate nodes may take a long time in the case of many relays. However, this can be improved by early decomposition of the environment and weeding out the unnecessary nodes.

CHAPTER 3

COMMUNICATION BASED ON LINE-OF-SIGHT

In the several specific environments, the communication medium may be constrained and unlike the previous chapter, the signal may not be allowed to pass through the obstacles and terrain. These systems may use visible light communication, human gestures, smoke that require the sender and receiver to be in the direct Line-of-Sight (LoS) of each other. Also besides communication, visibility plays a vital role in many robotic systems such as coverage and patrolling. Therefore, in this chapter, we investigate different scenarios to preserve a LoS based system where the deployed robotic nodes must ensure mutual visibility. We analyze the computational complexity of this class of problems and propose different techniques that focus on setup/recovery of a relay network. We evaluate our theories through extensive experiments on a realistic computer simulation model (Gazebo simulator [KH04]) and through an outdoor experiment where a vehicle equipped with a number of sensors (GPS, camera, ZigBee communication antennas) and onboard computation modules (Raspberry Pi, Arduino) is able to monitor 2 distinct units. The methodologies of this chapter has been partially published previously and can be found in [RBRa].

3.1 Visibility Based Communication

Communication between mobile units located in geographically separated positions in an environment plays an important role in unmanned aerial or ground missions. However, the existing signal can be easily interrupted by natural features such as terrain, atmospheric effects, and electromagnetic interference. Intentional jamming of communications and sniffing by an enemy may also pose a serious risk. In order to mitigate these problems, Line-of-Sight (LoS) communication can be established. This form of communication is more difficult to intercept or jam because it requires

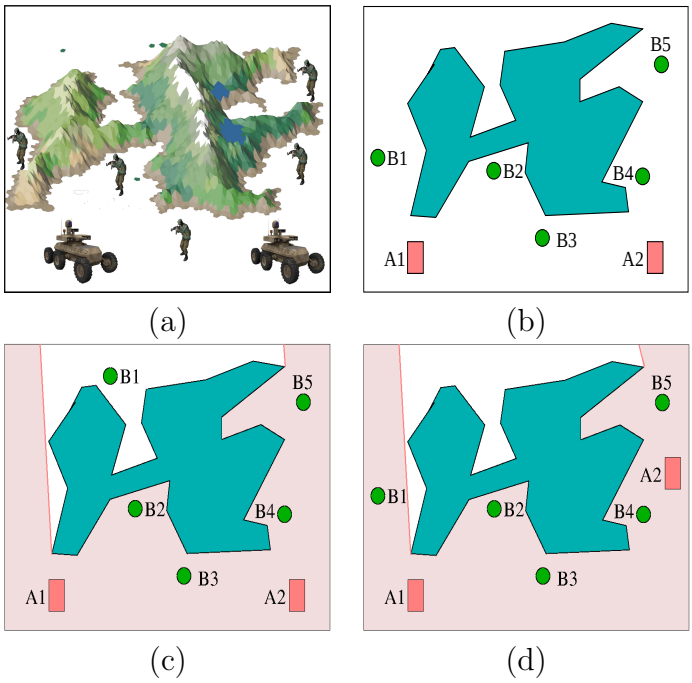


Figure 3.1: (a) A sample field mission where two autonomous servicing vehicles and five units are deployed; (b) The corresponding environment geometry in 2D space. The red rectangles are vehicles while the green circles are mobile units. The polygonal hole in the middle represents the obstacles and terrain $\mathcal{O}$; (c) A communication-invalid state where the unit B_1 is not seen by any of the vehicles; (d) Another communication-invalid state as vehicles A_1 and A_2 do not have any relay communication.

the attacker to be directly between the sender and receiver. Because mission-related movements of land forces may naturally cause them to lose LoS with their friendly units, it is desirable to provide additional nodes or relays that can maintain communications between the units. A group of autonomous ground vehicles can fulfill this role, by moving from place to place as needed for the purposes of establishing relayed contact. One such environment is exemplified in Figure 3.1(a)-(b) where a LoS-based relay network has been established among the autonomous vehicles (rectangles) and units (circles).

Once a connected group of autonomous ground vehicles has been established in the field, its computational and storage capacity can be used to provide services to the units that it serves. This can provide additional military value, by analyzing tactical data, detecting threat patterns, or searching for information that would not otherwise be readily available. The task of a servicing ground vehicle is to maintain LoS to one or more units and at least one other servicing vehicle if more than one vehicle is deployed. Collectively they need to cover all the units and also to maintain a relay network among themselves.

As the units move around the environment, they frequently get isolated from the network as shown in Figure 3.1(c). This problem requires one or more autonomous vehicles to relocate into the visibility polygon of the disconnected unit in order to stay in the LoS and provide service. The relocation may damage the existing LoS-based connectivity of the network as shown in Figure 3.1(d) where the two vehicles become disconnected. Also, in some cases we may not have enough vehicles to form a fixed relay network.

3.2 Related Work

Since the group of *servicing vehicles* must have LoS communication with all the units they serve, our ideas are naturally connected to *Art Gallery* problems [O'R87, O'R04] and other visibility-based approaches in computational geometry [Kir83]. Art-gallery based approaches have been used in robotics to solve sensor [GBL01] and landmark placement [EL11] problems. Another computational geometry problem that is connected to our ideas is the *Watchman Route* problem [Mit13]. Some of the differences between the traditional Watchman Route problem and our setup are: 1) We are not only concerned about the shortest path but also about a path that will

keep the most visibility with all units and other vehicles; and 2) The vehicle's paths should respect differential constraints.

In [OOD12], the authors proposed a scheme to visit all the visibility polygons using a single vehicle, which leads to redundancy when polygons intersect. Also, they do not consider formation of a constrained relay network among *multiple* robots, which is the main goal of this work. The solution is based on a genetic algorithm, and the patrolling route computed for a single robot is unable to guarantee the optimality due to random mutation. In contrast, here we propose a solution that minimizes the number of regions to visit by a single vehicle that either eliminates or reduces the patrolling route.

Closely related to our problem are *visibility-based pursuit* schemes whose goal is to find a path that will guarantee that an evader is captured regardless of his motion [GLL$^+$97]. Our work is also closely connected to path planning approaches that attempt to maintain visibility to a single static landmark [BMCH07, MMCH05].

The idea of a powerful mobile unit uploading, downloading, and distributing data to a set of dynamic units has been explored in *data muling* and *data ferrying* [MAZ$^+$15, BTI11, DCIVR06, TILT09]). One important difference between our formulation and the data muling approach is that communication is based on Line-of-Sight instead of the proximity of the sensor nodes. In the area of communication, Free-Space Optical Communications (FSOC) [JDH$^+$06] is being considered as an alternative for military network-centric operations. Particularly related is the work in [KY14] where the problem of two mobile nodes that try to maintain LoS alignment is studied.

3.3 Preliminaries

Let $A_1, A_2, \ldots, A_n$ be a set of n servicing *vehicles*, with configuration spaces $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ and a set of m mobile *units* be, $B_1, B_2, \ldots, B_m$. Units and vehicles are deployed in a 2D world $\mathcal{W} = \mathbb{R}^2$ which we assume to be a connected polygon. Let $\mathcal{O}$ be the set of obstacles that block communication and are modeled as polygons. The collision-free space is defined as $E = \mathcal{W} \setminus \mathcal{O}$. The mobile units can move freely in the world and are modeled as point robots without rotation. Accordingly, the configuration for a mobile unit B_j is defined as, $r_j = (x, y) \in E$, and the set $\mathcal{B} = E^m$ is the configuration space for the mobile units. The servicing vehicles can move inside the bounded environment E using both translation and rotation actions, and are modeled as point robots with an orientation and configuration defined as $q_i = (x, y, \theta) \in E \times [0, 2\pi)$ [LaV06b]. These vehicles are car-like, and a given vehicle A_i must satisfy differential constraints and dynamics defined as $\dot{x}_i = u_s^i \cos\theta; \dot{y}_i = u_s^i \sin\theta$, and $\dot{\theta}_i = \frac{u_s^i}{L^i} \tan u_\phi^i$; [IKH11], where u_s^i is the forward speed and u_ϕ^i is the steering angle of the vehicle. Together, the n vehicles compose the configuration space $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n$. We define the entire system state space to be $X = \mathcal{C} \times \mathcal{B}$. Let $X_{obs} = \{x \in X : x \cap O \neq \emptyset \; where \; O \in \mathcal{O}\}$ be the obstacle state space. The collision-free state space is then $X_{free} = X \setminus X_{obs}$.

3.4 Problem Statement

3.4.1 Communication State Validity

Communication can only be established among servicing vehicles and between servicing vehicles and mobile units through LoS. Mobile units cannot communicate

with each other. The communication range can be characterized by a *visibility polygon*. The visibility polygon $V(p)$ for a point $p \in E$ is defined as [EGA81]:

$$V(p) = \{w | w \in E \text{ and } \overline{pw} \cap E = \overline{pw}\}. \tag{3.1}$$

Definition 3.4.1 *A state $x \in X$ is considered* communication-valid *if and only if each unit is visible by at least one servicing vehicle and the servicing vehicles form a connected network. We define this set of configurations as $X_{comm} \subset X$.*

According to the definition, we must satisfy the following conditions in order to have a communication-valid state:

$$\forall j, \exists i \text{ s.t. } r_j \in V(q_i) \text{ for } 1 \leq j \leq m \text{ and } 1 \leq i \leq n \tag{3.2}$$

$$\{(q_j, q_k) | q_k \in V(q_j) \text{ for } 1 \leq j, k \leq n,\ k \neq j\} \equiv CC(x) \tag{3.3}$$

where $CC(x)$ is a *connected component* formed by all the vehicle-vehicle connections.

The state $x \in X$ is changed whenever a unit or vehicle changes its configuration. The units move autonomously, and in response to those movements we may need to plan the trajectories and new configurations for the servicing vehicles depending on communication-validity. Therefore, we have a decision problem in which we want to know whether a given state is *communication-valid* ($x \in X_{comm}$). This problem can be formulated as follows:

Problem 1: Communication State Validation

Given the workspace $\mathcal{W}$, a set of obstacles $\mathcal{O}$, a set of configurations $\mathcal{C}$ for servicing vehicles, and $\mathcal{B}$ for mobile units, determine whether a state $x \in X_{comm}$ or not.

3.4.2 Invalid-to-Valid Communication State Restoration

Initially, we assume that the visibility-based network is connected as shown in Figure 3.1 and $x \in X_{comm}$. Since the mobile units are allowed to move freely throughout the environment E, the system becomes *communication-invalid* frequently. A unit is marked as disconnected if and only if it is not visible to any of the servicing vehicles. A set of disconnected units, $D \subseteq \{B_1, B_2, \ldots, B_m\}$, is defined based on a given state $x \in X$, and we dispatch an available vehicle A_i from its current location x_s^i to a newly computed goal region X_G^i in order to reconnect the strayed units. As the event of vehicle relocation must not break the existing partially connected network, the selection of vehicles to be moved must be done carefully. This motivates the following problem:

Problem 2: Communication Validity Restoration

Given $\mathcal{W}$ and $\mathcal{O}$, the current state $x \in X$, and a set of disconnected units D, select one or a number of vehicles to relocate and compute their new goal regions, X_G, that will reconnect all the units in D.

3.4.3 Patrolling and Trajectory Estimation

There may be situations where there are not enough vehicles to serve all of the units and maintain a connected relay network. In these cases, we need to calculate the optimal regions on the free space so that placing the available vehicles on those areas can serve as many units as possible. Furthermore, a patrolling tour may be required by one vehicle which will connect the remaining disconnected units and vehicles and act like a dynamic relay link. Because the vehicle designated to serve the disconnected units and other vehicles along the patrolling route may lose its existing connection to one or more units or vehicles that it is already servicing, the tour

must be chosen optimally, not arbitrarily, such that the traveling time is minimized. Therefore, if we have a set of disconnected units, D, a tour, $\tau : [0, T] \rightarrow X_{free}$, must satisfy,

$$\forall i \exists t \text{ s.t. } r_i \in V(\tau(t)) \text{ where } 1 \leq i \leq m, t \in [0, T]. \tag{3.4}$$

Problem 3: Patrolling Trajectory Estimation

Given $\mathcal{W}$ and $\mathcal{O}$, the current state $x \in X$ and a set of disconnected units D, compute the optimal patrolling trajectory $\tau : [0, T] \rightarrow X_{free}$, such that τ touches the minimum number of discrete regions.

3.5 Communication State Validation

We propose two algorithms: centralized and distributed to estimate whether the current state $x \in X$ is communication-valid, which solves *Problem 1* demonstrated in Section 3.4.

3.5.1 Centralized Algorithm

Initially, we are given the positions of units and vehicles and we do not know which vehicle is providing service to which unit. In order to solve *Problem 1*, we propose Algorithm 3 that works based on graph theoretic network connectivity [ZEP11, SJK08] solutions. A visibility-based graph can be constructed where the node set is composed of all the components (vehicles and units) and an edge is added between two nodes if the corresponding components are visible to each other. However, checking the algebraic connectivity [ZEP11] on this graph is not sufficient. For example, the graph shown in Figure 3.2(b) is connected but not communication-valid. This type of graph occurs if there are obstacles between vehicles. Therefore our proposed

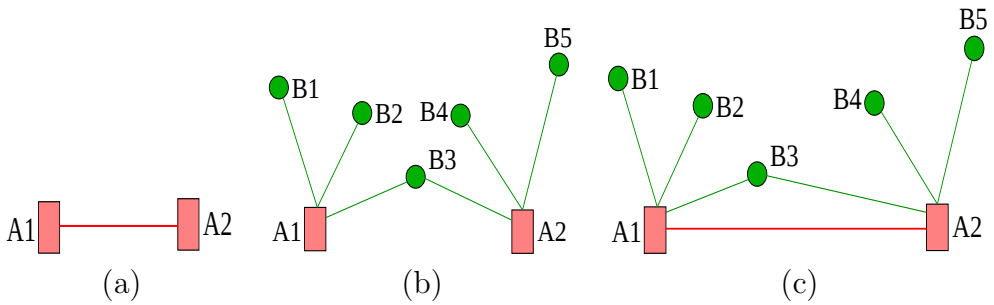


Figure 3.2: (a) Vehicle relay graph $\mathcal{G}_{\mathcal{A}}$ generated from a vehicle-vehicle relay network for the environment demonstrated in Figure 3.1; (b) Unit graph $\mathcal{G}_{\mathcal{B}}$ from the vehicle-unit connectivity; (c) Union of the two graphs, $\mathcal{G} = \mathcal{G}_{\mathcal{A}} \cup \mathcal{G}_{\mathcal{B}}$.

validation is two-fold and we generate the following two types of undirected graphs based on a particular state x.

Vehicle Relay Graph ($\mathcal{G}_{\mathcal{A}}$): This state-dependent undirected graph is a mapping $g_{\mathcal{A}} : X \to \mathcal{G}_{\mathcal{A}}(\mathcal{V}_{\mathcal{A}}, \mathcal{E}_{\mathcal{A}})$, where $\mathcal{V}_{\mathcal{A}} = \{A_1, A_2, \ldots, A_n\}$ is the set of vehicle nodes (see Figure 3.2(a)). $\mathcal{E}_{\mathcal{A}}$ denotes the set of edges defined as,

$$\mathcal{E}_{\mathcal{A}} = \{e_{ij} | q_i \in V(q_j)\} \tag{3.5}$$

where q_i and q_j are the positions of vehicles A_i and A_j in the environment E. This implies that an edge e_{ij} exists if and only if the vehicle A_i is inside the visibility polygon, $V(q_j)$, of vehicle A_j. We then compute the $n \times n$ Laplacian matrix, $\mathcal{L}(\mathcal{G}_{\mathcal{A}}) = DEG(G_{\mathcal{A}}) - ADJ(\mathcal{G}_{\mathcal{A}})$, where $ADG(\mathcal{G}_{\mathcal{A}})$ is the familiar $(0, 1)$ adjacency matrix, and $DEG(\mathcal{G}_{\mathcal{A}})$ is the diagonal matrix of vertex degrees [Mer94], also called the valency matrix of $\mathcal{G}_{\mathcal{A}}$. The entries of $\mathcal{L}$ are as follows [BB10]:

$$i)\ l_{ij} = \begin{cases} -1 & \text{if an edge exists between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$ii)\ l_{ii} = -\sum_{k=1, k\neq i}^{n} l_{ik}$$

In line 2 of Algorithm 3, we check the second-smallest eigenvalue $\lambda_2(\mathcal{L}(\mathcal{G}_\mathcal{A}))$ of $\mathcal{G}_\mathcal{A}$ to see whether it is positive. A non-positive value indicates that the relay network formed by all the vehicles does not exist and the network is communication-invalid (line 3). If $\lambda_2 > 0$, then we go to the second step of validation where we check the entire network connectivity (lines $5 - 11$).

Unit Graph ($\mathcal{G}_\mathcal{B}$): The unit graph $\mathcal{G}_\mathcal{B}$, which is also undirected and is a mapping $g_\mathcal{B} : X \to \mathcal{G}_\mathcal{B}(\mathcal{V}_\mathcal{B}, \mathcal{E}_\mathcal{B})$, is computed in line 5. In contrast to $G_\mathcal{A}$, the graph $\mathcal{G}_\mathcal{B}$ includes all the m units and n vehicles in its node set $\mathcal{V}_\mathcal{B}$. Accordingly, $\mathcal{V}_\mathcal{B} = \{A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m\}$ is indexed by the vehicles, followed by the units, so that all units have indices greater than n. The edge set $\mathcal{E}_\mathcal{B}$ has the following form:

$$\mathcal{E}_\mathcal{B} = \{e_{ij} | r_j \in V(q_i) \text{ where } n < j \leq n + m \text{ and } 0 < i \leq n\}. \tag{3.6}$$

This means that an edge is added if and only if a unit's position r_j is visible from some vehicle's position q_i (see Figure 3.2(b)).

Finally, we form a state-dependent graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as shown in Figure 3.2(c), which is the union of the two graphs $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$. Accordingly, the vertex set $\mathcal{V} = \mathcal{V}_\mathcal{A} \cup \mathcal{V}_\mathcal{B}$ and the edge set $\mathcal{E} = \mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}$. Therefore we conclude that the graph is communication-valid if the second-smallest eigenvalue λ_2 of the $(m + n) \times (m + n)$ Laplacian matrix, $\mathcal{L}(\mathcal{G})$, of graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is greater than *zero* (lines $7 - 11$).

Analysis of Algorithm 3: The graph creation in lines 1 and 5 uses a visibility polygon computation algorithm to determine the edges of the graphs. Each polygon computation takes $O(n)$ [EGA81] and for n vehicles the running time is $O(n^2)$. The dominant factor, however, is in computing the eigenvalues (lines 2 and 7) which generally takes $O(n^3)$ in the worst case. Therefore the running time of Algorithm 3 is $O(n^3)$.

Algorithm 3 communicationCheck($x, \mathcal{O}$)

1: $\mathcal{G}_{\mathcal{A}} = g_{\mathcal{A}}(x)$
2: **if** $\lambda_2(\mathcal{L}(\mathcal{G}_{\mathcal{A}})) \leq 0$ **then**
3: $\quad return\ false$
4: **end if**
5: $\mathcal{G}_{\mathcal{B}} = g_{\mathcal{B}}(x)$
6: $\mathcal{G} = \mathcal{G}_{\mathcal{A}} \cup \mathcal{G}_{\mathcal{B}}$
7: **if** $\lambda_2(\mathcal{L}(\mathcal{G})) \leq 0$ **then**
8: $\quad return\ false$
9: **else**
10: $\quad return\ true$
11: **end if**

3.5.2 Distributed Algorithm

An improvement can be made over the centralized Algorithm 3 if we use the computational power of all the vehicles in a distributed manner. Algorithm 4 presents pseudo code for the *distributed* communication-validity checking *program* which will run in each of the vehicles. In a *distributed processing* system we rely on message passing through network protocols. Algorithm 4 will be triggered once it receives a request or control message. In Line 1 we collect all the elements visible from the vehicle. Line 2 sends a query message to each of the neighboring vehicles except the requester ($reqV$), to share their coverage information. The program then waits for all the vehicles' response messages. Accordingly, line 4 merges the vehicle's own visibility information with that of its neighbors. If the current vehicle is the *initiator*, lines $5-10$ will check the network to see if all the units and vehicles in service were discovered or not. Otherwise, in lines $11-13$, the resulting status from the current vehicle will be sent back to the requester.

Analysis of Algorithm 4 : All the lines in Algorithm 4 except lines $2, 3$ and 12 run in $O(1)$. Lines 2 and 3 will run at most $O(n)$ if all other $n-1$ vehicles are visible. Therefore on a single vehicle the algorithm takes $O(n)$ time. Similarly if all

Algorithm 4 stableStateDist($C_i, reqV$)

1: $\alpha = h_i(x)$
2: *query* $h_j(x)$ *to all adjacent vehicles,* $A_j \in N \setminus reqV$
3: *wait and receive* $h_j(x)$ *from all neighbor vehicles* A_j
4: $\alpha = \alpha \cup h_1(1) \cup h_2(x) \cdots \cup h_j(x)$
5: **if** A_i *is initiator*() **then**
6: **if** $\alpha == N \cup M$ **then**
7: *return true*
8: **end if**
9: *return false*
10: **end if**
11: **if** *receiver*(A_i) **then**
12: *send* α *to the requester*
13: **end if**

$n-1$ other vehicles are the requester, line 12 will take $O(n)$. The drawback of this algorithm is the *messaging* overhead and *waiting time* for responses. The number of messages being sent can be vast if the graph is *dense*.

3.6 Recovering a Communication-valid State with a Single Vehicle

As units are on the move, this may result in disconnections from their respective servicing vehicles. Here we propose a solution that dispatches a single vehicle in order to reconnect a strayed unit from the visibility-based network. Any movement inside a network triggers Algorithm 5, which identifies any disconnections and relocates a vehicle that best re-establishes a communication-valid state without affecting existing network connections. The set of disconnected units D is defined as:

$$D = \{B_j | \forall i, r_j \notin V(q_i) \text{ where } 1 \leq i \leq n\}. \tag{3.7}$$

Therefore D is the set of units that are not visible to any of the vehicles due to obstacles. In other words, the set of all the units with degree *zero* in the graph $\mathcal{G}_{\mathcal{B}}$

compose the disconnected set D. If there is any such non-visible unit (i,e., $D \neq \emptyset$), we attempt to resolve disconnections for each $B_j \in D$ using Algorithm 5.

Next, we define the set H_i as the set of *hard constrained* units of a vehicle A_i which are only visible from A_i and to which no other vehicles can provide service. A unit is said to be a hard constrained unit if and only if it is visible from only one vehicle. Lines $1-3$ of Algorithm 5 compute H_i for all the vehicles according to the following equation:

$$H_i = \{r_j | r_j \in V(q_i) \text{ and } \forall k \neq i, r_j \notin V(q_k)\}. \tag{3.8}$$

Therefore, the unit nodes (nodes corresponding to the units) with degree *one* in the graph $\mathcal{G}_\mathcal{B}$ are the members of the hard constrained sets. In line 4 we compute the intersecting polygon $V(H_i)$ of all visibility polygons of all members in H_i. Initially the set of candidate vehicles for relocation is $C = \{A_1, A_2, \ldots, A_n\}$. However, we may not be able to relocate all the vehicles in C as this may break the existing connected graph topology $\mathcal{G}_\mathcal{A}$ among the vehicles. Therefore, we check the second-smallest eigenvalue of the Laplacian matrix of a graph generated by removing the corresponding vehicle nodes $A_i \in C$ along with their incident edges from graph $\mathcal{G}_\mathcal{A}$. We remove the nodes from C that make $\lambda_2 \leq 0$ (line 6 of Algorithm 5).

The new goal polygon X_G^i of a candidate vehicle $A_i \in C$ must be inside the visibility polygons of 1) the disconnected unit B_j and 2) at least one other vehicle that is a part of the existing relay network. Moreover, if there is any hard constrained unit and $H_i \neq \emptyset$ then X_G^i must be inside the polygon $V(H_i)$. As the visibility polygons may be concave in an environment filled with obstacles, we may get multiple goal polygons. In such cases, we take the largest one. Therefore we compute X_G^i

for $A_i \in C$ as follows (see line 8):

$$X_G^i = \begin{cases} \max\limits_{A_k \neq A_i, 1 \leq k \leq n} V(r_j) \cap V(q_k); \text{ if } H_i = \emptyset \\ \max\limits_{A_k \neq A_i, 1 \leq k \leq n} V(r_j) \cap V(q_k) \cap V(H_i); \text{ otherwise} \end{cases} \tag{3.9}$$

Once the goal regions for all the candidate vehicles in C are computed, we only retain the vehicles that have nonempty goal regions (line 10). We then select the optimal vehicle A_{r_j} in terms of the motion cost. In brief, the *motionCost*() method in line 14 computes the relocation cost of a vehicle from its current position x_s^i to the computed goal region X_G^i using a motion planning algorithm such as Rapidly-exploring Random Trees Star (RRT*) [KWP+11a].

Algorithm 5 singleMoveComm $(B_j \in D, \mathcal{G}_\mathcal{B}, \mathcal{G}_\mathcal{A})$

1: **for** each vehicle A_i **do**
2: $\quad H_i = computeHardConstrained(\mathcal{G}_\mathcal{B})$
3: **end for**
4: $V(H_i) = \bigcap_{k=1}^{|H_i|} V(r_k \in H_i)$
5: $C = \{A_i | 1 \leq i \leq n\}$
6: $C = C \setminus A_i$ s.t. $\lambda_2(\mathcal{L}(\mathcal{G}_\mathcal{A}(\mathcal{V}_\mathcal{A} \setminus A_i, \mathcal{E}_\mathcal{A} \setminus e_i))) \leq 0$
7: **for** $A_i \in C$ **do**
8: $\quad X_G^i = \begin{cases} \max\limits_{A_k \neq A_i, 1 \leq k \leq n} V(r_j) \cap V(q_k); \text{ if } V(H_i) = \emptyset \\ \max\limits_{A_k \neq A_i, 1 \leq k \leq n} V(r_j) \cap V(q_k) \cap V(H_i); \text{ otherwise} \end{cases}$
9: **end for**
10: $C = C \setminus A_i$ s.t. $X_G^i = \emptyset$
11: **if** $X_G^i = \emptyset$ for all $1 \leq i \leq |C|$ **then**
12: $\quad return\ failure$
13: **end if**
14: $A_{r_j} = \underset{A_i \in C}{\text{argmin}}[motionCost(x_s^i, X_G^i)]$
15: *return success*

3.7 Recovering a Communication-valid State with Multiple Vehicles

3.7.1 Hardness of Relocating Multiple Robots

If *Problem 2* cannot be solved by Algorithm 5, we need to move more than one vehicle. New configurations for more than one vehicle cannot be calculated efficiently as different combinations of vehicle movements are possible. Another important constraint is the number of vehicles; a sufficient number of vehicles may not be available to support all the units. Therefore, we first assume that we have only one vehicle that follows a travelling route τ ([OOD12]) to visit the visibility polygons of the m units. An analysis of the hardness of this problem follows.

Definition 6.0: LoS Communication Problem : Given a set of m visibility polygons each for one unit $B_i \in M$, find the shortest tour $\tau : [0, T] \rightarrow X_{free}$ to visit at least one point in each polygon.

Proposition 3.7.1 *LoS Communication problem is NP-Hard.*

Proof. We will prove the hardness of the problem by polynomially reducing the *Traveling Salesman Problem with Neighbors (TSPN)* [DM01], a well-known NP-hard problem, to our LoS communication problem. Suppose TSPN takes as an input a set of convex polygons, $\Gamma = \{P_1, P_2, \ldots, P_m\}$, and the goal is to find a minimum cost tour that touches at least one point in each of the polygons. The convexity of the polygon does not reduce the difficulty of the problem [DM01].

The reduction algorithm will take as an input Γ from TSPN and will place a unit, B_i, in the centroid of each polygon P_i (see Figure 3.3). This centroid calculation can be done in polynomial time. Therefore, P_i will work as visibility polygon for

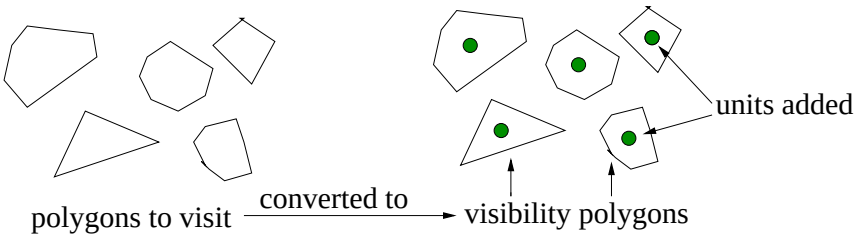


Figure 3.3: An instance of TSPN is reduced to an instance of LoS Communication problem. Each polygon in TSPN will act as visibility polygon of an assigned unit.

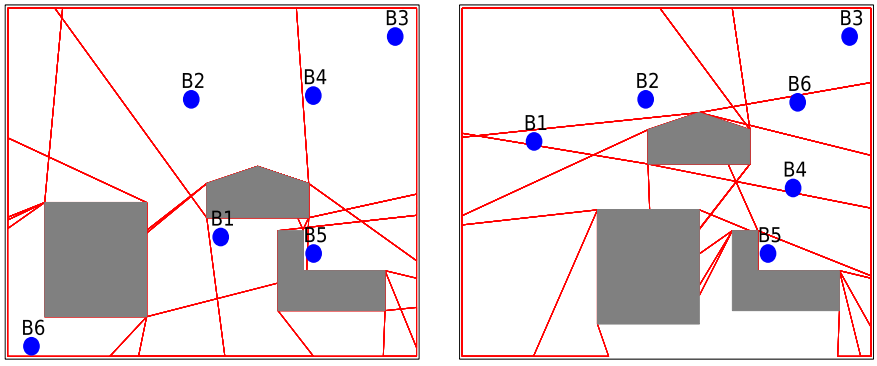
the unit B_i. These will convert the input of TSPN problem to the input of LoS Communication in polynomial time. The output transformation is trivial, since the solution for the LoS Communication problem τ is clearly a solution for the TSPN as each of the units $B_i \in M$ belong to a polygon P_i that will be touched by the tour, solving the TSPN.

Conversely, suppose that we have a solution tour, τ for the TSPN that touches each of the polygons $P_i \in \Gamma$. We can use this as a route that will be followed by the vehicle in our LoS communication problem since the vehicle will go into all the visibility polygons in each tour and will provide service to all the units.

□

3.7.2 Approximated Solution

Since our problem is NP-hard, it cannot be solved exactly in polynomial time unless P=NP. We must use an approximation algorithm for TSPN to get a near optimal solution. As previously mentioned in Section 3.2, a genetic algorithm solution of this problem can be found in [OOD12] for a UAV where the aerial robot is allowed to fly over the obstacles. However, we cannot use this solution or the approximate solution for TSPN due to obstacles in the environment. We must instead use a motion planning algorithm once the sequence of polygons to visit is computed.



(a) (b)

Figure 3.4: Two sample environments are partitioned using visibility polygon based decomposition.

Since visibility polygons may intersect, the number of regions to visit may be fewer than the total number of individual polygons; this allows for a significant improvement over the works in the literature ([OOD12]). Let Γ be the set of goal regions to be visited that are inside the area composed by the visibility polygons of all the units ($\Gamma \subseteq \bigcup_{i=1}^{m} V(B_i)$). There is an intractable number of regions and sub regions in the obstacle free plane E. Therefore, we developed Algorithm 6 which computes Γ, the finite set of goal polygons to be visited by a vehicle. In line 1, we compute the visibility polygons $V(B_1), V(B_2), \ldots, V(B_m)$ corresponding to the units. Next, we decompose the obstacle free environment E into a countable set of polygonal faces $\mathcal{F} = \{P_1, P_2, \ldots, P_\rho\}$ based on the intersections of the visibility polygons as shown in Figure 3.4. We need to select a set of polygons Γ from $\mathcal{F}$, which implies that $\Gamma \subseteq \mathcal{F}$. One important fact is that $\mathcal{F}$ contains all the original polygons and the split polygons resulting from their intersections after the decomposition process. This helps us to select bigger polygonal regions for vehicle placements that cover large areas, making it easier for the motion planner to compute paths, given large goal regions.

The visibility-based decomposition is a vital step towards solving the problem as the edges of a face $P \in \mathcal{F}$ inflict at least one visibility event. Crossing an edge results in the appearance or disappearance of a unit. As a result, each of the polygonal faces P is visible by a set of units. We need to choose the minimum number of such polygons so that they collectively cover all the units. This resembles the well-known geometric *Set Cover* problem [HP11, BG95], and computing the optimal solution is NP-Hard. An instance of such problem consists of a finite set $\mathcal{U} = \{B_1, B_2, \ldots, B_m\}$ and a set of allowable polygons $\mathcal{F} = \{P_1, P_2, \ldots, P_\rho\}$, such that every point in $\mathcal{U}$ is covered by at least one polygon in $\mathcal{F}$ [HP11]. We use a modified greedy set cover approximation algorithm [Cor09, Mit00] to solve this problem (lines 3-11 of algorithm 6).

In line 3 of algorithm 6, we assign a label y_P to a polygon P, which is a set,

$$y_P = \{B_j, B_k, \ldots, B_l\};\ \forall c \in \{j, k, \ldots, l\}\ V(B_c) \cap P_i = P_i. \tag{3.10}$$

This means the label y_P of a polygon P contains the names of the units whose visibility polygons completely enclose P. Next we assign a score to all the polygons as,

$$\hat{s}(P) = \gamma \cdot area(P) + \sum_{B_k \in y_p} \Big[\alpha - \beta \cdot d(P, B_k)\Big] \tag{3.11}$$

Here, $area : \mathcal{F} \rightarrow \mathbb{R}^{\geq 0}$ is used to compute the area of the polygon and $d : \mathcal{F} \times \mathcal{B} \rightarrow \mathbb{R}^{\geq 0}$ is the distance function to compute the distance between any visible unit and the polygon. $\alpha, \beta, \gamma \in \mathbb{R}^{\geq 0}$ are the variables and α is chosen to be very large compared to β and γ to make the visible number of units ($|y_P|$) of the polygon P the most dominant factor of $\hat{s}$. Therefore, it is obvious that the highest scoring polygon covers most of the units. In the cases where more that one polygons cover same number of units, we chose the largest and the nearest one to the units.

Line 7 of Algorithm 6 greedily selects a polygon that covers as many units as possible. Breaking the tie is done using the score $\hat{s}$. We remove the units from $\mathcal{U}$ in line 8 that are covered by P (i.e. in y_P) and add the polygon in the resulting set Γ in line 9. This process ends when $\mathcal{U}$ becomes empty and we terminate the algorithm by returning Γ as the goal regions to visit.

Algorithm 6 multiRobotPlacement $(\mathcal{B}, \mathcal{O})$

1: $\mathbb{V} = \{V(B_1), V(B_2), \ldots, V(B_m)\}$
2: $\mathcal{F} = decompose(\mathbb{V})$
3: $\forall P \in \mathcal{F},\ y_P = assignLabel(\mathbb{V})$
4: $\forall P \in \mathcal{F},\ \hat{s}(t_i) = assignScore(P, y_P)$
5: $\Gamma = \emptyset$; $\mathcal{U} = \{B_1, B_2, \ldots, B_m\}$
6: **while** $\mathcal{U} \neq \emptyset$ **do**
7: Select $P \in \mathcal{F}$ that maximizes $|y_P \cap \mathcal{U}|$
8: $\mathcal{U} = \mathcal{U} - y_P$
9: $\Gamma = \Gamma \cup \{P\}$
10: **end while**
11: *return* Γ

Analysis: Algorithm 6 is composed of two different algorithms. Lines 1-4 calculate a set of polygons $\mathcal{F}$ which is fed as an input to the set cover approximation of lines 5-11. The polygon set $\mathcal{F}$ is produced through the intersection of m visibility polygons which are concave. From a pairwise visibility polygon intersection, we get $O(cm^2)$ polygons for some constant c. These resultant polygons also intersect, yielding $O(c^2m^4)$ polygons. This implies $|\mathcal{F}| = c^2m^4$, which is the input of the set cover approximation that runs in $O(|\mathcal{U}||\mathcal{F}| \min(|\mathcal{U}|, |\mathcal{F}|))$ time [Cor09].

3.7.3 Multi-Robot Placements and Patrolling

In the presence of n vehicles, we have three different cases:

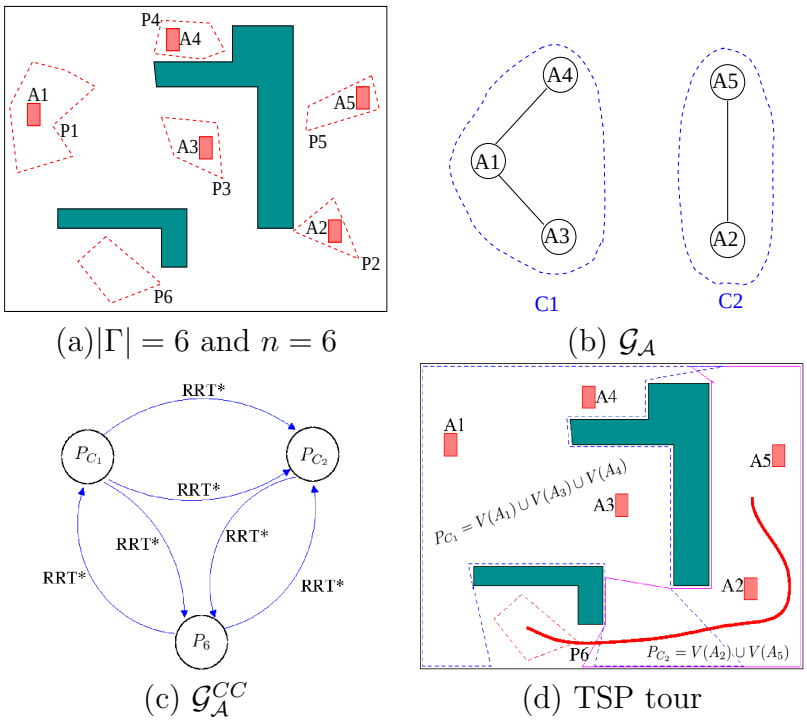
Case $|\Gamma| = 1$: This is a trivial case where we deploy a vehicle in the sole polygon that is visible to all the units.

Case $|\Gamma| < n$: In this case we have enough vehicles that they can be placed as static servers to each of the polygons. We use $|\Gamma|$ vehicles where each of the polygons receives one vehicle. We may need to use one vehicle for patrolling among the polygons depending on the components in the visibility-based vehicle graph $(\mathcal{G}_{\mathcal{A}})$.

Case $|\Gamma| \geq n$ and $|\Gamma| > 1$: In such cases we do not have sufficient vehicles to cover all the polygons. Therefore, we keep assigning one vehicle per polygon in Γ, prioritizing based on their scores, $\hat{s}$, until we are left with one vehicle. The last vehicle will perform a tour, τ, among all the polygons. Therefore the polygons $P_1, P_2, \ldots P_{n-1}$ are covered and all other polygons $P_i \in \Gamma$ s.t. $i \geq n$ are not covered. In both of the above cases, we may need an optimal patrolling strategy in order to establish a dynamic link among the covered polygons (having an assigned vehicles) and uncovered polygons (having no assigned vehicle). Once a number of available vehicles are deployed, as shown in Figure 3.5(a), we compute the vehicle-graph $\mathcal{G}_{\mathcal{A}}(\mathcal{V}_{\mathcal{A}}, \mathcal{E}_{\mathcal{A}})$ as explained earlier in Section 3.5.1 (see Figure 3.5(b)). We then apply the connected component algorithm [HT73] to get a set of subgraph components $C_1, C_2, \ldots, C_\kappa$ where $\mathcal{V}_{\mathcal{A}} = \bigcup_{i=1}^{\kappa} C_i$. By definition, any two member vertices A_j, A_k in a component C_i are connected through a path (visibility path in our case) as shown in Figure 3.5(b). We merge all the visibility polygons of the vehicles under a component C_i to make a single polygon,

$$P_{C_i} = \bigcup_{A_j \in C_i} V(A_j) \tag{3.12}$$

There may be some polygons that are not covered due to insufficient vehicles (if $|\Gamma| \geq n$). These are the polygons denoted as $\Gamma^U = \Gamma \setminus \{P_1, \ldots, P_{|\mathcal{V}_{\mathcal{A}}|}\}$. We thereafter create a directed connected-component graph $\mathcal{G}_{\mathcal{A}}^{CC}(\mathcal{V}_{\mathcal{A}}^{CC}, \mathcal{E}_{\mathcal{A}}^{CC})$, as shown in Figure 3.5(c), where the vertices are composed of the component polygons and uncovered



(a) $|\Gamma| = 6$ and $n = 6$ (b) $\mathcal{G}_{\mathcal{A}}$

(c) $\mathcal{G}_{\mathcal{A}}^{CC}$ (d) TSP tour

Figure 3.5: (a) A set of six polygons, $\Gamma = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ computed by approximate set cover (Algorithm 6) that are to be covered by $n = 6$ available vehicles; (b) Two connected components C_1 and C_2 are computed from vehicle graph $\mathcal{G}_{\mathcal{A}}$; (c) Connected component graph $\mathcal{G}_{\mathcal{A}}^{CC}$; (d) TSP tour and RRT* path to be followed by the 6-th vehicle.

polygons,

$$\mathcal{V}_{\mathcal{A}}^{CC} = \{P_{C_1}, P_{C_2} \ldots, P_{C_\kappa}\} \cup \Gamma^U \tag{3.13}$$

Graph $\mathcal{G}_{\mathcal{A}}^{CC}$ is a complete graph which means any polygon is reachable from any other polygon, as our environment E is connected. The weighted directed edge $e_{ij}^{CC} \in \mathcal{E}_{\mathcal{A}}^{CC}$ between two vertices $P_i^{CC}, P_j^{CC} \in \mathcal{V}_{\mathcal{A}}^{CC}$ is computed using a motion planning algorithm such as RRT*, A* or combinatorial planning [KWP+11a, LaV06b], that finds a path (edge) between the polygons while avoiding the set of obstacles $\mathcal{O}$. Once all the edges are computed, we apply the approximate Geometric TSP [Chr76] algorithm to compute the sequence of polygons to visit. Finally, a motion planner

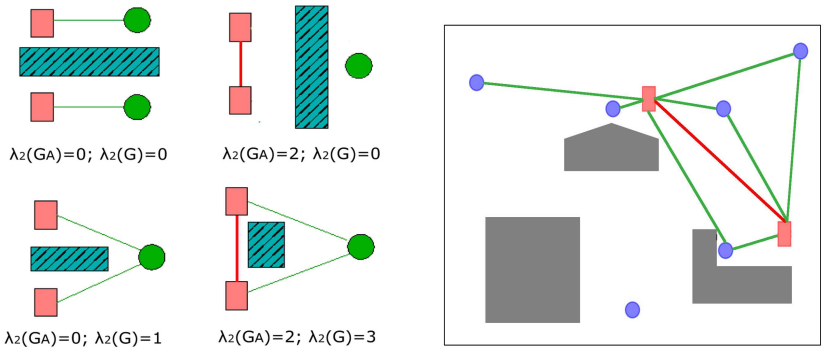
computes a sub-optimal tour that touches all the polygons according to the sequence with minimal motion cost (see Figure 3.5(d)).

3.8 Experimental Results

3.8.1 Checking Communication-Valid State

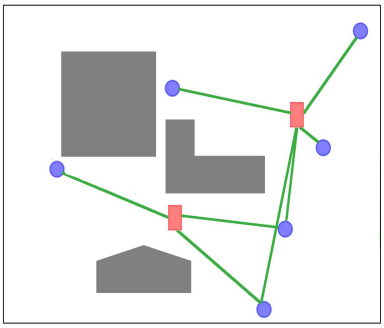
In the first case study, we validate the correctness of Algorithm 3 to check the communication-valid state space. The results from our experiments on different setups of the environment are shown in Figure 3.6. In Figure 3.6(a) we have a few trivial environments where only one graph is communication-valid (bottom right with $\lambda_2(\mathcal{G}_\mathcal{A}) = 2$ and $\lambda_2(\mathcal{G}) = 3$). A complex environment with three obstacles, two vehicles and six units is presented in Figure 3.6(b). Here, the relay network is connected, as the second-smallest eigenvalue of the vehicle graph's Laplacian is $\lambda_2(\mathcal{G}_\mathcal{A}) = 2 > 0$. However, the union graph including vehicles and units results in $\lambda_2(\mathcal{G}) = 0$, which indicates that the setup is not communication-valid.

Another environment is shown in Figure 3.6(c) where the relay network is not communication-valid ($\lambda_2(\mathcal{G}_\mathcal{A}) = 0$), although the union graph is connected. Finally, in Figure 3.6(d) we demonstrate a communication-valid network with three vehicles where both the relay graph and union graph are connected ($\lambda_2(\mathcal{G}_\mathcal{A}) = 1$ and $\lambda_2(G) = 0.5024$).

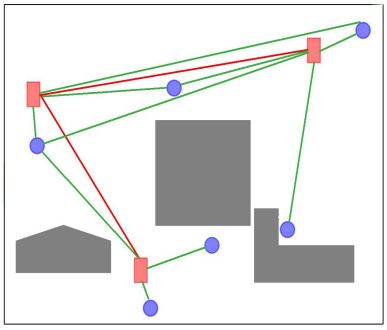


(a)

(b) $\lambda_2(\mathcal{G}_\mathcal{A}) = 2; \lambda_2(\mathcal{G}) = 0.$



(c) $\lambda_2(\mathcal{G}_\mathcal{A}) = 0.$



(d) $\lambda_2(\mathcal{G}_\mathcal{A}) = 1; \lambda_2(\mathcal{G}) = 0.5024.$

Figure 3.6: (a) A few trivial environment setups. Only the bottom right state is communication-valid; (b) and (c) are two communication-invalid states as $\lambda_2 \leq 0$ for at least one graph (relay or union graph) in each environment. (d) A communication-valid state as $\lambda_2(\mathcal{G}_\mathcal{A}) > 0$ and $\lambda_2(\mathcal{G}) > 0$.

3.8.2 Regaining a Communication-valid State by Single Vehicle Movement

We used the Bonnmotion Library [AEGPS10] to generate different mobility models. The CGAL library [FP09] was also used to perform the geometric polygon computation, and the Python programming language was used for visualization. The SMP library [KFb] was used for RRT* algorithm implementation.

In Figure 3.7, we present the test cases for a random waypoint mobility model where $four$ servicing vehicles are assigned to monitor six deployed units. While the units are moving randomly, unit E gets disconnected from the network. Therefore,

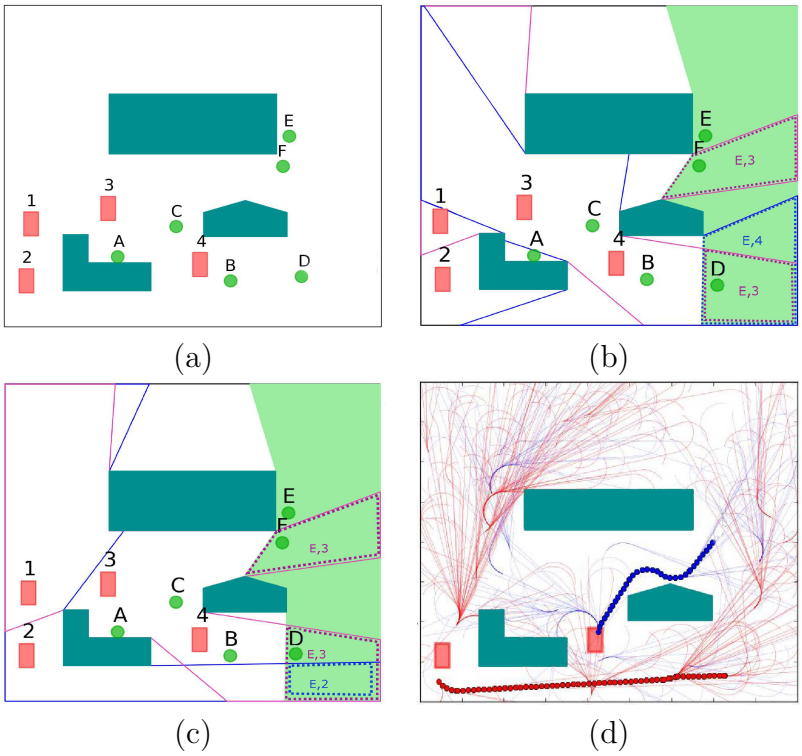


Figure 3.7: Bonnmotion random waypoint experiment: (a) Unit E gets disconnected; (b) Goal region computation for candidate vehicle 2. The green shaded region is the visibility polygon of E. Purple dashed regions are the intersections of vehicle 3 and unit E's visibility polygon while blue dashed area is the intersecting polygon of 4 and E. (c) Goal region for candidate vehicle 4. (d) RRT* trees and resulting trajectories for the two candidate vehicles 2 and 4.

we demonstrate the computation of our proposed Algorithm 5 in Figures 3.7(a)-(d) in order to recover the network. As the relocation of vehicle 1 or 3 would cause other vehicles to become disconnected from the network, they are both eliminated from consideration and the candidate vehicle set becomes $C = \{2, 4\}$. In Figure 3.7(b), we compute the goal region X_G^2 for vehicle 2. We have three regions to consider from the intersection of the visibility polygons marked by dotted lines. From among those, we select the region labeled as "$E, 4$" as X_G^2, which is the largest of the three. Similarly, we compute the region "$E, 3$" as the goal region X_G^4 for vehicle 4 shown in Figure 3.7(c). Finally, as shown in Figure 3.7(d), we

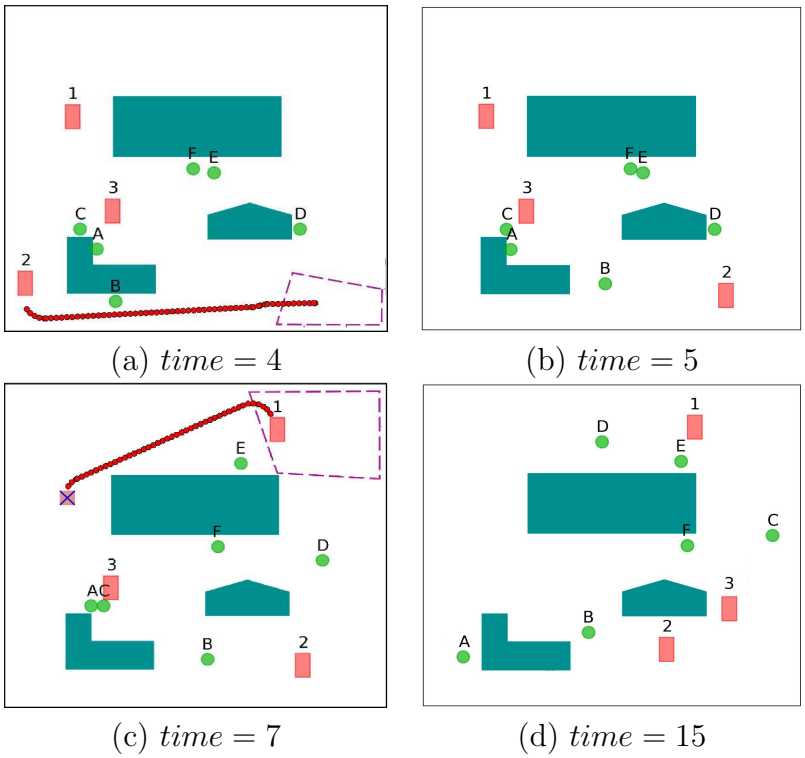


Figure 3.8: Bonnmotion random waypoint experiment at different times: (a) System reconnected by relocation of vehicle 2 to recover D. (b) System is still connected at $time = 5$. (c) Unit E is disconnected and the system is recovered through relocation of vehicle 1. (d) An example system that is unrecoverable by a single vehicle movement.

generate two motion paths corresponding to the vehicles 2 (red) and 4 (blue) using the RRT* [KWP+11b] motion planning algorithm for a Dubins car [IKH11]. We select vehicle 4 for relocation by following the blue trajectory as it gives an optimal cost compared to the red trajectory which requires a longer path to travel.

In Figure 3.8(a) we have three available vehicles forming a relay network while serving six units. Unit D gets disconnected and the candidate vehicle set for relocation is $C = \{2, 3\}$, as relocating vehicle 1 makes the relay network broken. As both of them have hard constrained units ($H_2 = \{B\}$; $H_3 = \{E, F\}$), we use (3.9) to calculate the intersecting polygons X_G^2 and X_G^3 as their respective goal regions.

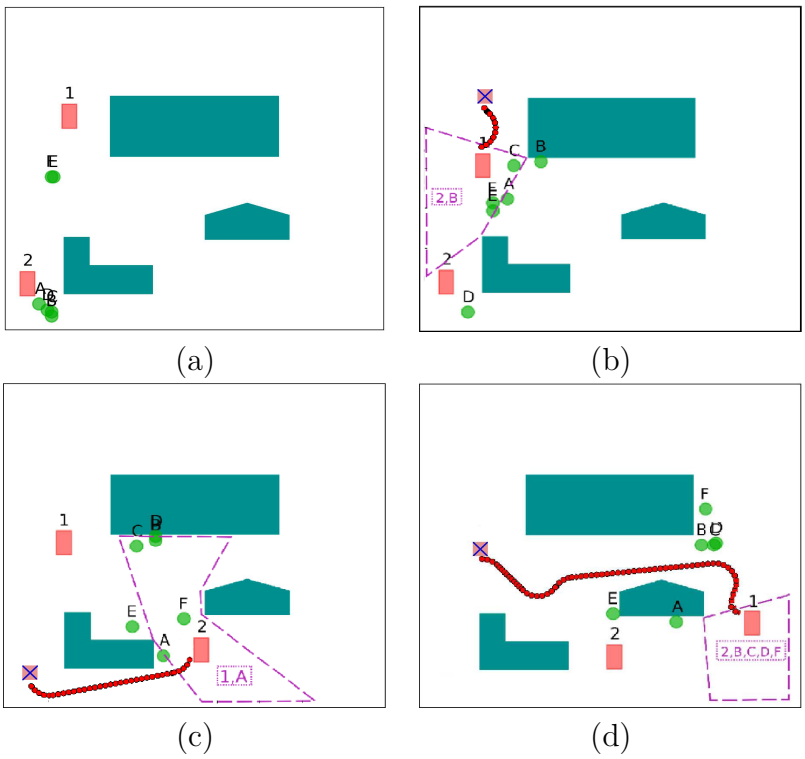


Figure 3.9: Bonnmotion nomadic mobility experiment: (a) All components are connected and the units form two groups; (b) Vehicle 1 is relocated to its goal region X_G^1 (purple area, which is the intersection of vehicle 2 and unit B) in order to serve disconnected unit B; (c) Vehicle 2 moves to serve disconnected unit A; (d) Again, vehicle 1 is dispatched to serve the disconnected unit F.

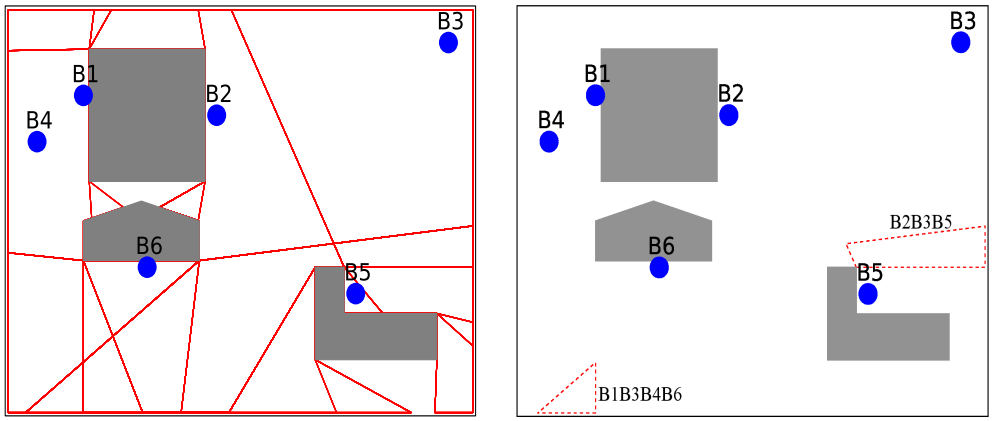
The resulting region that optimizes the visibility is shown as a dashed area for vehicle 2, which is the intersecting visibility region of vehicle 3, hard constrained unit $H_2 = \{B\}$, and disconnected unit D. Vehicle 2 is then relocated into the purple dashed region following the trajectory generated by the RRT* algorithm. At $time = 5$ as shown in Figure 3.8(b), the hard constrained unit B of vehicle 2 changes position and does not break the connectivity. At $time = 7$ (Figure 3.8(c)) we dispatch vehicle 1 to serve the disconnected unit E after the same computations done for the above cases. However, at $time = 15$ (Figure 3.8(d)), the system becomes non-recoverable when unit A gets disconnected. We cannot move vehicle

1 or 2 because there is no common visibility polygon among hard constrained units, another vehicle, and the disconnected unit A. Vehicle 3 cannot be moved due to relay connectivity.

Nomadic Mobility Model: Units move in groups according to the nomadic mobility model and an example scenario is presented in Figure 3.9. We deployed two servicing vehicles in order to provide service to six units that are distributed into two groups (see Figure 3.9(a)). Unit B gets disconnected in the next time-stamp shown in Figure 3.9(b). Accordingly, vehicle 1 goes to the intersection of the visibility polygons of vehicle 2 and unit B (purple dashed). This small movement is highlighted by a red curvature generated by the RRT* algorithm. In Figure 3.9(c), unit A is disconnected and vehicle 2 is dispatched to its calculated goal location X_G^2 (the intersection of the visibility from vehicle 1 and unit A). Then, in the next time-stamp, unit F is disconnected. Only vehicle 1 has a common intersection with vehicle 2, hard constrained unit set $H_1 = \{B, C, D\}$ and disconnected unit F. Therefore, we relocate vehicle 1 to repair the LoS-based visibility network. We observed that the nomadic mobility model is easier to repair than the random waypoint model with a single vehicle movement as the units move in groups.

3.8.3 Re-Establishing a Communication-valid State

We have tested the methodology discussed in section 3.7.2 to establish a communication-valid network (static or dynamic) in case 1) a new setup is needed, or 2) there is no solution with a single vehicle movement once a connected network becomes communication-invalid. At first, a visibility-based polygonal decomposition of the environment was obtained using the VisiLibity [OC08] and Shapely [GBLT] libraries as shown in Figure 3.10(a). After applying algorithm 6, we get the two polygons



(a) (b)

Figure 3.10: (a) Decomposition of an environment using visibility polygons. (b) Selected polygons using approximate set cover algorithm.

presented in Figure 3.10(b) that collectively see all the units. The bottom-left polygon is completely visible from units B_1, B_3, B_4 and B_6 while the units B_2, B_3 and B_5 can see the middle-right polygon.

We tested our approximate solution on several randomly generated environments with six ($m = 6$) units shown in Figure 3.11. Our algorithm was able to select the best polygons according to the labels y_i and scores $\hat{s}_i$ in $\mathcal{F}$. Accordingly, a single vehicle can serve the units deployed in Figure 3.11(a) as we found a single polygon visible to all the units. In the case of 3.11(b), two vehicles are sufficient to cover the two selected polygons. Moreover, the polygons are completely visible to each other and form a single connected component. Therefore, no extra vehicle is required to do patrolling. Then, we need a minimum of three vehicles in the case shown in Figure 3.11(c), where two of them are assigned to each of the polygons while the remaining one connects the two polygons using an approximate TSP tour τ. A scenario is presented in Figure 3.11(d) with three selected polygons. We need three vehicles to form a static relay network as the three polygons are completely visible

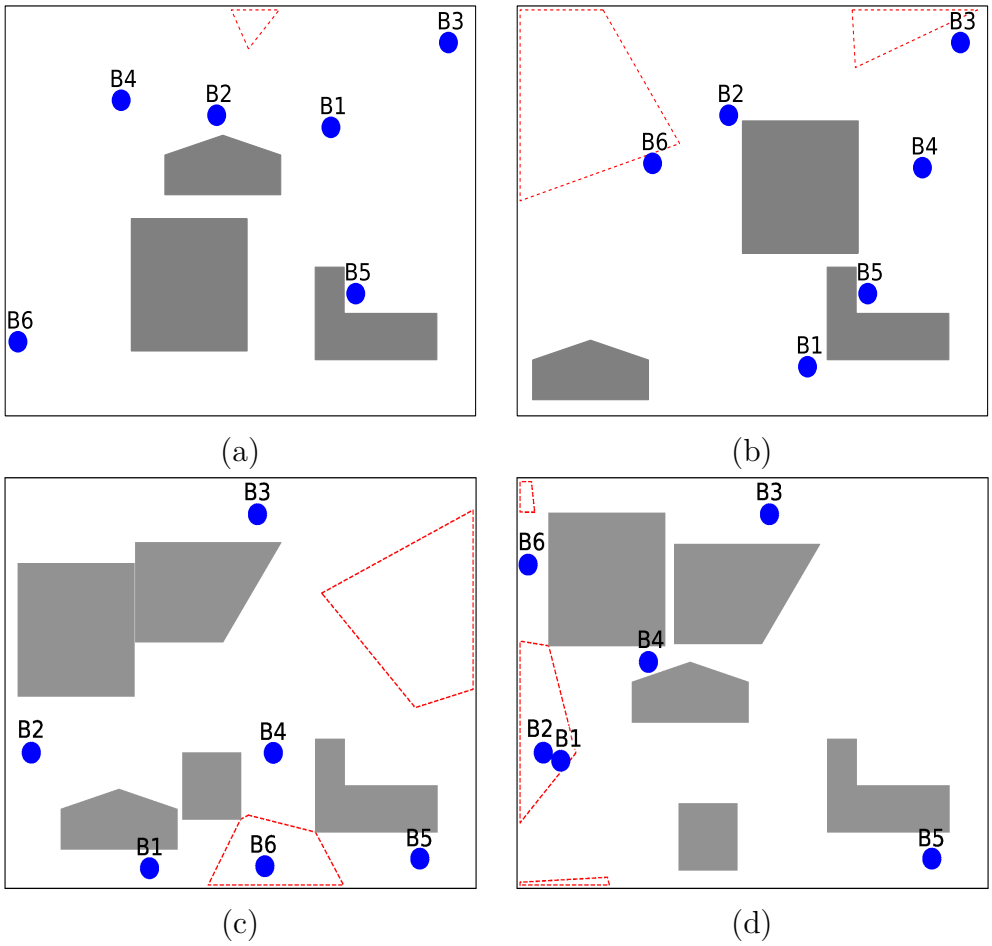


Figure 3.11: (a) A single vehicle is sufficient to serve all the units as $|\Gamma| = 1$; (b) Two vehicles are sufficient as their deployment will result in a single connected component; (c) Three vehicles are required where two of them will be deployed in two goal polygons and the remaining one will do the patrolling between their visibility polygons; (d) Three vehicles can form a static relay network as the three goal polygons are completely visible to each other.

to each other and therefore no further patrolling is required. However, patrolling needs to be planned in case we have less than three available vehicles.

An animated simulation model is developed using ROS and the Gazebo 3D simulator [KH04] where we use a number of Husky cars as our robot vehicles as shown in Figure 3.12. The Husky is a simulated version of *Clearpath Robotics* real UGV, and is widely used in research for its enabling of realistic simulation of real

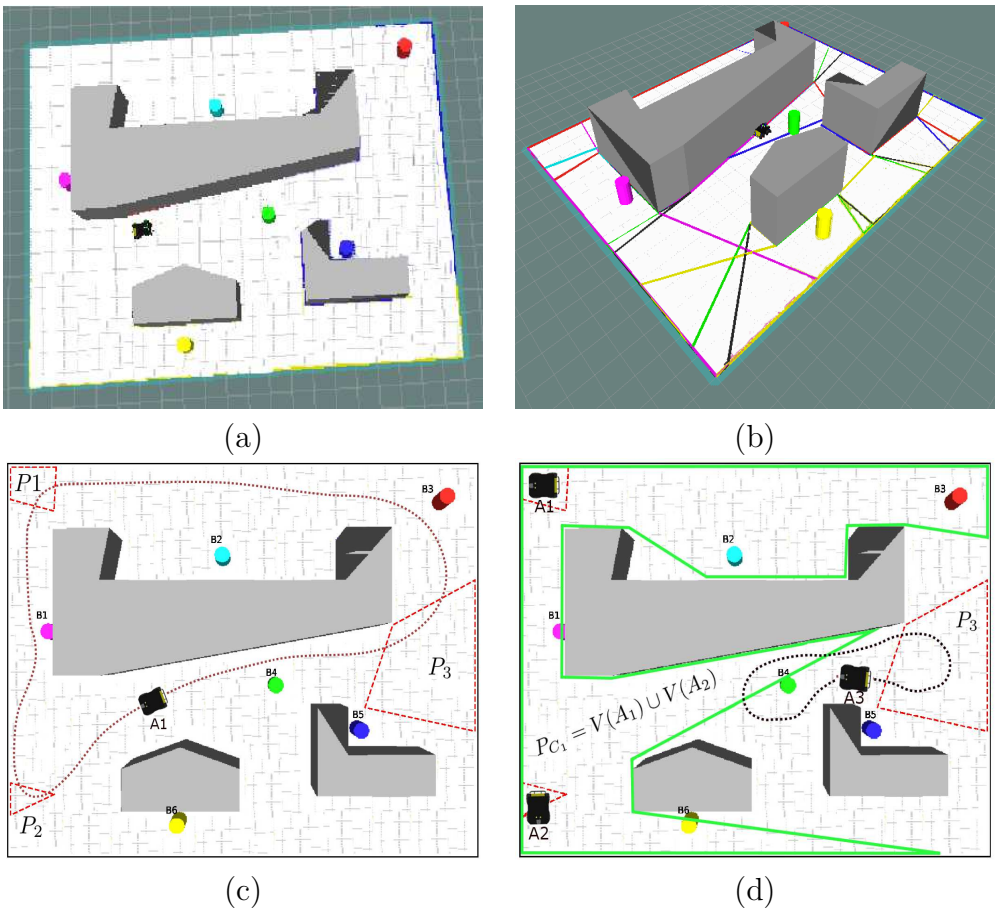


Figure 3.12: (a) ROS and Gazebo simulation environment containing six units presented with various colors; (b) Visibility based decomposition; (c) Planning with one vehicle; (d) Planning with three available vehicles.

world hardware capabilities. The six cylindrical objects in Figure 3.12(a) represent the six ($m = 6$) units, while the black vehicle around the center is the Husky UGV. We first decompose the environment based on visibility polygons of the units as shown in Figure 3.12(b) and three polygons P_1, P_2 and P_3 are selected by the approximate set cover method of Algorithm 6 (see Figure 3.12(c)).

Next, we simulated two cases with one ($n = 1$) and three ($n = 3$) *Husky* cars as shown in Figure 3.12(c)-(d). Given a single vehicle, a motion planning algorithm generates a tour τ that touches the chosen polygons, avoiding the obstacles

(Figure 3.12(c)). The visiting order of the polygons is obtained from the approximate TSP [Chr76] algorithm as discussed in Section 3.7.3. The Husky vehicle uses a motion planner that combines A* with Adaptive Monte Carlo Localization (AMCL) [FBDT99] to move the car from source to goal. However, one can use any other motion planner that conforms the robots dynamics and configuration, such as RRT* or PRM*. In Figure 3.12(d), a case is presented with three available vehicles, so we can assign one vehicle per polygon which serves as static servers. As a result, we get a connected component P_{C_1} and a uncovered polygon P_3 as explained in Section 3.7.3. Therefore, the remaining vehicle follows a patrolling trajectory τ in between P_{C_1} and P_3. Detailed simulations with animation can be found in the attached multimedia of this paper.

3.8.4 Physical Deployment

We performed a physical experiment of our ideas using a modified Traxxas Slash Dakar Truck Series Edition as a servicing vehicle as shown in Figure 3.13(a). An *ArduPilot* controller (ArduPilot Mega APM 2.5) was added to control the movement of the vehicle. A Turnigy 9X radio was used to place a series of waypoints to be followed by the vehicle and as a safety feature in case of communication loss. The Raspberry Pi (version 2) unit was mounted for on-board processing and an external compass/GPS unit was mounted for localization.

We connected a camera (Vilros 5MP Camera Board Module) to the Raspberry Pi as a visibility sensor and used simple color segmentation algorithms for unit detection using the *OpenCV* computer vision library. On top of each unit, a Zigbee communication module (Zigbee+Arduino) was used to communicate with the vehicle as shown in Figure 3.13(b). In Figure 3.13(c) and (d) we see the output of unit

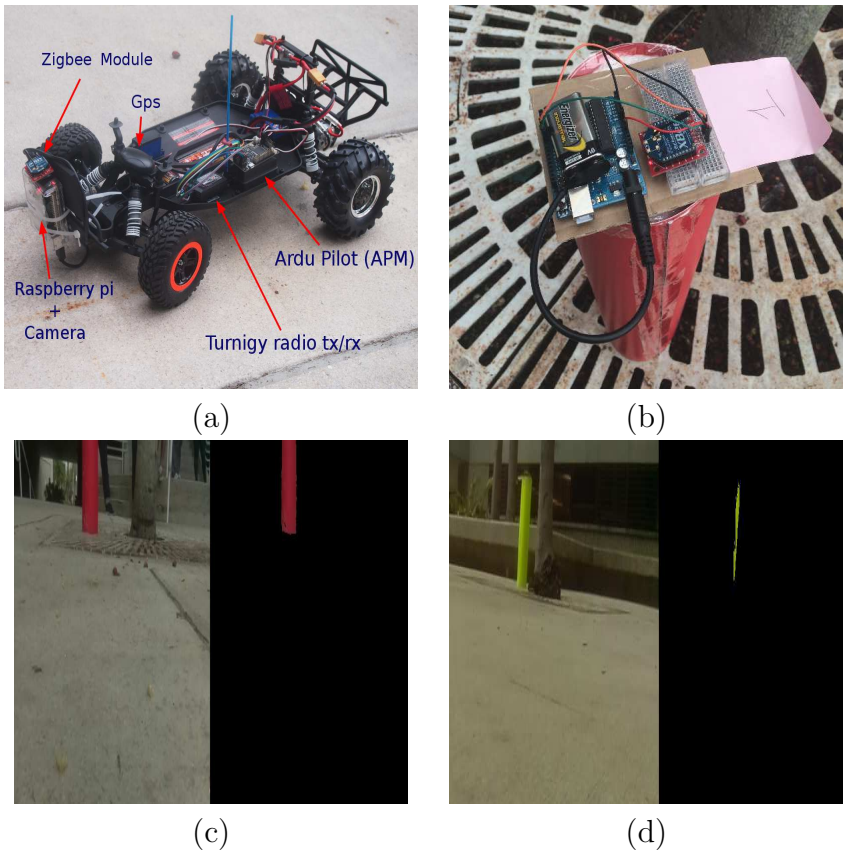


Figure 3.13: (a) Modified robotic truck as a servicing vehicle with APM, Raspberry pi, GPS, Zigbee and Camera mounted on it; (b) A sample unit (Red) with a Zigbee module mounted on it as a communication device; (c) and (d) are the images captured by the camera mounted on the vehicle along with their real time Computer vision output after color based segmentation (to detect red and yellow) shown on the right side of each image.

detection captured by the on-board camera mounted on the vehicle when the units come within the visibility region of the vehicle. The images are processed in real time using onboard processing power while the vehicle is in motion. The Zigbee module is used for passing messages between the vehicle and the units. Detailed experiments with the robots in action can be found in the attached multimedia of this paper.

3.9 Summary

In this paper, we study the problem of establishing Line-of-Sight (LoS) communication between a number of moving vehicles and a group of mobile units. We proposed algorithms to determine if a configuration of units and vehicles is connected through LoS communication. Two polynomial-time versions of the algorithm, one centralized and one distributed, were developed to be deployed for different types of ground missions. Secondly, we proposed a complete algorithm that gives a solution, if there is any, to recover a system by relocating a single vehicle.

In a complex and highly dynamic situation, where a single vehicle fails to repair the network, we solve the general problem of multi-robot relocation and placement. We proved that the exact solution to this problem is NP-hard and then presented heuristic procedures based on set cover approximation to calculate goal locations and paths. In a patrolling scenario with insufficient vehicles, we use the TSP algorithm to visit the calculated goal polygons. Further optimization was achieved in terms of motion or patrolling cost through visibility-based geometry and graph algorithms. Finally, the ideas were extensively tested in a realistic simulated environment with the help of ROS, Gazebo, and the simulated Husky UGV, and in an outdoor experimental deployment with a modified RC car. Several interesting directions are left for future work.

We found that the problem of interest is NP-hard and can only be approximated with an $O(\log n)$ ratio at best. We presented a heuristic solution inspired by set cover approximation that uses visibility polygon decomposition as input and TSP with neighbors for a patrolling sequence. It is clear that this finds a feasible solution, but calculating the exact approximation ratio is still an open problem.

Another extension of our work is to remove assumptions about the known world, $\mathcal{W}$, and obstacles, $\mathcal{O}$. We assumed that the obstacles are known beforehand and the layout can be perfectly decomposed. Ideally, a robot equipped with sensors can create a strategy based on visibility events [LaV06b] to explore the environment and find good LoS locations. We are exploring the related problem of finding competitive strategies for a kernel polygon search and determining if they can be implemented in a mobile vehicle with sensors [IK95]. We also want to remove the need to estimate the state of the units to follow the proposed path. A feedback based planning approach using $\hat{s}$ as a navigation function may help to overcome this problem.

CHAPTER 4

COMMUNICATION AWARE SAFE PLANNING

A communication aware safe planning model assumes that the communication may not be readily available during the entire mission and therefore an ex ante assessment is required to estimate a safe robotic plan before the actual work takes place. Accordingly, we eliminate the strict requirement for strong communication and optimal robot placements presented in chapter 2 and chapter 3. This Chapter solves the problem of communication aware safe motion planning and we choose to analyze an automated building construction project as our study case. Such places are highly hazardous, contain uncertainty and pose a great risk of collision between heavy equipment and human workers as the movements of different objects are not pre-planned and well-communicated. This chapter includes results from our previous publications [RBM+16, RCBM, RCB+].

Construction jobsites are a source of potential accidents which include a significant loss of lives every year due to struck-by collisions involving moving machinery and workers [OSH]. Recent data shows that the percentage of struck-by accidents constituted 17.6% of fatalities and serious injuries among construction workers [CPR13]. During construction planning activities, safety managers and construction engineers might not be aware of the potential hazards on a construction site. Often times, activity sequences are planned to optimize time, available resources, precedence constraints, site congestion etc. However, the overall safety of a plan is frequently neglected as there is no suitable automated safety estimation tool.

We identify two coupled phenomena that affect the level of safety hazards related to struck-by accidents in construction jobsites: (1) the sequence of activities and jobsite layout, and (2) the movement patterns of workers and equipment [BLS14].

The layout of a construction jobsite affects the movement of equipment and workers within the jobsite. In addition, the movement patterns (trajectories) of the equipment and workers continuously evolve due to changes in the requirements of each construction task as well as the addition or removal of obstacles on the jobsite.

4.1 Approach

We focus on better understanding construction operations to reduce hazardous conditions created by a selected construction plan. There are high variability and dynamic changes in construction operations that affect the performance and safety of a project. However, the main activities (e.g. excavation and concrete pouring) can be anticipated and the corresponding equipment such as trucks, cranes, and drill machines can be modeled using state space formulations and motion planning algorithms. Therefore, we propose an *ex ante* analysis model of the deterministic aspects of a construction project to identify its risks. We believe that the high-level deterministic aspects dominate the stochastic aspects and if analyzed properly, can help to prevent and reduce risks.

To the best of our knowledge, our approach is one of the first to consider to use motion planning techniques to evaluate safety scores or determine obstacle-free trajectories for workers and moving equipment. The concrete contributions of our work are the following: 1) We generate a number of distinct alternate construction plans that are possible after considering the precedence constraints. Afterward, we select the plan that would be the best in terms of safety; 2) We develop an activity and event scheduler to simulate all the plans using discrete event simulation and motion planning; 3) We generate a number of safe trajectories for workers to avoid static obstacles and develop a navigation policy in order to avoid moving equipment; 4) We decompose the layout of a construction site in order to generate heatmaps

of the construction layout to identify dangerous hotspots at discrete times; 5) We develop a model that guides the managers to select one of the Pareto optimal plans resulting from the sensitivity/trade-off analysis among the resource, speed, layout modification, duration etc.

4.2 Related Work

In one stream of research, different studies (e.g., [NX13]) have developed optimization-based methodologies for safety assessment of construction site layouts. In another stream of research, discrete event simulation has been adopted for construction planning [Mar96]. These studies have two main limitations: (1) the lack of consideration of the impact of the layout of construction job sites on the spatio-temporal motion trajectories related to the workers and equipment, and (2) lack of consideration related to the dynamic changes in the layout of construction sites at different stages of a project schedule. Our approach for obtaining the safety score is different compared to [NX13] and [Mar96] since our methodology is based on the motion trajectories of workers and equipment. We convert the construction projects into a state space model and investigate deeply into the motion planning layers as movement patterns of equipment and workers are the main causes of struck-by hazards. This allow us to generate time indexed dynamic safety scores based on construction events which is an improvement to the static scores found in related literature.

An effective approach requires to be able to translate high level plans into low level state trajectories in order to enable better safety assessment. Therefore, our ideas are connected to approaches that use Linear Temporal Logic [BKV10] to create high-level specifications that can be translated to low-level trajectories. Our ideas also share commonalities with STRIPS-like representations [GNT04] that connect

with motion planning algorithms [CA09]. However, in contrast to these approaches, we use *Activity Graphs* and *Discrete Event Simulations (DEVS)*. The activity graph enables us to efficiently generate a number of alternate plans while the DEVS model helps us to simulate them in detail using low level motion planning methods.

Some attempts [KL90] have been made in the construction community to incorporate planning algorithms in the analysis of projects. Motion planning has been used to analyze crane motions and their safe operations. In [ZAH10] and [ZHB11] a modification of the Rapidly Exploring Random Tree (RRT) algorithm for re-planning of crane motions was used in real time along with positioning systems for simulation and safety purposes. However, these tools [ZAH10, ZHB11, KL90] are intended only to capture a small part (e.g. one equipment or a single activity simulation) of the activities in a construction project.

Different models are used to simulate construction activities such as [CT13, AH11, KM01]. However, these tools are intended only to provide graphical modeling in a virtual construction site without providing any conclusion about the safety level. Moreover the prior works cannot suggest alternate plans that might be better in terms of safety and other constraints such as project duration and cost. We developed an automated system that can quantify safety level of a plan, suggest alternate plans and compare among those in order to reduce the chance of fatalities during a construction project.

Our ideas are also connected to [PKV10, GFMG04] as these researches propose a hierarchy of task decomposition to accomplish a large task. In contrast, in our work, the decomposition in sub-activities is an input given by the manager as an Activity Graph. We focus on all alternative plans and simulate them to compare them in terms of construction safety, cost, time, and space distribution.

In [LCH+09], the authors are concerned about identifying the possible mistakes in a construction plan and repair them by using a virtual simulation. Although we share similar motivations, [LCH+09] did not consider the effects of moving machinery and workers, layout, and sequence of activities on the plan's safety level. Another stream of research [LW08] is mostly concerned with profit maximization by optimizing resources and cash-flow but they ignore safety aspects of a project.

4.3 Problem Formulation

4.3.1 Activity Graph

The *Critical Path Method (CPM)* [LL03] is widely used in construction projects to determine the minimum amount of time needed to complete a project. An activity graph is a type of CPM with no timing information. The activity graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a directed acyclic graph. An edge, $(v, v') \in \mathcal{E}$; $v, v' \in \mathcal{V}$ is formed if and only if an activity denoted by node v is a precondition of another activity represented by a node v'. It is helpful to consider v as a parent of v'. Additionally, $\mathcal{V}_s \subset \mathcal{V}$ is a set of starting nodes with no incoming edges while $\mathcal{V}_f \subset \mathcal{V}$ is the set of finish nodes who have no outgoing edges. Finally, a sequence of all nodes, $\pi = (v_1, v_2, \ldots, v_n)$, conserving precedence constraints form a *construction plan.*

4.3.2 Construction Physical State Space

Assume that a construction project takes place in a 2D world, $\mathcal{W} = \mathbb{R}^2$. Let the construction timeframe be defined as $T = [0, \infty)$. The initial set of static obstacles is $\mathcal{O}(t) \subset \mathcal{W},\ t \in T$ where the obstacle set, $\mathcal{O}(t)$, is a time variant set, since

new obstacles may appear on the jobsite and old obstacles may disappear as the construction project progresses.

We define the system state space as X^v for an individual activity or node, $v \in \mathcal{V}$, in the planning graph. A particular system state, $x^v \in X^v$, is composed of a number of parameters that describe a subproblem. The parameters in x^v can be configurations, orientations and velocities of moving bodies (e.g. trucks and cranes) as well as the amount of resources (e.g. soil and concrete) used by an activity. Altogether, the entire system state space is defined as $X = X^1 \times X^2 \times \ldots, X^{|\mathcal{V}|}$.

The *time varying state space* is the Cartesian product $Z = X \times T$ and a state, $z \in Z$, is denoted as $z = (x, t)$. There is a number of moving equipment in the system such as trucks, excavators, mixers and cranes represented by the set $\mathcal{B}(t) = \{B_1, B_2, \cdots, B_k\}$. Considering both the moving bodies and static obstacles, the obstacle state space is defined as,

$$Z_{obs} = \{(x, t) \in Z | \mathcal{B}(t) \cap \mathcal{O}(t) \neq \emptyset\} \tag{4.1}$$

and the free space is defined as, $Z_{free} = Z \setminus Z_{obs}$. An *initial state* is defined as, $z_I \in Z_{free}$ and the set of *goal states* are defined as, $Z_G \subset Z_{free}$:

$$Z_G = \{(x, t) \in Z | x \in X_G, t \in T\}. \tag{4.2}$$

4.3.3 Augmented Discrete Event System Specification

Each node of a high-level construction plan in an *activity graph* is represented as an *Augmented Discrete Event System Specification (DEVS)* [Zei84] model. This model is used along with geometric information from the construction site to generate obstacle free paths and policies for moving bodies. Each node in the *activity graph* is associated with an augmented DEVS model.

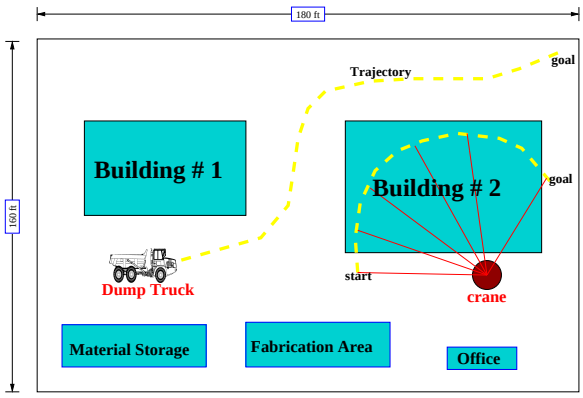


Figure 4.1: An example layout of a construction site. Excavation and concrete pouring need to be done in two buildings. Yellow dotted lines are trajectories of a moving truck and a crane's hook.

The $DEVS$ formalism proposed by [Zei84] and detailed in [Van] and [Van01] is used to formalize discrete event simulation as an extension of finite state automata. An event scheduling model is a tuple ES^v for the activity $v \in \mathcal{V}$ and is represented as:

$$ES^v = (E^v, Z^v, EL^v, f_\eta^v, f_z^v, z_I), \tag{4.3}$$

where $Z^v = X^v \times T$ is the subset of the states of the system. Any activity in a construction site consists of a set of events. The i^{th} event is denoted by η_i and if there are ξ unique events, and we define the finite event set as, $E^v = \{\eta_1, \eta_2, \ldots, \eta_\xi\}$. The event list EL^v is defined by $EL^v = \{(\eta_1, t_1), (\eta_2, t_2), \ldots\}$.

The system starts at time t_0^v with starting state, z_I. The system state is modified based on the current state and an event of an activity:

$$f_z^v : Z^v \times E^v \to Z^v. \tag{4.4}$$

In some cases f_z^v is controlled by the availability of resources (for example the amount of soil that needs to be excavated) and system time. The next event to be scheduled is controlled by f_η^v, based on the current event and system state:

$$f_\eta^v : E^v \times Z^v \to E^v. \tag{4.5}$$

A number of alternate construction plans $\pi_1, \pi_2, \ldots, \pi_k$ are extracted from the CPM graph. To carry out the simulation for each of the plans, π_i, we need to compute the collision free trajectories in Z_{free} space for the moving equipment and workers knowing the initial and goal configurations in X space.

Problem 1: Finding Collision-Free Trajectories for Moving Equipment

Given an initial configuration, x_I, a set of goal states ,X_G, and the set of static obstacles, $\mathcal{O}(t)$, find a collision free trajectory, $\widetilde{Z} : [0,1] \rightarrow Z_{free}$, such that $\widetilde{Z}(0) = z_I$ and $\widetilde{Z}(1) \in Z_G$.

There are m workers, $\mathcal{A}^1, \mathcal{A}^2, \cdots, \mathcal{A}^m$, present in the workspace, who have to travel from their initial position x_I to a destination region X_G. Accordingly our next problem is to compute the safe trajectories for the workers that avoid both the static obstacles $\mathcal{O}(t)$ and moving equipment $\mathcal{B}(t)$.

Problem 2: Finding Safe Trajectories for Workers

Given an initial configuration, x_I, a set of goal regions X_G, the set of static obstacles, $\mathcal{O}(t)$, and the trajectories of the moving equipment, $\mathcal{B}(t)$, find an obstacle free trajectory, $\tilde{x}_{worker}$, such that $\tilde{x}_{worker}(0) = x_I$, $\tilde{x}_{worker}(1) \in X_G$.

Therefore by solving Problems 1 and 2, we have a set of trajectories for each feasible plan π_i.

4.3.4 Safety Evaluation for Different Plans

We need to calculate safety scores for each of the plans $\pi_1, \pi_2, \ldots, \pi_k$ in order to choose the best plan. Problem 3 calculates the safety score for the plans based on the trajectories $\tilde{X}_{worker}$ and $\tilde{Z}$ calculated by solving Problem 1 and 2. To calculate the safety score for individual plans, we evaluate the entire plan by simulating all the nodes. A safety score is defined as a function (detailed definition is provided in

section 4.9),

$$R : \widetilde{Z} \rightarrow [0, 1], \tag{4.6}$$

Where 0 is the safest score and 1 is the most dangerous score for a plan. Therefore we try to minimize the safety score. We calculate the safety score for a plan based on the trajectory paths.

Problem 3: Safety Score Assessment for Different Plans

Given a set of time variant system trajectories, $\widetilde{Z}$, calculate a safety score for the corresponding plan, π, in the closed interval range $[0, 1]$.

Once the safety score is calculated for the alternate plans, the planning managers need to extract the optimal one which provides minimal completion times and optimal safety scores.

Problem 4: Managerial Implication

Given a number of safety scores for several plans $\pi_1, \pi_2, \ldots, \pi_k$, calculate the optimal plan which minimizes the project's finishing time, cost while optimizing the safety score.

4.4 System Overview

A construction plan starts with a 2D layout of the construction site as shown in Figure 4.1. An example critical path management graph (CPM) for this layout is shown is Figure 4.2 where we have two excavation activities (EX) followed by two concrete pouring activities (CP).

The system block diagram of our model used to extract the safest plan is shown in Figure 4.3. An activity scheduler subsystem is responsible for generating alternative sequences of activities. It communicates with the event scheduler subsystem to simulate one or a number of activities. The event scheduler then uses motion

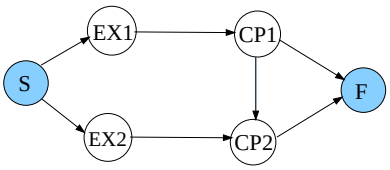


Figure 4.2: An example activity graph of a construction site.

planning algorithms to generate paths for the moving bodies and a coordinator calculates a way to schedule them without colliding with the bodies of other activities.

The obstacles, whether moving or static, have a different impact on the safety of the construction plan. Also, different sequences of plans yield different safety scores.

Definition 4.4.1 *Moving equipment, $\mathcal{B}$, does not affect the safety of two sequential activities. The moving equipment, $\mathcal{B}^u$ and $\mathcal{B}^v$ of two parallel activities, u and v, affect the safety of one another.*

Definition 4.4.2 *Static obstacles, $\mathcal{O}^u$, generated by an activity, u, have a succeeding effect on the safety score of all the successor activities, $v \in \mathcal{V}$, unless the obstacle built earlier is removed by some later activity.*

Proposition 4.4.3 *Different plans yield different safety scores, R.*

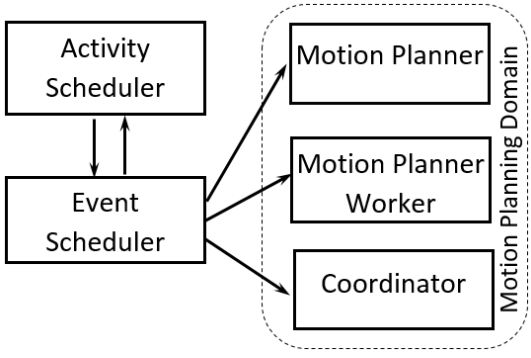


Figure 4.3: System framework and subsystem interaction.

Proof. Suppose we have two alternate plans, π_1 and π_2, from a graph, $\mathcal{G}$. We choose two activities, u and v, where in plan π_1, u is scheduled before v, and in plan π_2, v is scheduled before u. By definition 4.4.1 their safety score is the same. However by definition 4.4.2 if the static obstacles generated by u and v are not same, then the plans yield different safety scores. □

4.5 Plan Extraction from an Activity Graph

A *topological sorting* algorithm is used to extract all possible valid plans. Given n vertices, and a set of integer index pairs, (i, j), of the nodes of the graph, $\mathcal{G}$, where $1 \leq i,\ j \leq n$, the problem of topological sorting is to find a permutation $v_1, v_2, \ldots, v_n$ such that i appears to the left of j for all pairs (i, j) [KS74].

There might be more than one start and finish activities in CPM graph. Accordingly, two dummy activities, v_s and v_f, are added to the graph as starting and final activities with a duration zero in order to create single starting and finishing points (nodes S and F in Figure 4.2). Also the floating activity nodes (without precedence constraint) are added to $\mathcal{G}$ by making v_s as parent and v_f as their child node. By default v_s is labeled as $Visited$ and is the *parent* of all initial nodes, $\mathcal{V}_s \subset \mathcal{V}$, while v_f is the *child* of all the finishing activities, $\mathcal{V}_f \subset \mathcal{V}$.

Given a plan π, produced by the topological sorting algorithm, Algorithm 7 is used for scheduling the activities inside π. A queue, Q_t at time t, is initialized to hold the active (not yet scheduled/visited) activities in Line 2. Line 3 starts a for loop to go over all the activities $u \in \pi$ starting from index 1 (remember activity 0 is the dummy starting activity). Line 5 uses the $ParentVisited$ function to check whether all the parents of the current activity have been scheduled. If not, the activities in Q_t are scheduled by calling the $EventSchedule(Q_t)$ routine and the

time is updated. The corresponding activity nodes are all set as $Visited$ from lines 7 to 9. At line 11, the current activity node is enqueued into partition Q_t at current time t, whose parent nodes have already been scheduled.

A notable property of this algorithm is that it tries to schedule activities in parallel using the activity queue Q_t whenever possible to reduce project completion time. Accordingly, Q_t continues to hold the activities for which the dependency has been met in the CPM graph at time t. We must schedule the activities in Q_t once the parent of a new activity has not been scheduled as it implies that the parent is in Q_t.

Algorithm 7 ActivityScheduler(π,$\mathcal{G}$)

```
1: t ← 0
2: Q_t ← ∅
3: for i = 1 to |π| do
4:    u ← π[i]
5:    if ¬u.ParentsVisited() then
6:       t ← t + EventScheduler(Q_t)
7:       for all v ∈ Q_t do
8:          v.Visited ← true
9:       end for
10:   end if
11:   Q_t.Insert(u)
12: end for
```

The running time of algorithm 7 depends on various sub-methods. The for loop at line 3 runs in $O(|\pi|)$ time. Each activity is scheduled and simulated exactly once either individually or simultaneously with other activities. Therefore the total aggregated calls of line 8 are $O(|\pi|)$ and together the loops in lines 3 and 7 run $O(2|\pi|)$ instead of $O(|\pi|^2)$. The only factor that dominates the running time of the algorithm is $EventScheduler$ sub-method in line 6. Accordingly the running time of algorithm 7 is $O(|\pi|.O(EventScheduler))$.

4.6 Event Scheduling Using Augmented DEVS

Each node in a plan needs to be evaluated through our event simulation method. A queue of activities Q is received from the *ActivityScheduler* routine. Each activity is a collection of events, $\eta \in E$. All of the events in E are motion planning problems which have to be solved before going on to the next event.

Algorithm 8 is used to simulate a number of nodes in the activity graph using our *augmented DEVS* model. In order to carry out the simulation in line 2, we first create an event scheduling model, ES, as defined earlier, for each node. Line 3 extracts the initial state, $z_v \in Z^v$ and line 4 takes the first event from the event set to populate the empty event list. The *while* loop in line 6 is used for scheduling all the events from multiple activities. The *min* method in line 8 helps to extract the immediate event's time from the event list to be scheduled if more than one event is in the list. Consequently, line 9 provides the next event. The $MotionPlanner$ routine in line 10 generates a number of trajectories, $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|})$, each of which contains a sequence of configurations. Any state of the system involves some construction workers moving in the workspace. We generate the trajectories of the moving workers, $\tilde{x}_{worker}$ based on the system state z_v in line 12 using the $MotionPlannerWorker$ subroutine. We will describe their details shortly.

Line 13 calls the *Coordination* routine to generate a set of collision-free-time-variant trajectories, $(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{|Q|})$, for each activity. Lines 14-17 are the updating steps of the system states. On line 15 a new system state, z_v, is calculated based on a current state and event. State z_v keeps track of the *resources, configurations* and other attributes of moving bodies for each of the events along with other information. If $z_v \in Z_G$, then the function f_η in line 16 will generate a $Null$ event. The routine stops when no activity generates any event other than $Null$.

Algorithm 8 EventScheduler(Q)

1: **for all** $v \in Q$ **do**
2: $\quad ES^v \leftarrow CreateDEVS(v)$
3: $\quad z_v \leftarrow initial\ state \in ES^v$
4: $\quad EL_v \leftarrow ((\eta_1^v, 0) : \eta_1^v \in E^v)$
5: **end for**
6: **while** $[(EL_1 \neq \emptyset) \vee \cdots \vee (EL_{|Q|} \neq \emptyset) \wedge t < t_{th}]$ **do**
7: $\quad$ **for all** $v \in Q$ **do**
8: $\quad\quad t^v \leftarrow min\{t : (\eta, t) \in EL^v\}$
9: $\quad\quad \eta^v \leftarrow \{\eta : t^v \in (\eta, t)\}$
10: $\quad\quad \tilde{x}_v \leftarrow MotionPlanner(\eta^v, z_v)$
11: $\quad$ **end for**
12: $\quad \tilde{x}_{worker} \leftarrow MotionPlannerWorker(z_v)$
13: $\quad (\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{|Q|}, \tilde{z}_{worker}) \leftarrow Coordination(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|}, \tilde{x}_{worker})$
14: $\quad$ **for all** $v \in Q$ **do**
15: $\quad\quad z_v \leftarrow f_z^v(\eta, z_v)$
16: $\quad\quad \eta_{new}^* \leftarrow f_\eta^v(\eta, z_v)$
17: $\quad\quad EL_v \leftarrow (EL_v \setminus (\eta, t)) \cup (\eta_{new}^*, t + \tilde{z}_v.t)$
18: $\quad$ **end for**
19: **end while**
20: *return* t

As a *DEVS* simulation system, the running time of Algorithm 8 does not depend on input size. The necessary end conditions for the while loop in line 6 are self generating. To terminate the simulation we put a maximum time t_{th}. This threshold t_{th} (defined by the planning manager) forces the simulation to terminate if all the event lists EL_i from different activities do not finish in time. Once termination is guaranteed, the running time of lines $6 - 19$ is polynomial as the methods $MotionPlannerWorker$ and $Coordination$ take polynomial time which we will show in the subsequent sections. $MotionPlanner$ in line 10 is resolution complete but it takes polynomial time as we have finite search space and specific goal regions.

4.7 Motion Planner

We need two motion planners: one for the moving equipment and another for the workers.

4.7.1 Planning under Differential Constraints

The *MotionPlanner* routine called in Line 10 of the *EventScheduler* routine works based on existing motion planning algorithms. Sampling based algorithms like Rapidly exploring Random Trees (RRT) [LK01] or Probabilistic Road Maps (PRM) [KSLO96] can be applied to calculate the trajectories, $\tilde{x}$, of the moving equipment. Also, RRT* [KWP+11b], an optimized version of RRT, can be applied to generate a better path than a non-optimized RRT. If the planning domain is low dimensional (i.e, a 2D domain), then certain combinatorial planning algorithms, like *trapezoidal decomposition* (see chapter 6 of [LaV06a]), can be applied to achieve an efficient path.

To apply any motion planning algorithms, we have to take motion constraints into consideration. As an example, trucks have the differential constraint of not being able to move sideways. To model the motion of such a truck, let the speed of the truck and the steering angle be specified by the actions u_s and u_ϕ respectively. The transition equation for two consecutive configurations is, $\dot{x}_{tr} = u_s \cos\theta_{tr}$, $\dot{y}_{tr} = u_s \sin\theta_{tr}$, $\dot{\theta}_{tr} = \frac{u_s}{L} \tan u_\phi$ (see chapter 13 of [LaV06a]), where L is the length of the truck.

4.7.2 Planning for the Workers

The subroutine $MotionPlannerWorker$ called in line 12 of the Algorithm 8 is responsible for generating a number of safe trajectories $\tilde{x}_{worker}$, for the workers, avoiding static obstacles, $\mathcal{O}(t)$ and moving equipment, $\mathcal{B}(t)$.

First we discuss the *static obstacle* avoidance policies. We will use the *generalized Voronoi diagram* (GVD) or *maximum-clearance roadmap* [LD81, LaV06a] to compute safe trajectories $\tilde{x}_{worker}$ for workers. The generalized Voronoi diagram was chosen because it is a roadmap whose paths provide maximum clearance from static obstacles (see Figure 4.6). Recall that the set of static obstacles at time t is given by $\mathcal{O}(t) = \{O_1, O_2, \cdots, O_n\}$. We assume that the obstacles are *convex polygons*. If the obstacles are not convex, they can be approximated by surrounding them with a convex shape.

Algorithm 9 presents the pseudo-code of the implemented procedure to find all possible safe paths using the algorithm for a Voronoi diagram of a set of points [BG08]. The obstacle set, $\mathcal{O}$, contains both the static obstacles and the boundary region as the boundary walls are also considered obstacles. In lines 1 and 2 of Algorithm 9, we obtain a set of points, P, containing the midpoints of all the polygonal obstacles and sample points from the boundary region. We apply an existing Voronoi diagram algorithm [For92] (which takes $O(|P| \log |P|)$ time) in line 3 ($GetVoronoi$) to P to generate the Voronoi diagram. Let L be the set of Voronoi edges.

Once the Voronoi edges are generated, we remove the edges that pass through the obstacles. Removing all the Voronoi edges (line 6) that intersect with the *obstacle line segments* result in a set of line segments that approximate the generalized Voronoi diagram (see Figure 4.6). Two for loops in lines $4 - 5$ run in $O(|L|^2)$.

Algorithm 9 CalculateAllPaths($\mathcal{O}$,q_I,q_G)

1: $P \leftarrow \mathcal{O}.getEdges().midPoints$
2: $P.Append(Lines_{boundary}.GetSamplePoints())$
3: $L \leftarrow GetVoronoi(P)$
4: **for** $l \in L$ **do**
5: **for** $l' \in \mathcal{O}.getEdges()$ **do**
6: **if** $l.Intersect(l') == True$ **then**
7: $L.Remove(l)$
8: **end if**
9: **end for**
10: **end for**
11: $G(V, E) = G(L.EndPoints, L)$
12: $E.Add(q_I, Nearest(E, q_I))$
13: $E.Add(q_G, Nearest(E, q_G))$
14: $S = GenAllPath(q_I, q_G, G)$
15: $return\ S$

Finally, we construct a weighted undirected graph, $G = (V, E)$, with weights given by $w : E \to \mathbb{R}_{\geq 0}$. In this graph, V is the set of vertices of the Voronoi diagram and an edge, e, is added for each Voronoi edge. The weight, $w(e)$, for $e \in E$ is given by the Euclidean distance between the vertices that compose the edge, e.

Let $x_I = (q_I, t_I)$ be the initial configuration of a worker and let his goal configuration be $x_G = (q_G, t_G)$ where the points $q_I, q_G \in \mathcal{W} \setminus \mathcal{O}$. We need to connect these two points on the roadmap given by the Generalized Voronoi Diagram. In Line 13 of Algorithm 9, this is achieved by connecting q_I to the nearest Voronoi line, $e \in E$. This introduces a new point on e which is the intersection of e and the normal line of q_I on e. The same procedure is also applied to q_G.

To choose a safe trajectory $\tilde{x}_{worker}$ for the workers, we first compute all possible paths in graph G from q_I to q_G. The method $GenAllPath(q_I, q_G, G)$ in Line 14 generates all possible paths using a variation of Breadth First Search (BFS) [CLRS01]. Afterwards a path is selected as safe if it has no or infrequent collisions with the moving bodies $\mathcal{B}(t)$. This procedure is discussed in the next section 4.7.3.

The Method $GenAllPath$ takes linear time and overall the running time of Algorithm 9 is $O(|L|^2)$ which is taken by lines $4-9$.

4.7.3 Safest path avoiding moving bodies

The worker $\mathcal{A}^j$ must move along his path from $\tilde{x}_{worker}(t_i)$ to $\tilde{x}_{worker}(t_f)$ while the equipment $B^i(t)$ must move along its path over the time interval $T = [t_i, t_f]$ at a speed of ω_i. To avoid colliding with moving equipment on a trajectory, a worker must yield and make a $STOP$ to let the moving equipment pass.

We will modify the *velocity tuning method* (see [KZ86] and [LaV06a], Chapter 7 for details) to obtain a *plan* for the workers with a fixed speed, and two actions, $STOP$ and $MOVE$. Let $U = \{STOP, MOVE\}$ be the two allowable actions. We call a *policy* a mapping, $\pi : T \to U$.

Initially, at t_i both the worker and the moving body $B^i(t)$ start at their initial point of their respective trajectories $\tilde{x}_{worker}$ and $\tilde{x}_i$. Moving bodies at different times in $T = [t_i, t_f]$ occupy different spaces on the worker's trajectory, $\tilde{x}_{worker}$. The solution to the problem of avoiding moving bodies lies in a *space-time* coordinate system. Let $S = [0, |\tilde{x}_{worker}|]$ be the space axis, where $|\tilde{x}_{worker}|$ is the length of the trajectory, $\tilde{x}_{worker}$. We define the time-space as $Y = S \times T$ in which each (s, t) indicates a worker's position along the path, $s \in S$, and time, $t \in T$ [KZ86, LaV06a]. The space occupied by the moving body on the workers' path (obstacles in Y) can be calculated in this space-time coordinate system (see section 7.1.3 in [LaV06a] for details).

Algorithm 10 presents a procedure that creates a plan for the worker using the space-time coordinate system. In line 1 we calculate the straight line in the space-time system from the original trajectory $\tilde{x}_{worker}$. In an S-T system, the worker

starts in $(0, 0)$ and moves along a line having a slope, $m = \frac{dt}{ds}$ and $m = \frac{1}{\omega_{worker}}$ (see Figure 4.7) where ω_{worker} is the speed of the worker. Lines $2 - 5$ calculate all the obstacle regions in the space-time system (blue blocks in Figure 4.7). An obstacle list *obsList* in the S-T system is generated in line 4 for all the moving equipment that cross the workers' trajectory in workspace $\mathcal{W}$. Lines $6-12$ find a policy π which avoids all the space-time obstacles. In line 7 we check whether the line intersects an obstacle region. To avoid the moving equipment the worker needs to stop, which is recorded by updating the policy π in lines 8 and 9. When the line in the S-T system intersects with the computed obstacle regions, it goes up vertically which means that time is moving forward but the worker does not move ($\frac{ds}{dt} = \frac{0}{dt} = 0$) (see STOP mark in Figure 4.7). This waiting essentially makes a delay for the worker to complete his trajectory $\tilde{x}_{worker}$. The *UpdateLine* method in line 10 shifts the starting point of the remaining line section to the upper left corner of the intersecting obstacle region and the process repeats for other obstacles in the S-T system.

Algorithm 10 CalcVelocityProfile($\omega_{worker}, \omega_i, \tilde{x}_{worker}, \tilde{X}_i$)

```
 1: line = CalcLine(x̃_worker)
 2: for i = 0 to |X̃| do
 3:    obs_{s×t} = FindObs(x̃_worker, x̃_i, ω_worker, ω_i)
 4:    obsList.Add(obs_{s×t})
 5: end for
 6: for b ∈ obsList do
 7:    if intersect(b, line) then
 8:       π(b.lowerLeftY) = STOP
 9:       π(b.upperLeftY) = MOVE
10:       line = UpdateLine(line)
11:    end if
12: end for
13: return π
```

In Algorithm 10, Lines $2-5$ that are responsible for calculating the *obstacle blocks* that dominate the total running time. Method $FindObs$ runs in $O(|\tilde{x}_{worker}|.|\tilde{x}_i|)$ [RCBM]. Therefore the running time of lines $2-5$ is roughly $O(n^3)$.

4.8 Coordination Space to Prevent Robot-Robot Collision

The sequence of trajectories of moving equipment, $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|}$, is generated by the motion planner for each activity regardless of whether they collide or not with the bodies (equipment or worker) of the other activities which may run in parallel. Hence, the bodies following the trajectories may collide with the bodies of other parallel activities or the moving workers. Given m moving bodies, an m-dimensional coordination space, $\Gamma = [0,1]^m$, is represented as a unit cube that schedules collision free paths for the moving equipment [LH98a]. The i^{th} coordinate of Γ represents the domain, $\Gamma_i = [0,1]$, of the path $\tilde{x}_i$. Let γ_i denote a point in Γ_i. The pairwise robot-robot (body-body) obstacle region is, $\Gamma_{obs}^{ij} = \{(\gamma_1, \ldots, \gamma_m) \in \Gamma | \mathcal{B}^i(\tilde{x}_i(\gamma_i)) \cap \mathcal{B}^j(\tilde{x}_j(\gamma_j)) \neq \emptyset\}$ which is combined to yield $\Gamma_{obs} = \bigcup_{i,j \ i \neq j} \Gamma_{obs}^{ij}$. Therefore, $\Gamma_{free} = \Gamma \setminus \Gamma_{obs}$.

At state $(0, 0, \ldots, 0) \in \Gamma$, all bodies are in their initial configurations, $x_i = \tilde{x}(0)$, and at state $(1, 1, \ldots, 1) \in \Gamma$, all bodies are in their goal configurations. Any continuous path, $h : [0,1] \rightarrow \Gamma_{free}$, for which $h(0) = (0, 0, \ldots, 0)$ and $h(1) = (1, 1, \ldots, 1)$ moves the bodies to their goal configurations (see chapter 7 of [LaV06a]). We applied the A^* search algorithm [RN09] on Γ to generate a path h avoiding robot-robot collisions. A body is allowed to move with a constant speed or directed to remain stopped to yield the other bodies to pass by moving horizontally or vertically in Γ (see chapter 7 of [LaV06a]). This A^* search takes polynomial time as the search space is finite and we have a single goal.

By applying the above methodologies of coordination among the bodies and workers, we get the set of time variant trajectories, $\tilde{Z}$.

4.9 Safety Model

We consider a construction boundary to be a perfect rectangular area. If it is not perfect, we can approximate it with a bounding rectangle and convert the added area into static obstacles. We need to decompose the environment into a number of regions to assign a safety score and generate a risk heatmap that provides a visualization of dangerous regions in a workspace over time. Any primitive geometric shape can be used. We used squares because we approximated the environment as a rectangle. The size of the squares does not affect the computation of the algorithms as all the algorithms are used to either generate trajectories or conduct discrete event simulations.

We decompose the workspace $\mathcal{W}$ into δ number of squares. The safety scores of all the squares at a time, t, contribute to the safety of the plan at that time. Safety score of a square is dynamic, time dependent and is inversely proportional to its distance to moving equipment.

Assume that the duration of a plan, π, is T where T is divided into a number of discrete time points $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, j\Delta t\}$ with constant time intervals, Δt, such as $j = \frac{T}{\Delta t}$. We calculate the safety scores in discrete times of $\mathcal{T}$. Let $R(g_i, t)$ denote the score for square g_i of the grid at time, t. Then the definition of $R(g_i, t)$ is,

$$R(g_i, t) = \sum_{j=0}^{|Q_t|} \sum_{k=0}^{|B_j|} \frac{\alpha}{d(g_i, B_k(t)) + \beta}. \tag{4.7}$$

where $d(.,.)$ is a distance function (such as the *Euclidean Distance*) and Q_t is the queue of activities at time t. Parameters α and β are the scaling factors for a better

score. The safety scores for the squares inside the obstacles (static or dynamic) are,

$$R(g_i, t) = 1. \tag{4.8}$$

The average safety score, $r_{grid} : T \rightarrow [0, 1]$, for a grid with δ squares at time t is,

$$r_{grid}(t) = \frac{\sum_{i=0}^{\delta} R(g_i, t)}{\delta}. \tag{4.9}$$

The safety score at time, t for a particular activity plan, π, depends on $r_{grid}(t)$ and equipment-equipment distances (e.g. vehicle-vehicle, vehicle-crane from Coordination). Therefore the total safety score r_π can be calculated by averaging these values over $\mathcal{T}$,

$$r_\pi = \frac{1}{|\mathcal{T}|} \sum_{t=0}^{|\mathcal{T}|} \left[r_{grid}(t) + \sum_{i=1}^{|\mathcal{B}(t)|} \sum_{j=i+1}^{|\mathcal{B}(t)|} \frac{1}{d(B_i, B_j)} \right]. \tag{4.10}$$

We also calculate aggregated safety score over a time interval, $[t_i, t_f]$ where $t_i, t_f \in \mathcal{T}$. The safety score $r_{agg}(g_i)$ for a square g_i then is,

$$r_{agg}(g_i) = \frac{\sum_{t=t_i}^{t_f} R(g_i, t)}{t_f - t_i}. \tag{4.11}$$

4.10 Optimal Plan Computation

The proposed discrete event based simulation system is a novel decision support tool that presents the project manager with a quantified safety score. However a safe plan may be the slowest one or an increase in resources may incur additional safety hazards while competing the project early. These phenomena lead to Pareto optimality where we may not have a plan that is better in terms of all the attributes to be optimized.

Project Duration: T is defined as the project completion time that we get from the $DEVS$ simulation model. This is usually the difference of the starting and finishing times of simulation.

Cost: A slow sequential plan yields the lowest safety score that has less obstacles present at any time. But it is undesirable as modern construction projects have cost, resource and time constraints. Therefore, a plan π is partitioned and all activities in a partition Q_t (from Algorithm 7) are carried on in parallel. The total cost L of a construction project is defined as,

$$L = \sum_{Q_t} \sum_{j=0}^{|Q_t|} \left(\sum_{B \in \rho(Q_t[j])} l_B.\kappa(B, Q_t[j]).t_{Q_t[j]} \right) + l_F.t_{Q_t}. \tag{4.12}$$

Where $\rho : \mathcal{V} \to \mathcal{B}$ gives the name of required moving equipment $B \in \mathcal{B}$ (e.g. truck, crane) for an activity $v \in \mathcal{V}$ and $\kappa : \mathcal{B} \times \mathcal{V} \to \mathbb{N}$ gives the count of each piece of equipment. l_B is the rental cost of the equipment and l_F is the fixed cost (salary, material cost etc) per day. $t_Q \in [0, T]$ is the time required to complete all activities in partition Q.

Safety: The quantified safety values $r_{grid}(t)$ from (4.9) over discrete times, $\mathcal{T}$ are used to calculate the mean safety value μ_π and standard deviation of safety σ_π for a particular plan π. A plan is safer if both the values are low.

The decision to select an optimal plans requires the evaluation of all the above attributes/objectives and is defined as a tuple,

$$Y_{\pi_i} = (\ L_{\pi_i},\ r_{\pi_i},\ \mu_{\pi_i},\ \sigma_{\pi_i},\ T_{\pi_i}\ ) \tag{4.13}$$

Optimal Plan Selection: Therefore, we need a plan which optimizes the construction cost, safety scores and finishing times. There might not be any single tuple (Y_{π_i}) that minimizes all of these objectives. This tradeoff among the attributes leads to a well-known *Pareto Optimization* [KHB02] problem. Exact solutions for multi-criteria optimizations are NP-hard [CP07].

We therefore design an approximate solution model that is compromise of all the objectives. Accordingly an optimum tuple is computed by taking the minimum for

each objective from all the available tuples Y_{π_i},

$$Y^{min} = (\ L^{min},\ r_{\pi}^{min},\ \mu_{\pi}^{min},\ \sigma_{\pi}^{min},\ T^{min}\ ) \tag{4.14}$$

A plan π_i dominates π_j (denoted by $\pi_i \prec \pi_j$) if $\forall k\ Y_{\pi_i}[k] \leq Y_{\pi_j}[k]$. We discard all such dominated plans and the remaining non-dominated plans are then Pareto optimal [Tar07b]. The normalized tuples, $Y_{\pi_i}^{norm}$ for the plans are,

$$Y_{\pi_i}^{norm} = \frac{Y_{\pi_i}}{Y^{min}} = \left(\ \frac{L_{\pi_i}}{L^{min}},\ \frac{r_{\pi_i}}{r_{\pi}^{min}},\ \frac{\mu_{\pi_i}}{\mu_{\pi}^{min}},\ \frac{\sigma_{\pi_i}}{\sigma_{\pi}^{min}},\ \frac{T_{\pi_i}}{T^{min}}\ \right) \tag{4.15}$$

Therefore we choose the plan π_i that yields 1) the closest distance of $Y_{\pi_i}^{norm}$ (e.g. Euclidean distance) to the optimal tuple Y^{min}; 2) the safety score in which r_{π_i} is below the median ($\Phi = med([r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_k}])$) safety score.

$$\underset{\pi_i}{\operatorname{argmin}} \left[\sum_{k=0}^{|Y_{\pi_i}|} (Y_{\pi_i}^{norm}[k] - Y^{min}[k])^d\right]^{\frac{1}{d}},\ s.t.\ r_{\pi_i} < \Phi \tag{4.16}$$

The term in the bracket represents the Euclidean distance when $d = 2$.

4.11 Case Study Examples

In the activity graph shown in Figure 4.4(a), the nodes S and F are dummy nodes created to hold starting and final points. Concrete pouring in building site 1 ($CP1$) cannot be carried out before excavation($EX1$). Therefore, $CP1$ has precedence constraints on $EX1$. Likewise, the activity $EX2$ must be completed before $CP2$ as it depends on the completion of $EX2$.

4.11.1 Alternative Plans and Activity Scheduling

We used the Python programming language to implement a topological sorting algorithm as proposed in [VR81]. The following are three alternate plans (sequence of activities) generated for the activity graph shown in Figure 4.4(a):

Plan 1(π_1)	$EX1 \rightarrow CP1 \rightarrow EX2 \rightarrow CP2$
Plan 2(π_2)	$EX1 \rightarrow EX2 \rightarrow CP1 \rightarrow CP2$
Plan 3(π_3)	$EX2 \rightarrow EX1 \rightarrow CP1 \rightarrow CP2$

For the plan, $\pi_1 = [EX1, CP1, EX2, CP2]$, the Activity Scheduler routine in Algorithm 7 initially loads activity $EX1$ in Q. Q always holds the activities that can be executed in parallel. During the second iteration of the algorithm's loop, it cannot load activity $CP1$ into Q as its parent activity $EX1$ is still in Q. Therefore $EX1$ is scheduled using Algorithm 8 and $CP1$ is loaded into Q. $EX2$ is also loaded in the next iteration, since its parent S is a dummy node. Before loading $CP2$, we simulate the two activities in the queue $(CP1, EX2)$ simultaneously using the event scheduler. In the final run $CP2$ is simulated. π_2 is also simulated in the same way, but we simulate π_3 sequentially by scheduling one activity at a time to compare it with the parallel plans π_1 and π_2.

4.11.2 Discrete Event Scheduling

A Python program with the *SimPy* simulation module [Sim] was used to simulate the discrete event scheduler of Algorithm 8. An event scheduling model, $ES = \{E, Z, EL, f_\eta, f_z, z_I\}$, for each activity is created. For example, in Figure 4.4(b) there are three possible repeating events shown for the crane in charge of concrete pouring(CP). These are *Load(L), Rotate(RO) and Dump(D).* For excavation(EX), shown in Figure 4.4(c), a dump truck in charge of carrying soil has four such states: *Load(L), Haul(H), Dump(D) and Return(R).* The following are two example DEVS models for excavation and concrete pouring activities:

- The set of events for concrete pouring is $E^{CP} = \{L, RO, D\}$ and the set of events for excavation is $E^{EX} = \{L, H, D, R\}$.

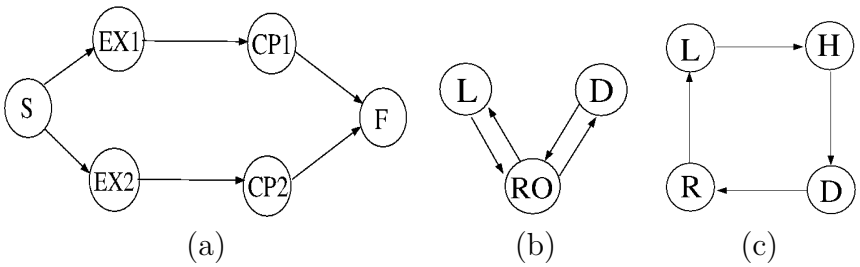
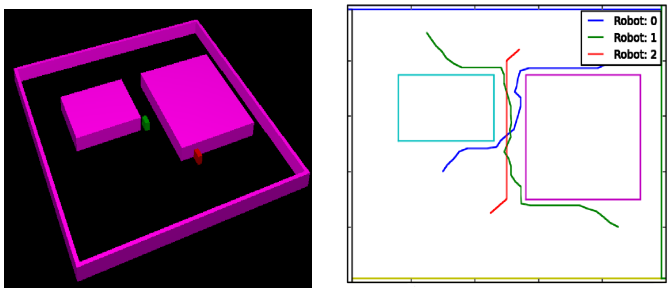


Figure 4.4: (a) A CPM activity graph for a construction plan. $DEVS$ event transition models for (b) Crane; (c) Truck.

- The state, Z, contains the configuration of all parameters such as *resources, interruption, deadline*, etc.
- The configuration of the dump truck is $\mathbb{R}^2 \times \mathbb{S}^1$ while the configuration for the *non-holonomic* [LWW+14] crane is $\mathbb{RP}^2$ as it can rotate with *pitch* and *yaw*, but no *roll*.
- An example state we use for the truck is, $z = (x_{tr}, y_{tr}, \theta_{tr}, \eta_{ex}, r_{ex}, t_{ex})$, and an example state we use for the crane is, $z = (\theta_{cr}^{pitch}, \theta_{cr}^{yaw}, \eta_{cp}, r_{cp}, t_{cp})$.
- An example event transition for the dump truck is, $f_\eta^{EX}(L, z) = H$, as hauling is carried out after loading. Similarly, the crane starts rotating once it is loaded with concrete, $f_\eta^{CP}(L, z) = RO$ (See Figure 4.4).
- An example state transition for an excavation is, $f_z^{EX}(L, z) = (x_{tr}^{new}, y_{tr}^{new}, \theta_{tr}^{new}, H, r_{ex} - r', t_{ex} + t')$. $(x_{tr}^{new}, y_{tr}^{new}, \theta_{tr}^{new})$ is the new configuration of the truck. The constant, $r' \in \mathbb{N}$, denotes the units of soil/resources consumed per iteration and $t' \in \mathbb{R}^{>0}$ is calculated from a *Coordination* function as described previously.

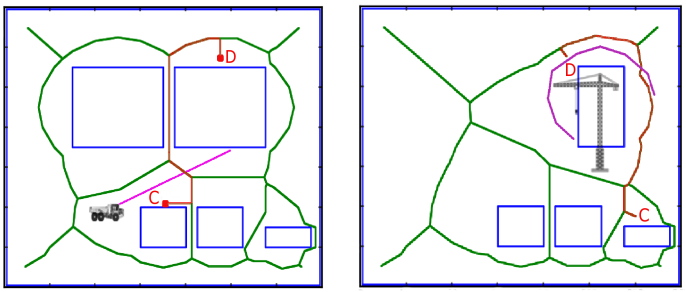
4.11.3 Motion Planning and Coordination

We used the Motion Strategy Library (MSL) [htt] to generate trajectories of moving equipment for different activities (See Figure 4.5(a)). Sample trajectories for two



(a) (b)

Figure 4.5: (a) Two trucks in MSL library colored red and green moving around pink excavation areas (b) Trajectories generated by the MSL library (blue and green). Red trajectory was added to simulate moving worker.



(a) (b)

Figure 4.6: Generalized voronoi diagram of two sample construction sites; (a) A truck is moving along the pink colored trajectory; (b) A crane is moving along a pink semicircle. The shortest trajectory colored in red from position C to position D for the workers is shown.

equipment, $\tilde{x}_1, \tilde{x}_2$ (colored blue and green) are shown in Figure 4.5(b). The red trajectory in Figure 4.5(b) is the path of a worker that we have generated using the *Generalized Voronoi Diagrams* [RCBM].

Two exemplary generalized Voronoi diagrams generated by Algorithm 9 of a site are shown in Figure 4.6. The red lines are the shortest trajectories ($\tilde{x}_{worker}$) derived for the workers, following the safe Voronoi edges. A dump truck is moving back and forth in Figure 4.6(a) while a crane in 4.6(b) is following a semi-circular path (see pink trajectory).

Safe trajectory avoiding dump truck: Figure 4.7 is the space time system generated by Algorithm 10. Initially the worker starts moving freely along the trajectory $\tilde{x}_{worker}$, at a constant speed of ω_{worker}. At some point (marked with "STOP") the worker has a possibility of colliding with the dump truck. The duration of the collision is 24 $units - 15$ $units = 9$ $units$. We advise the worker to stop, which is indicated by the vertical green line from time 10 to 24. The truck will come back to the opposite direction on the workers' path at time 43 $units$ as indicated by another rectangle centered at $(25, 43)$. This time the worker has already passed, so there will be no collision. The worker finishes at $(124, 76)$ which means that the worker took 76 $units$ of time to complete a 124 $unit$ long path.

We considered three alternative paths presented in Figure 4.8. The path of Figure 4.8(a) is a good one in terms of safety as it has no collision and takes 77 $units$ of time to finish the length of 155 $units$, which is longer than the shortest path, but safer. The path in Figure 4.8(c) is also safe, as it has no collisions, but it is long. Similarly, the path in Figure 4.8(e) is the longest with a length 300 $units$ and with some collision risk (see Figure 4.8(f)).

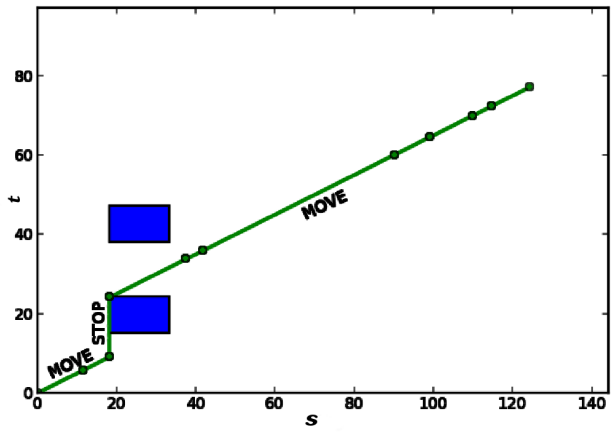


Figure 4.7: Obstacles in $s \times t$ space. Vertical line means STOP. Diagonal lines mean MOVE.

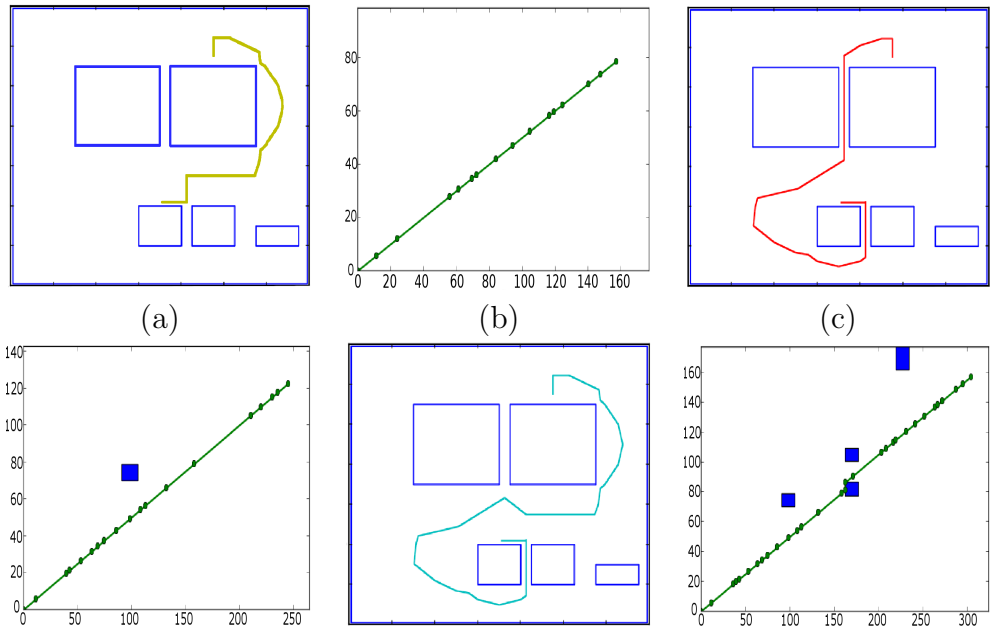


Figure 4.8: (a) (c) and (e); Three alternate paths that are not the shortest; (b), (d) and (f) Corresponding velocity profile guidelines from the $s \times t$ graph. There are no collisions for (b) and (d), but the paths are longer. (f) is totally unacceptable as it traverses a long distance having a high chance of collision too.

Safe trajectory avoiding moving crane: A high boom crane is present in the site to pour concrete into the Building#2 as shown in Figure 4.1. The workers must avoid the hook of the crane as the attached bucket full of concrete can suddenly fall on them which can cause serious injuries and fatalities.

Suppose a worker wants to visit the site from location $C \rightarrow D$. The shortest trajectory $\tilde{x}_{worker}$ using the Voronoi diagram is depicted in Figure 4.6(b). Figure 4.9 is the $s \times t$ space for other alternate trajectories. The workers trajectory collides with the crane's hook twice (as shown in Figure 4.9(b)), if he selects the shortest path shown in Figure 4.9(a). The worker must wait until the hook clears his path. We conclude that the worker will reach to his destination at time 115 *units* while the path is 130 *units* long. The alternate trajectory in Figure 4.9(c) is the second

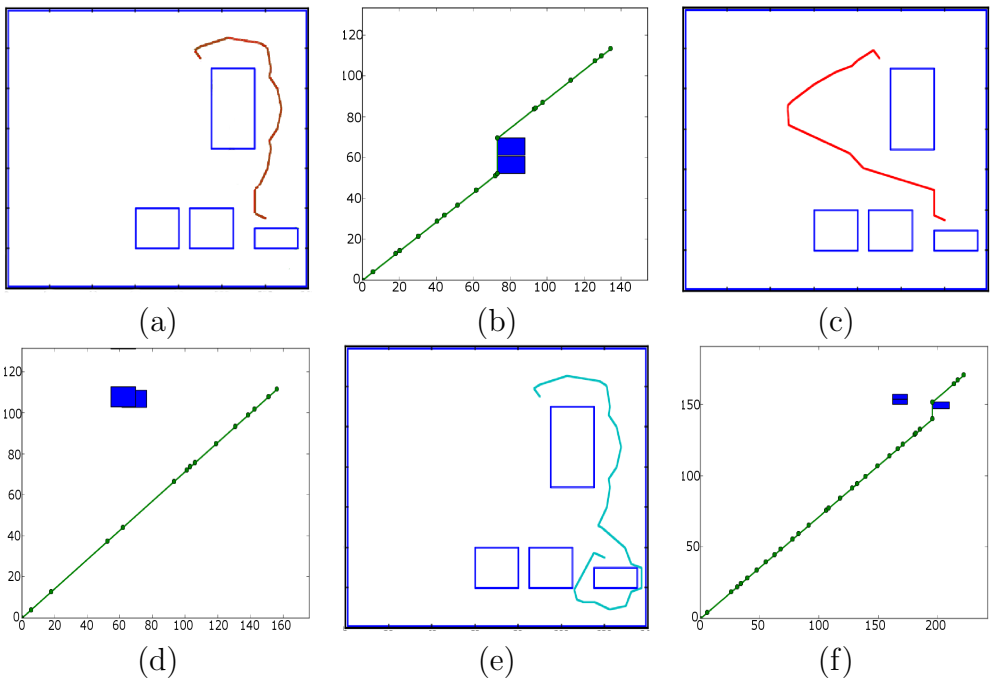
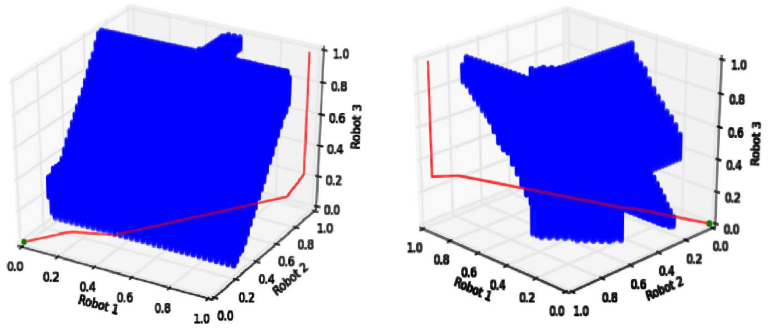


Figure 4.9: $s \times t$ space guiding the velocity profile for a crane. (a) Two consecutive obstacle regions found. (c) and (e) Two alternative paths that are not shortest; (d) and (f) are the corresponding velocity profile guidelines from $s \times t$ graph.

shortest path and does not have a collision. According to our *safety score*, this is a better choice than the shortest path in Figure 4.9(a). The worker reaches his destination in 115 *time units* travailing a path of length 160 *units*. Even though the path is longer, since it has no collisions, it has a lower safety score than the shortest path. We tested another alternate path as shown in Figure 4.9(e) which is much longer and involved in a collision (see Figure 4.9(f)).

Coordination: The *Coordination* subsystem generates policies for equipment-equipment and robot-worker collision avoidance. A three body coordination space is shown in Figure 4.10 using the trajectories of Figure 4.5(b). For better visual understanding we present the 3D image from two different viewing angles. Blue regions comprise collision configurations, Γ_{obs}, for three possible combinations of truck1-truck2, truck1-worker and truck2-worker. The continuous red path, h, is



(a) (b)

Figure 4.10: 3D Coordination space for robots from two different viewing angle. Blue regions are obstacle areas Γ_{obs}. Red line is the collision free path

computed using an A^* search algorithm which connects the point from the initial configuration, $(0, 0, 0)$, to the goal configuration, $(1, 1, 1)$.

4.11.4 Safety Evaluation

The safety scores were calculated using the motion profiles.We developed a Python tool for safety heatmap visualization as shown in Figure 4.11. A safety score for each square g_i in a grid was calculated by taking aggregated safety over time using equation (4.11). The green colored regions are the safest and red regions are the most dangerous. Figure 4.11(a) and (b) show the hazardous zones for two sample activities $(CP1, EX2)$ and $(EX1, EX2)$ where a dump truck and a crane were allocated for excavation and concrete pouring respectively. Two other heatmaps for an alternate plan where we double the resources (two trucks per excavation and two cranes per concrete pouring) are shown in Figure 4.11 (c) and (d). Finally we generated heatmaps for another plan where we relocated the equipment's starting and goal locations as shown in Figure 4.11(e) and (f) to complete the sensitivity analysis.

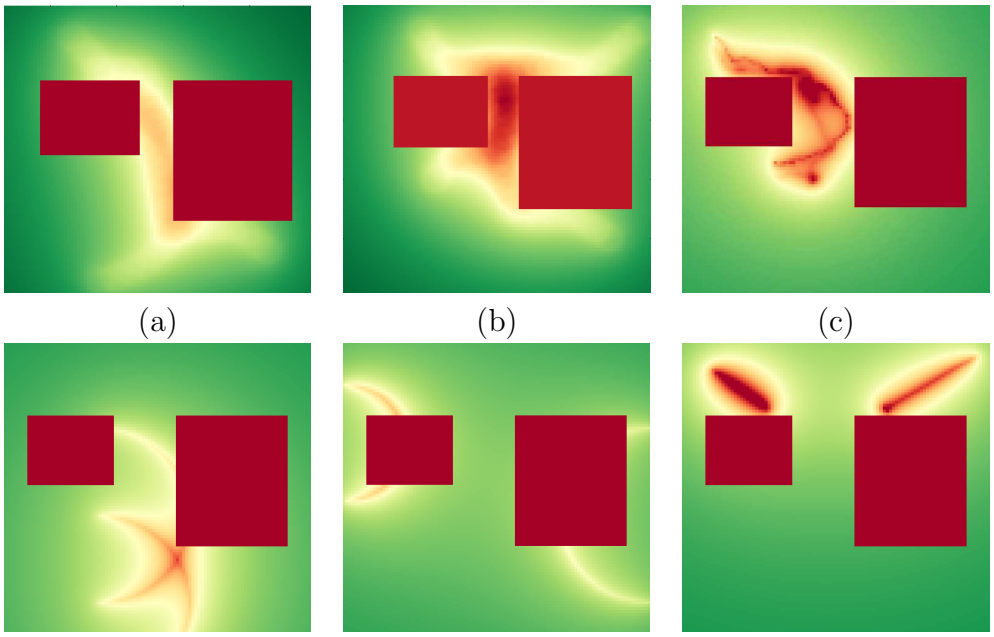


Figure 4.11: Aggregated heatmaps (using (4.11)) for the activities. (a) CP1EX2; (b) EX1EX2; (c) EX1 with increase number of trucks (two trucks); (d) CP2 with two cranes; (e) CP1CP2 with two cranes that have been relocated; (f) EX1EX2 when the initial loading and final dumping positions are changed for the trucks.

4.11.5 Sensitivity Analysis

A sensitivity analysis is used to identify the objective (see (4.13)) that affects construction safety most. The sensitivity test is conducted by keeping one attribute fixed while varying the other inputs. Here we carry out an exemplary sensitivity test that evaluates the cost (L_π), safety (r_π), timeline (T_π) etc. attributes.

The timeline chart for the plans, π_1, π_2 and π_3, is shown in Figure 4.12(a). Each box of this chart under a particular plan represents a partition Q_t composed of one or more activities that can be simulated together using $DEVS$. Figure 4.12(b) shows a graph that presents the change of safety scores over time for different plans.

In Figure 4.12(c), we demonstrate the effect of resource increase for activities that dominate the safety score of the plans. Most importantly it increases the

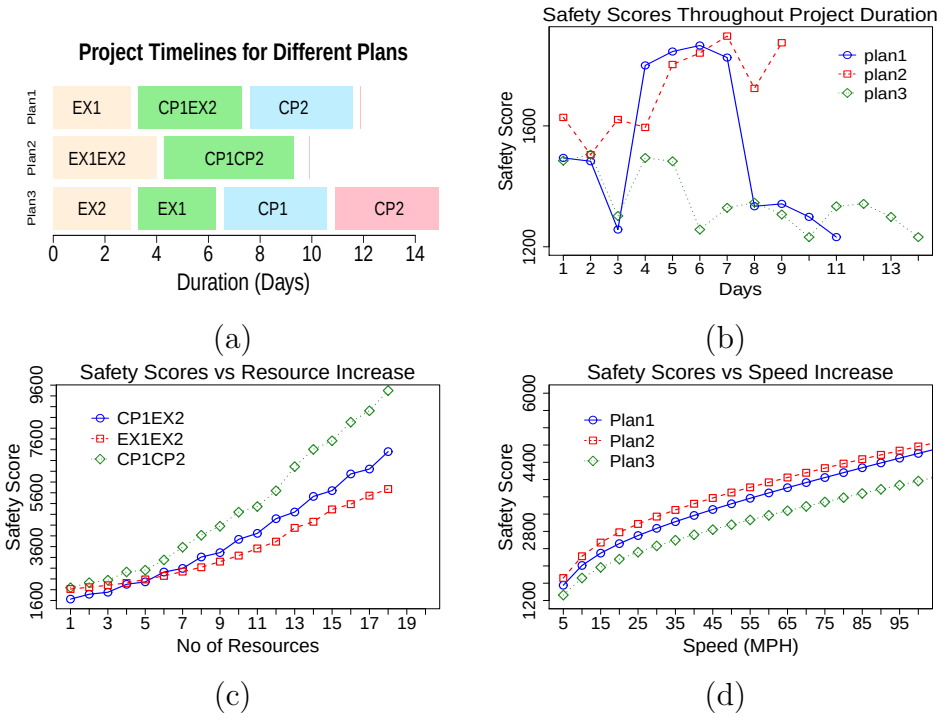


Figure 4.12: (a) Time chart for different plans; (b) Variation of safety over time; (c) Variation of safety due to resource increase; (d) Variation of safety due to space relocation.

safety score r_π following the increase of dump trucks and cranes as shown in Figure 4.12(c).

Next, we increase the speed of the equipment that make construction faster. The effect of speed increase was evaluated according to the speed-collision relationship described in [ECA04] which is adapted here in the form $r_\pi^n = r_\pi^{n-1} + \xi(\frac{s+\triangle s}{s})^4$. Here $\triangle\ s$ is the speed change and ξ is the user defined weighting factor. The safety score curves are shown in Figure 4.12(d) where we see that the scores are increasing rapidly with the increase in speed for all the plans.

A comparison analysis over various plans is presented in Table 4.1 where based on the original plans, we generate additional plans by changing 1) the amount of resources (π_{*2}), 2) site space organization (π_{*3}) and 3) speed of the equipment

Table 4.1: Safety Analysis and Optimal Plan Selection

Plan	Sensitivity	Cost (L_π)	r_π	Timeline (T_π)	L_π^{norm}	r_π^{norm}	T_π^{norm}	Domination	$Y_\pi^{norm} - Y^{min}$
π_1	Original (π_{11})	2250	1525	11	1.07	1.17	2.2	Dominated by π_{13}	1.21
	Resource Increase (π_{12})	2100	1745	6	1.0	1.34	1.2	Not Dominated	0.39
	Space Changed (π_{13})	2100	1429	10	1.0	1.09	2.0	Not Dominated	1.00
	Speed Increase (π_{14})	2150	1605	8	1.02	1.23	1.6	Dominated by π_{23}	0.64
π_2	Original (π_{21})	2300	1721	9	1.09	1.32	1.8	Dominated by π_{14}	0.86
	Resource Increase (π_{22})	2350	2076	5	1.11	1.59	1.0	Not Dominated	0.60
	Space Changed (π_{23})	2100	1553	8	1.0	1.19	1.6	Not Dominated	0.63
	Speed Increase (π_{24})	2150	1830	7	1.02	1.4	1.4	Dominated by π_{12}	0.57
π_3	Original (π_{31})	2500	1300	14	1.19	1.0	2.8	Not Dominated	1.81
	Resource Increase (π_{32})	2450	1592	7	1.16	1.22	1.4	Not Dominated	0.48
	Space Changed (π_{33})	2200	1348	12	1.04	1.03	2.4	Not Dominated	1.40
	Speed Increase (π_{34})	2300	1466	10	1.09	1.12	2.0	Dominated by π_{13}	1.01
	Y^{min}	2100	1300	5	1	1	1	Φ =median()=1.19; Selected Plan:π_{13}	

(π_{*4}). Speed and resource based plans are similar to the above description while space changed plans are achieved by changing the starting and goal locations of the equipment, changing the positions of cranes and by relocating the temporary buildings (fabrication, material storage etc.) in order to minimize safety score.

We assign the rental cost for the truck and crane as 50 and 100 per day respectively. The fixed cost varies in between 100 and 150 depending on the plan. The optimization tuples Y_π are calculated using (4.13) from which the minimum tuple Y^{min} is computed. Accordingly the attributes are normalized (Y_π^{norm}) using (4.15) and the difference ($Y_\pi^{norm} - Y^{min}$) is calculated from the minimum normalized tuple $Y^{min} = (1, 1, \ldots, 1)$. After discarding all the dominated plans, we have the remaining plans π_{12}, π_{13}, π_{22}, π_{23}, π_{31}, π_{32}, π_{33}. Among those, only the plans, π_{13}, π_{31} and π_{33} are candidate optimal plans according to (4.16) as these plans have the safety scores smaller than the median safety ($\Phi = 1.19$). Finally we select plan π_{13} that is the closest to the minimum among the three plans.

4.11.6 Managerial Implications and Discussions

The methodologies and case studies described above guide planning managers through choosing a suitable plan. Our system presents graphical heatmaps (such as in Fig-

ure 4.11) that enable practitioners to virtually realize the scenarios of the real plan execution. From Figures 4.12(a) and (b) we can conclude that the sequential plan, π_3 has low safety variation while the other two plans take less time to finish. In Figure 4.12(c), we observed that the increased resource raises the fixed cost per day ($l_F \uparrow$) as more workers and other resources are required to operate additional equipment. Therefore, the planning manager can set a threshold safety score r_π^{th} to prevent excessive increase of resources and compute the maximum number of resources that keeps the safety score under the allowed level ($r_\pi \leq r_\pi^{th}$). The same conclusion can be drawn for speed increase (see Figure 4.12(d)) which essentially raises the safety score by adding more chances of collision and also increases fixed cost (more material). Therefore, a threshold similar to resource increase is used in order to get the maximum allowed speed that keeps the safety score under the allowed limit.

Finally, a detailed analysis similar to Table 4.1 helps managers select a plan out of all possible alternate plans. This multi-objective optimization model also guides the planning managers to choose a slightly lesser safe plan, if this results in significant improvement to the other attributes (e.g project duration, cost).

4.12 Discussions and Future Work

In this chapter, we developed an easily implementable methodology for ex ante analysis of construction plans in terms of their safety hazards to minimize the risk of struck-by accidents in construction jobsites. Given an initial activity graph, our model extracts different sequences of activities, converts them to discrete event models and simulates them using discrete event scheduler algorithms. Motion planning methodologies generate the collision free trajectories for the moving bodies

and workers. An ex ante simulation and proactive safety visualization is provided during pre-planning phase using heatmaps and sensitivity analysis which effectively distinguish among the safe and dangerous places in a construction site.

The formalism presented in this chapter provides measurement metrics to construction project managers, such as quantified safety scores, cost and time spent by a construction plan. Based on these measures, the best plan and guidelines for workers can be calculated in a construction site.

One immediate extension of our work is to take into account the stochastic nature of construction jobsites. We assumed that the motions performed by the moving obstacles were deterministic, so in the future we plan to incorporate models that include bounded and probabilistic uncertainty. Another extension is to incorporate the movement of equipment in 3D to investigate possible collision states.

These results provide valuable information for project managers to evaluate construction plans in terms of their safety performance during the planning phase. In addition, the results could be used during the project execution for training workers and equipment operators with regards to hazardous zones and the corresponding safety policies. We would like to closely study deployments linked with real-time monitoring of construction activities to evaluate how likely it is for a worker to follow a suggested plan and what alternate action spaces for workers can be used.

In this paper, we evaluated two commonly performed construction tasks: excavation and concrete pouring. We will extend this work to evaluate our methodology using information for larger construction projects involving different activities with large equipment fleets and a large number of workers.

CHAPTER 5

COMMUNICATION PRESERVING MULTI-OPTIMAL MOTION PLANNING

Motion planning for an autonomous system, at its simplest, involves finding a trajectory that avoids obstacles and respects differential constraints [BL89a, LaV06a]. An objective function, such as travel time or length, will then be optimized [KF11]. However, for many real-world applications, multiple objectives may be relevant. It might be desirable to conserve fuel, provide a comfortable ride, and avoid locations with a high risk for accidents. Trade-offs are likely to be involved when optimizing such disparate objectives. The objectives can be combined using a weighted function, but the appropriate weights may not be known a priori. Additionally, simple multi-objective combinatorial problems such as a 2-criteria shortest path are proven to be NP-Complete [Ehr00]. Therefore, in this chapter, we propose sampling-based motion planning algorithms to generate a single path that best optimizes multiple cost functions. The methodologies of this chapter have been partially published previously and can be found in [RBRb].

5.1 Visibility as an Objective Function and Motivation

In some contexts like military operations, visibility becomes an important aspect of the objective. One typically wishes to maintain visibility with friendly units or targets of observation, while avoiding visibility by potential enemies. One such sample environment is shown in Figure 5.1 where a vehicle needs to monitor two blue units and avoid one enemy firing range throughout its traveling path from the purple starting location to the yellow goal area which also has to be the shortest path. Unless a weighted objective function can be specified for such a mission, it may be necessary to sample the problem space to obtain estimates for the weight of

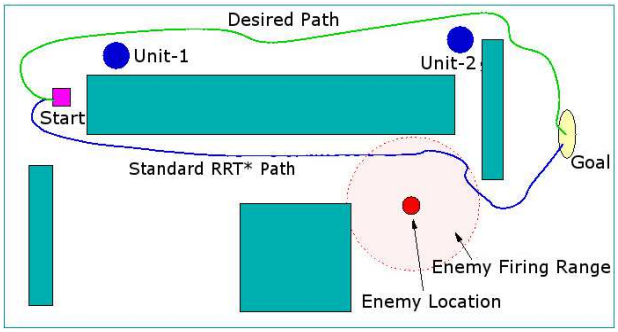


Figure 5.1: A Dubins vehicle is assigned to observe two blue circular units while avoiding obstacles and an enemy unit throughout its path from the start to the goal location. The path also needs to be shorter and a multi-criteria optimization path like the green trajectory is required. Existing sampling-based path planning may give an incorrect path like the blue one.

each objective. Integrating this process into the motion planning algorithm makes it possible to attempt an optimization of all of the objectives simultaneously. This chapter presents such a method and analyzes its applicability to certain multiple-objective motion planning problems.

Our work is motivated by the problem posed in [RRPS14] that requires one to determine the positions of a group of *units* that need to perform surveillance over a group of *targets* while simultaneously minimizing exposure to *enemies*. In [RRPS14], the units, targets, and enemies are static. The problem is formulated as a multi-objective correlated geometric optimization problem and it is solved through Markov chain Monte Carlo methods. Our goal is to extend this family of problems by allowing the units, targets, and enemies to move in an environment with obstacles while also attempting to optimize other variables such as completion time, clearance from obstacles, and communication maintenance. We will frame this family of problems as multi-objective optimal path planning problems.

The rest of the chapter is organized as follows: Section 5.2 presents a discussion about existing solutions along with their usefulness and shortcomings in the context

of our problem. Section 5.3 presents the preliminaries and the problem formulation. Section 5.4 to 5.9 introduces algorithms to solve the class of multi-objective path planning problems. We then analyze the complexity and behavior of the proposed methods in Section 5.10. Section 5.11 presents illustrative case studies of several field missions. Finally, conclusions and directions for further research are discussed in Section 5.12.

5.2 Related Work

Multi-objective optimization has been studied widely for many years in different domains. A solution for a multi-objective problem can be based on scalarization of objectives where the objectives are weighted according to their priority and added to form a single scalar value [Tar07a]. Related to our ideas are the methods proposed in [Tar07a] for multi-criteria shortest path computations that compute a number of possible paths from source to goal and then choose a Pareto efficient path [War87]. Although our methods are initially motivated by the techniques presented in [Tar07a], we propose a solution that works in incrementally building rapidly exploring random trees, such as RRT* [KWP+11b, KF11]. We are also focused on generating a single path optimizing all of the objectives.

Our ideas are also connected with the method described in [YGS15] that modifies the RRT* algorithm [KWP+11b] in order to adopt multiple criteria during expansion. In contrast with this approach our algorithm is able to produce a single path in terms of multiple costs rather than a number of Pareto optimal paths. Additionally, the weights in the *Tchebycheff* method and the weighted sum method [ZL07] used in [YGS15] can be difficult to tune as different objectives have different costs.

Instead, our work solves the multi-criteria optimization problem by normalizing the objectives using the *Utopian* optimal vector [ZL07].

Closely related to our work is [OAJRK14] where the scalarization of objectives was also used to get a single objective function. This work starts with a known graph and uses A^* search. A modified Bellman-Ford method is used in [DFS+92] to assign a normalized label to each node in order to find a multi-criteria shortest path. Another solution that prioritizes one objective over another is presented in [Fuj96]. This type of hierarchization biases the path mostly towards the top priority objective. Also, this solution is limited to a 2D grid and cannot be applied in a higher dimensional configuration space.

Our work has commonalities with [DW09] as we also assign multiple labels to each node in an RRT* tree. However, since we do not know the nodes and edges beforehand, the node reduction and edge pruning applied in [DW09] cannot be used directly in our algorithm.

Another stream of research proposes visibility-based solutions to monitor a number of units in an environment. A modified Traveling Salesman (TSP) algorithm was used by [OOD12] where the problem was solved without optimizing multiple criteria. Similarly in [GCB06], a vector field was generated to guide a robot that maximizes visibility regardless of the path length.

Related to our ideas is the pursuit evasion problem discussed in [KF10a] where multiple RRT* [KF11] trees were used, one for each unit, and the evader's tree was carefully expanded in order to avoid pursuers. We extend this idea and apply a modified version of the RRT* algorithm to avoid adversarial units using visibility and multiple objectives.

5.3 Preliminaries

Consider an environment $\mathcal{W} = \mathbb{R}^2$ where a mission is taking place. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_\zeta\}$ be the set of obstacles which are modeled as polygons. The collision-free space is defined as $E = \mathcal{W} \setminus \mathcal{O}$.

Let $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ be a number of vehicles which are deployed, with configuration spaces $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, respectively. The servicing vehicles move inside the bounded environment E as car-like robots. Therefore the configuration of each vehicle A_i is defined as $\mathcal{C}_i = E \times S^1$. These vehicles are like Dubins cars [Dub57], and a given vehicle A_i must satisfy differential constraints and dynamics defined as $\dot{x}_i = u_s^i \cos\theta, \dot{y}_i = u_s^i \sin\theta$, and $\dot{\theta}_i = \frac{u_s^i}{L^i} \tan u_\phi^i$, where u_s^i is the forward speed and u_ϕ^i is the steering angle of the vehicle [GV09]. There are a number of mobile units deployed in E that can move freely in the world and are modeled as point robots without rotation. Accordingly, the configuration for a mobile unit is defined as, $B_i = (x, y) \in E$.

Let X be the state space for servicing vehicles and for simplicity assume $X = \mathcal{C}$. There are n cost functions $l_1, l_2, \ldots, l_n$ where $l_i : X \to \mathbb{R}^{\geq 0}$. Each state $x \in X$ of the robot is associated with multiple objective costs. Accordingly, a vector valued function $L : X \to \mathbb{R}^n$ assigns n cost labels to a particular state x and is defined as,

$$L(x) = (l_1(x),\ l_2(x),\ \ldots,\ l_n(x)) \text{ where } x \in X \tag{5.1}$$

Let $X_{obs} = \{x \in X : x \cap O \neq \emptyset \ where\ O \in \mathcal{O}\}$ be the obstacle state space. The collision-free state space is then $X_{free} = X \setminus X_{obs}$. We define the initial configuration state of the vehicle as, $x_I \in X_{free}$, and a set of goal states as $X_G \subset X_{free}$.

Let σ be an obstacle-free feasible trajectory that starts from x_I and leads a vehicle to its goal region X_G.

Problem 1: Generation of multi-cost optimal paths for servicing vehicles.

Given an initial configuration x_I and a set of goal states X_G, find a collision-free continuous trajectory, $\sigma : [0, t] \rightarrow X_{free}$ for some time t, such that $\sigma(0) = x_I$ and $\sigma(t) \subset X_G$ attempts to minimize L.

We are also interested in finding paths in adversarial environments. In this subclass of multi-objective optimization, we study the problems of a unit required to avoid enemies and reach a goal location safely. This problem is related to pursuit evasion games [CHI11] and non-cooperative game theory [BO99]. We have a set of enemies $e_1(t), e_2(t), \ldots, e_\psi(t)$ at time t, each of which will have a visibility range $V(e_i(t))$.

Problem 2: Generation of multi-cost optimal paths for a servicing vehicle avoiding adversarial objects.

Given an initial configuration for a vehicle, x_I, and a set of enemy positions $e_1(t), e_2(t), \ldots, e_\psi(t)$ at time t, find a collision-free continuous trajectory, $\sigma : [0, t] \rightarrow X_{free}$, that solves Problem 1 and $\sigma(t) \cap V(e_i(t)) = \emptyset$ for all enemies e_i.

For other scenarios, we will have a number of vehicles deployed inside the environment which have a common objective and perform cooperative behavior. An example of such behavior is maintaining visibility for communication.

Problem 3: Generation of multi-objective optimal paths for cooperative robots.

Given a set of friendly robot vehicles $A_1, A_2, \ldots, A_k$, with a common cost l_c, find a set of collision-free continuous trajectories, $\sigma_1, \sigma_2, \ldots, \sigma_k$, that solve Problem 1 and best optimize $l_c : X_1 \times X_2 \times \cdots \times X_k \rightarrow \mathbb{R}^{\geq 0}$.

5.4 Traditional RRT* and Multiobjective Costs

Our ideas for multi-objective optimal motion planning are modifications to the algorithms proposed by Karaman and Frazzoli [KF11, KF10a]. Algorithm 11 presents a procedure that computes a trajectory σ from x_I to X_G based on the sampling algorithm RRT* [KF11]. The structure of the algorithm is similar to RRT* where we start a tree structure $\mathcal{T}$ (line 1) and continue to expand it by sampling random states. The resulting RRT* tree $\mathcal{T}$ will be used to get a single trajectory σ that attempts to optimize multiple cost functions.

Algorithm 11 MultiObjectiveRRTStar(x_{init})

1: $\mathcal{T}.init(x_{init})$
2: **for** $i \leftarrow 1$ *to* K **do**
3: $x_{rand} \leftarrow RandomConfig()$
4: $x_{nearest} \leftarrow NearestNode(\mathcal{T}, x_{rand})$
5: $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$
6: **if** $ObstacleFree(x_{new}, x_{nearest}, \mathcal{O})$ **then**
7: $x_{opt} \leftarrow ChooseParent(x_{nearest}, x_{new}, \mathcal{T})$
8: $L \leftarrow (l^a(x_{opt}) + c(x_{opt}, x_{new}), \frac{l^{na}(x_{opt}) + l^{na}(x_{new})}{2})$
9: $\mathcal{T} \leftarrow InsertNode(x_{opt}, x_{new}, \mathcal{T})$
10: $\mathcal{T} \leftarrow ReWire(\mathcal{T}, x_{new})$
11: **end if**
12: **end for**
13: *return* $\mathcal{T}$

In addition to the state cost $L(x)$, we assign edge cost, $(c_1, c_2, \ldots, c_n)$ to an edge $\tilde{x}_{ij}$ that connects two successive states x_i and x_j. Here, a cost function c_i for a particular objective i is defined as, $c_i : X \times X \rightarrow \mathbb{R}^{\geq 0}$.

Additive/Non-Additive Cost: There are two types of costs assigned to the nodes in a tree. Additive costs are cumulative and the costs have parent-child dependencies (e.g. Euclidean distance and time). An additive cost depends on a parent's cost and the *arc* cost that connects it to its parent. On the other hand a non-additive cost is independent and calculated based on the state (e.g. visibility, safety, clearance).

We define the additive costs as l^a and the non-additive costs as l^{na} that will be used is subsequent sections.

The primitives of the Algorithm 11 are similar to other sampling-based motion planning algorithms [LH98b, KL00, KF10b]:

Sampling: A tree is initialized in line 1 of Algorithm 11. In line 3 the method $RandomConfig()$ samples a random configuration $x_{rand} \in X_{free}$.

Nearest Node: In line 4, $x_{nearest} = NearestNode(\mathcal{T}, x_{rand})$ returns the node $x_{nearest}$ of the tree $\mathcal{T}$ that is nearest the sampled node x_{rand} in terms of a distance metric.

Steer: The method $Steer(x_1, x_2)$ is used to solve control inputs u_s and u_ϕ and produces x_{new} from x_{rand} for a dynamic control system.

Collision Checking: A collision-free path is required to connect the sampled node x_{new} to its nearest neighbor $x_{nearest}$ in order to expand the tree $\mathcal{T}$. Method $Obstaclefree()$ checks whether a path from x_{new} to $x_{nearest}$ avoids all the obstacles $\mathcal{O}$.

Until now the above discussed methods and steps are more or less the same as the ones used in a standard RRT* algorithm. To introduce the multi-objective optimization, we completely modified the $ChooseParent()$ and $ReWire()$ methods.

5.5 Choosing a Parent

Algorithm 12 is used to select the best parent x_{opt} of the newly sampled node x_{new} in terms of a multi-objective optimization.

Domination and Non-Domination: We choose a set of candidate nodes, X_{near} based on a nearness metric (e.g. point distance in Euclidean space) using method $NearestNeighbours()$ in line 1 of Algorithm 12. Each node $x_i \in X_{near}$

has an associated cost vector $L(x_i) = (l_1(x_i), l_2(x_i) \ldots, l_n(x_i))$. However, we do not consider all of them as a potential parent. Only the nodes whose costs belongs to the *Pareto frontier* are considered. A node's cost is a member of the Pareto frontier set if it is not dominated by the costs of any other node. $L(x_i)$ dominates $L(x_j)$ if and only if all the objective costs associated with x_i are less than or equal to the objective costs associated with x_j. Node x_i is chosen over node x_j if cost $L(x_j)$ is dominated by cost $L(x_i)$ (denoted by $L_i \prec L_j$).

$$x_i \prec x_j \;\Leftrightarrow\; \forall k, l_k(x_i) \leq l_k(x_j);\; where\; 1 \leq k \leq n. \tag{5.2}$$

The set of all nodes such as $x_j \in X_{near}$ from (5.2) comprises the set of dominated nodes D_X which are dominated by some node $x_i \in X_{near}$,

$$D_X = \{x_j \in X_{near} | \exists i \; x_i \prec x_j\}. \tag{5.3}$$

Accordingly the set of non-dominated nodes P_X comprises the Pareto frontier,

$$P_X = X_{near} \setminus D_X. \tag{5.4}$$

Choose Parent: This procedure chooses a single node from the set of Pareto frontier nodes P_Q calculated from the previous step. As the essential characteristic of the nodes in the Pareto set is that none of them is unequivocally better than the others, it is difficult to select a single node. Several options are available to solve this problem. We employ a combinatorial solution that approximately solves the problem to find a feasible path.

We first calculate the minimum costs l_i^* for each objective i, among the Pareto node set P_Q. So the optimal cost tuple L^* therefore is,

$$L^* = (\min_{1 \leq i \leq |P_X|} l_1(x_i), \min_{1 \leq i \leq |P_X|} l_2(x_i), \ldots, \min_{1 \leq i \leq |P_X|} l_n(x_i)) \tag{5.5}$$

Next, we calculate the *arc* costs, $C = (c_1, c_2, \ldots, c_n)$ associated with an *arc*, $\tilde{x}_{j,new}$, which connects the new node x_{new} to a nearest non-dominated node $x_j \in P_X$. The minimum costs c_i^* designated to an arc for each objective i are calculated and the optimum cost tuple for arc costs is:

$$C^* = (\min_{1 \leq i \leq |P_X|} c_1(x_i, x_{new}), \ldots, \min_{1 \leq i \leq |P_X|} c_n(x_i, x_{new})) \tag{5.6}$$

A weighting variable $\alpha_i = [0, 1]$ is assigned to each objective i to control the effect of the objective cost on a path to be planned where $\sum_{1 \leq i \leq n} \alpha_i = 1$. This weighting variable is different from that used in *Tchebycheff* and weighted sum methods used in [ZL07, YGS15] and is only used to define priority. Our algorithm is capable of running without α unlike the methods described in [ZL07, YGS15] in which α is the essential part of those algorithms. Accordingly, we select a node $x_j \in P_X$ as the parent (line 7) that yields the minimum *normalized* costs to come to x_{new},

$$\underset{x_j \in P_X}{\operatorname{argmin}} \sum_{i=1}^{n} \alpha_i \left[\frac{l_i(x_j)}{l_i^*} + \frac{c_i(x_j, x_{new})}{c_i^*} \right] \tag{5.7}$$

Finally the cost $L(x_{min})$ of the current optimal node x_{min} is updated in line 9 and the algorithm terminates by returning the best parent node in terms of *normalized* multi-objective costs (line 12).

5.6 Updating the Tree

Different Costs: In line 8 of Algorithm 11, we assign a cost label L defined in (5.1) to the current node x_{new} once a parent x_{opt} is selected for that node. There are two types of costs, 1) additive and 2) non-additive. Allocating costs to a node is therefore not the same for all the objectives. *Additive* costs l^a like *distance* are allocated by combining the cost of a parent and the arc cost. Therefore this cost

Algorithm 12 ChooseParent($x_{min} \leftarrow x_{nearest}, x_{new}, \mathcal{T}$)

1: $X_{near} \leftarrow NearestNeighbours(x_{new}, \mathcal{T}, \mathcal{O})$
2: $D_X \leftarrow \{x_i \in X_{near} | \exists j \ \forall k \ l_k(x_i) \leq l_k(x_j)\}$
3: $P_X \leftarrow X_{near} \setminus D_X$
4: $L^* \leftarrow (l_1^*, \ l_2^*, \ \ldots, \ l_n^*)$
5: $C^* \leftarrow (c_1^*, \ c_2^*, \ \ldots, \ c_n^*)$
6: **for** $x \in P_X$ **do**
7: **if** $\sum_i \left[\frac{l_i(x)}{l_i^*} + \frac{c_i(x,x_{new})}{c_i^*}\right] < \sum_i \left[\frac{l_i(x_{min})}{l_i^*} + \frac{c_i(x_{min},x_{new})}{c_i^*}\right]$ **then**
8: $x_{min} \leftarrow x$
9: $L(x_{min}) \leftarrow L(x)$
10: **end if**
11: **end for**
12: *return* x_{min}

resembles the total cost from the root node to the currently extended node x_{new}. Suppose we have k additive costs and $n-k$ non-additive costs among the n objective costs. An additive cost $l_i^a(x_{new})$ is therefore computed by:

$$l_i^a(x_{new}) = l_i^a(x_{opt}) + c_i(x_{opt}, x_{new}); \forall \ i, \ 1 \leq i \leq k. \tag{5.8}$$

Non-additive costs l^{na}, like *visibility* require in-place computation and are assigned regardless of the total cost of the path from the root node. In other words, these costs do not propagate. Only the parent's cost and the current nodes cost that is computed by an in-place computation are averaged to assign and update the cost vector L.

$$l_j^{na}(x_{new}) = \frac{l_j^{na}(x_{opt}) + l_j^{na}(x_{new})}{2}; \forall \ j, \ k+1 \leq j \leq n. \tag{5.9}$$

Node Insertion: Once we know the parent x_{opt} of the newly sampled node x_{new}, we add it to the tree $\mathcal{T}$ along with the corresponding edge $\tilde{x}_{opt,new}$ and *arc* cost $c(\tilde{x})$. The modified costs $L(x_{new})$ for the involved node is also updated in this step (line 9 in Algorithm 11).

5.7 Refining Connections

Algorithm 13 is used to refine the existing connections in the neighborhood of a newly connected vertex x_{new}. This procedure makes x_{new} the parent of the neighboring nodes $x \in X_{near}$ if this yields optimum costs compared to the costs incurred through its current parent. Method $NearestNeighbours()$ in line 1 computes the nearest node set X_{near} and works the same as in Algorithm 12. A candidate neighbor $x \in X_{near}$ is connected through the newly added node x_{new} if the following two conditions are satisfied:

$$\forall\ i, 1 \leq i \leq k;\ l_i^a(x_{new}) + c(x_{new}, x) \leq l_i^a(x) \tag{5.10}$$

$$\forall\ j, k+1 \leq j \leq n;\ l_j^{na}(x_{new}) \leq l_j^{na}(x.parent) \tag{5.11}$$

The above conditions are connected to our former explanation that the additive and non-additive costs need to be evaluated separately. Firstly, in case of an additive cost $l^a(x)$, we make x_{new} the parent of the neighboring node x if the connecting cost reduces the existing cost $l^a(x)$ as shown in (5.10). Secondly, we only change the existing parent $x.parent$ to x_{new} if x_{new} provides a better cost than $x.parent$ in terms of non-additive costs $l^{na}(x)$. For example, if a node's parent has better visibility or safety than x_{new}, then making x_{new} the new parent is not desirable. Therefore satisfying both the (5.10) and (5.11) make x_{new} the new parent of x and Algorithm 13 terminates by updating the cost vector $L(x)$ and tree $\mathcal{T}$ in lines 5 and 6 respectively.

Algorithm 13 ReWire($\mathcal{T}, x_{new}$)

1: $X_{near} \leftarrow NearestNeighbours(x_{new}, \mathcal{T}, \mathcal{O})$
2: **for** $x \in X_{near}$ **do**
3: **if** $\bigwedge_{i=1}^{k}(l_i^a(x_{new}) + c(x_{new}, x) \leq l_i^a(x))$ and $\bigwedge_{j=k+1}^{n} l_j^{na}(x_{new}) \leq l_j^{na}(x.parent)$ **then**
4: $x.parent \leftarrow x_{new}$
5: $L(x) \leftarrow (l^a(x_{new}) + c(x_{new}, x), \frac{l^{na}(x_{new}) + l^{na}(x)}{2})$
6: $\mathcal{T}.update(x)$
7: **end if**
8: **end for**
9: *return* $\mathcal{T}$

5.8 Avoiding Adversaries

As the deployed robots frequently want to avoid enemies while moving through the environment, the computed paths need to maintain a safe distance from certain ranges where enemy impact is unavoidable. Additionally, a very long path may be out of the reach of enemies, but be undesirable because it is practically impossible to traverse. Therefore, we must find a path that respects multiple objectives and is safe from enemies.

We propose Algorithm 14 that avoids the adversarial units while minimizing other objectives. This solution is based on the pursuit-evasion game explained in [KF10a]. We expand two types of trees, $\mathcal{T}_v$ for the servicing vehicle and $\mathcal{T}_e$ for an enemy. $\mathcal{T}_v$ expands in such a way that it maintains a safe distance from $\mathcal{T}_e$. We assume that the enemy can fire from a certain visibility range r_{vis} and does not travel beyond a distance d_e from an initial position x_e^{init}. In line 4 of Algorithm 14, method $MultiRRT^*\_Expansion()$ is used to expand the vehicle's tree $\mathcal{T}_v$ that minimizes multiple objective functions. Method $MultiRRT^*\_Expansion()$ consists of only the expansion part of Algorithm 11 (lines $4-10$ of Algorithm 11). The $NearbyRiskNodes()$ in line 6 calculates the set of vertices X_{near} from the enemy

tree $\mathcal{T}_e$ that are in the firing range r_{vis}. An enemy present on node $x_{risk} \in X_{near}$ can potentially attack the newly sampled node x_v and therefore it is removed at line 9.

Algorithm 14 RRTStarAdversary(r_{vis}, d_e)

```
 1: i ← 0
 2: while i ≤ K do
 3:    x_v ← RandomConfig()
 4:    T_v ← MultiRRT*_Expansion(T_v, x_v)
 5:    if x_v ≠ NULL then
 6:       X_near ← NearbyRiskNodes(T_e, x_v, r_vis)
 7:       for x_risk ∈ X_near do
 8:          if ObstacleFree(x_v, x_risk) then
 9:             Remove(T_v, x_v)
10:          end if
11:       end for
12:    end if
13:    x_e ← RandomConfig()
14:    T_e ← RRT*_Expansion(T_e, x_e)
15:    if x_e ≠ NULL and dist(x_e, x_e^init) > d_e then
16:       Remove(T_e, x_e)
17:    else
18:       X_near ← NearbyRiskNodes(T_v, x_e, r_vis)
19:       for x_risk ∈ X_near do
20:          if ObstacleFree(x_e, x_risk, O) then
21:             Remove(T_v, x_risk)
22:          end if
23:       end for
24:    end if
25: end while
```

On the other hand, an enemy tree $\mathcal{T}_e$ is grown by sampling new nodes x_e in lines $13 - 14$ using a regular RRT* tree expansion [KF10a]. We only retain the nodes that are inside the allowed enemy patrolling range d_e (lines $15 - 16$). The vertices X_{near} from the vehicle tree $\mathcal{T}_v$ that have the potential risk of attack from the newly sampled enemy node x_e are extracted at line 18. All such nodes are deleted from $\mathcal{T}_v$ if they are not blocked by any obstacle in $\mathcal{O}$.

5.9 Cooperative Path Generation for Multiple Robots

An opposite scenario of an adversarial situation is a cooperative one where multiple friendly units want to communicate while optimizing their own objectives. One particular case of cooperative path planning is where two or more vehicles want to maintain visibility with each other. We propose Algorithm 15 where two cooperative trees $\mathcal{T}_u$ and $\mathcal{T}_v$ expand in parallel while affecting each other. Both vehicles have multiple objectives to satisfy and accordingly lines $4-10$ of Algorithm 11 are used for their expansion in lines 4 and 6 of Algorithm 15. A function $l_c : X \times X \to \{0, 1\}$ is defined that checks whether the two newly sampled vertices x_u, x_v of the two trees cooperate (in line 8). A reward function $\omega_k : X \times X \to \mathbb{R}^{\geq 0}$ is defined that helps to decrease the costs for objective k if the two nodes cooperate (lines $9-10$). Otherwise, a penalty function $\rho_k : X \times X \to \mathbb{R}^{\geq 0}$ is used to increase each of the costs (lines $12-13$). Finally, $PropagateCost()$ is used in lines $15-16$ to pass the effect of the updated cost down towards all the nodes throughout the child chain.

5.10 Analysis

The algorithm proposed here is based on RRT* [KF11, KF10a], so most of its properties are directly inherited.

Running Time Analysis: The main modification in our proposed model is implemented in Algorithms 12 and 13. In order to take care of n objective costs, both of them run n times more than the standard RRT* algorithm. Line 2 and 7 in the Algorithm 12 and lines 3 and 5 in Algorithm 13 take $O(n)$ time to check and update n costs. Therefore the running time of our multi-objective RRT* algorithm is $O(n \cdot RRT^*)$, which is a constant multiple of the running time of the standard RRT*.

Algorithm 15 RRTStarCooperative(x_u^{init}, x_v^{init})

1: $i \leftarrow 0$
2: **while** $i \leq K$ **do**
3: $x_u \leftarrow RandomConfig()$
4: $\mathcal{T}_u \leftarrow MultiRRT^*\_Expansion(\mathcal{T}_u, x_u)$
5: $x_v \leftarrow RandomConfig()$
6: $\mathcal{T}_v \leftarrow MultiRRT^*\_Expansion(\mathcal{T}_v, x_v)$
7: **if** $x_u \neq NULL$ and $x_v \neq NULL$ **then**
8: **if** $l_c(x_u, x_v)$ **then**
9: $\forall k, 1 \leq n, l_k(x_u) \leftarrow l_k(x_u) - \omega_k(x_u, x_v)$
10: $\forall k, 1 \leq n, l_k(x_v) \leftarrow l_k(x_v) - \omega_k(x_u, x_v)$
11: **else**
12: $\forall k, 1 \leq n, l_k(x_u) \leftarrow l_k(x_u) + \rho_k(x_u, x_v)$
13: $\forall k, 1 \leq n, l_k(x_v) \leftarrow l_k(x_v) + \rho_k(x_u, x_v)$
14: **end if**
15: $PropagateCost(x_u, \mathcal{T}_u)$
16: $PropagateCost(x_v, \mathcal{T}_v)$
17: **end if**
18: **end while**

We now analyze whether our algorithm chooses the right nodes during expansion in terms of multiple cost optimization.

Proposition 5.10.1 *ChooseParent() selects a non-dominated optimal parent.*

Proof. (sketch) It is trivial that we select a non-dominated node as a parent as we select it from the non-dominated set P_X according to line 6 of Algorithm 12.

We now prove the optimality by contradiction. Let $ChooseParent()$ select x_p as a parent of a node x_{new} which does not provide the optimal costs and let there be a parent x'_p that provides the optimal costs such that:

$$\sum_{i=1}^{n} \left[\frac{l_i(x'_p)}{l_i^*} + \frac{c_i(x'_p, x_{new})}{c_i^*} \right] \leq \sum_{i=1}^{n} \left[\frac{l_i(x_p)}{l_i^*} + \frac{c_i(x_p, x_{new})}{c_i^*} \right] \tag{5.12}$$

This contradicts the fact in (5.7) where we select the node as a parent which minimizes the above cost. This condition is employed in line 7 of Algorithm 12. Therefore, x_p is chosen over x'_p implies that x_p and x'_p cannot be different. This essentially

proves that the method $ChooseParent()$ selects the optimal non-dominated parent in terms of multi-objective cost. □

Proposition 5.10.2 *$ReWire()$ selects the optimal parent in terms of multiple cost vector L in a particular tree $\mathcal{T}$.*

Proof. (sketch) This is trivial from the conditions in (5.10) and (5.11) where the current parent of a node $x \in X_{near}$ is changed to the newly sampled node x_{new} if and only if x_{new} is better in all n cost metrics l_k where $1 \leq k \leq n$. See line 3 of Algorithm 13. □

Proposition 5.10.3 *A solution path σ is a non-dominated solution for a particular MultiObjectiveRRT* tree $\mathcal{T}$.*

Proof. (sketch) The proposed MultiObjectiveRRT* is a modification of RRT* which guarantees asymptotically an optimal path. It is a necessary condition that a sub-path of an optimal path is also optimal. From propositions 5.10.1 and 5.10.2, we guarantee that the local sub-solutions are non-dominated. This implies that once a tree $\mathcal{T}$ is generated, the corresponding path σ is a non-dominated path. □

5.11 Case Studies

We developed an implementation on top of the MIT SMP library [KFa] that was originally developed by the authors of RRT*. The core functionality of the available code was modified in order to support multiple cost vectors. Afterwards a Python visualization was employed to present the raw output extracted from the modified SMP library.

5.11.1 Problem Modeling

We modeled a problem based on Figure 5.1 where a number of point robots $B_1, B_2, \ldots, B_m$ must be served through visible light communication [YBZ13]. This means that the units must come into the line of sight (LoS) of the serving module. There are a number of serving vehicles (Dubins cars) $A_1, A_2, \ldots, A_k$ that are assigned to provide support to these units through patrolling. An optimal trajectory σ must maintain visibility while optimizing the traveling distance from the starting location (purple) to the goal location (yellow). Additionally, there might be a number of friendly units we want to observe and a number of unfriendly units we want to avoid.

5.11.2 Case Study 1: Single Unit Visibility and Patrolling

In Figure 5.2, we present a case where a single vehicle is present to serve the blue unit. The environment contains an obstacle $\mathcal{O}$ at the middle of the map. We used C++ code to calculate a serving trajectory that minimizes the traveling distance and maximizes LoS visibility to the units.

Figure 5.2(a) and (c) are the resulting tree and trajectory (red) that are computed by the standard RRT* [KF11] algorithm after 500 and 3000 iterations respectively. We then apply our algorithm and the results are shown in Figure 5.2(b) and (d) for 500 and 3000 iterations respectively. Clearly the trajectory of Figure 5.2(b) is better than Figure 5.2(a) as it goes close to the unit before reaching to the goal location. Most parts of the trajectory σ in Figure 5.2(a) is obstructed by the obstacle $\mathcal{O}$ which is undesirable (see Figure 5.1). Similarly the path converges to optimality in terms of length as shown in Figure 5.2(c), but it has poor visibility. Finally, our algorithm converges to a better trajectory after 3000 iterations as shown

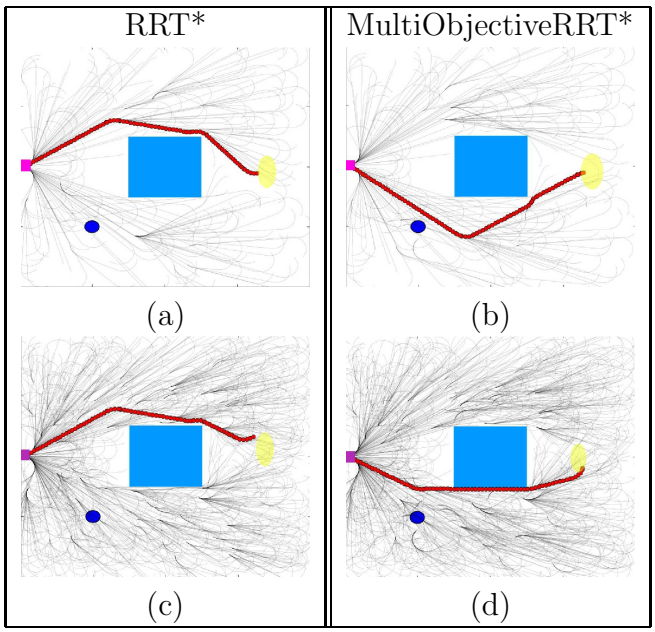


Figure 5.2: Trajectory finding for a car-like vehicle while monitoring the blue circular landmark: (a) Standard RRT* tree and trajectory after 500 iterations. The purple rectangle is the initial position and the yellow region is the goal. (b) Our MultiObjectiveRRT* tree and trajectory after 500 iterations. (c) Standard RRT* tree after 3000 iterations. (d) Our MultiObjectiveRRT* tree after 3000 iterations.

in Figure 5.2(d). Along with visibility, this path optimizes the length as compared to the longer path generated by Figure 5.2(b).

5.11.3 Case Study 2: Two Vehicles, Two Units

In Figure 5.3, we present a case where two blue units B_1 and B_2 need to be monitored by two Dubins vehicles A_1 and A_2 while reaching their respective goal regions. The calculated trajectory for A_1 is green and A_2 is red. Figure5.3(a), (c) and (e) are the results from our algorithm and Figure 5.3(b), (d) and (f) are the outcomes of the weighted sum and *Tchebycheff* methods used in the state of art [ZL07, YGS15] solutions.

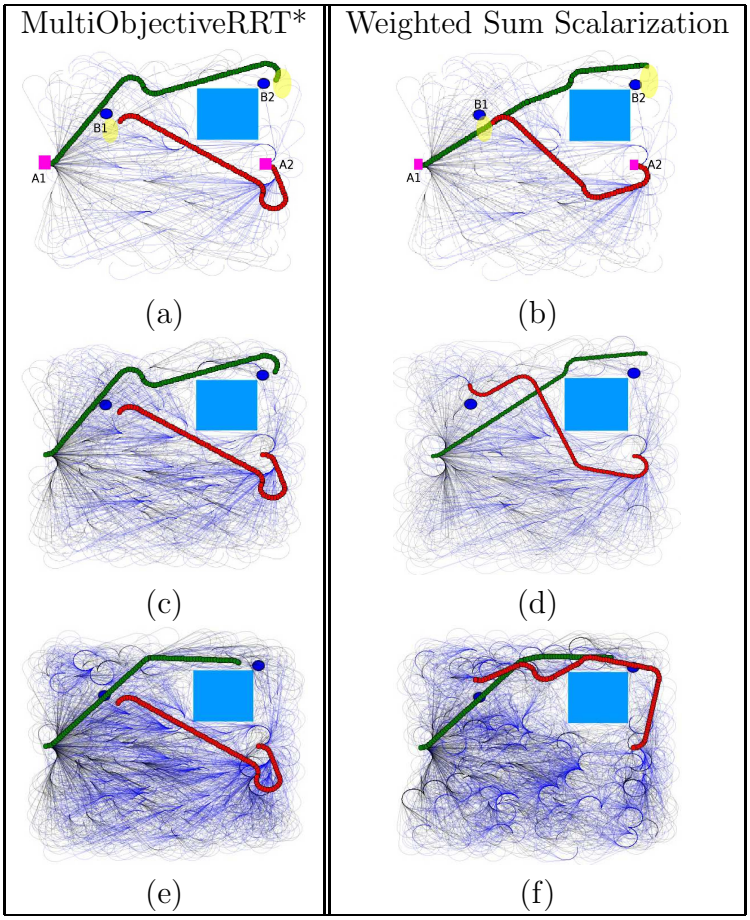


Figure 5.3: Dubins car trajectory finding for two car-like robot. Vehicles start from two small rectangular positions (purple colored): (a) Our MultiObjectiveRRT* tree and trajectory after 500 iterations. (b) RRT* tree with weighted sum (scalarization) method after 500 iterations. (c) Our Multi RRT* at 2000 iterations.(d) Tchebycheff (scalarization) method after 2000 iterations. (e) Our Multi RRT* after 5000 iterations.(f) Tchebycheff (scalarization) method after 5000 iterations.

The path for A_1 (green) goes upwards around the unit B_1 and turns towards the unit B_2 to maximize visibility while minimizing path length after 500 iterations as shown in Figure 5.3(a). Similarly A_2's path (red) makes a turn to maximize visibility and then follows the optimal distant path. With the presence of the rectangular obstacle, it is not possible to provide maximum service to both units B_1 and B_2 throughout the path while minimizing the traveling distance. Therefore, A_2 makes

a circular turn to stay in B_2's visibility range a little more before serving B_1 through a shortest distance path.

The weighed sum and Tchebycheff [ZL07, YGS15] methods both use scalarization of objectives and show similar characteristics after 500 iterations as shown in Figure 5.3(b). Although we tried to select the best weights for the objectives, these methods frequently become biased towards a particular objective. The trajectory generated for A_1 optimizes path length over visibility and the trajectory for A_2 is mostly out of the visibility range of unit B_2. In Table 5.1, we provide a numerical comparison of our method with these prevalent techniques. We found that the weighed sum method generates a slightly shorter path (98 vs 111) than our method, while the difference of visibility (0.62 vs 0.46) is larger (a lower value means better visibility).

In Figure 5.3(c), we present the tree using our MultiObjectiveRRT* after 2000 iterations. Here our method generates a similar tree and trajectories as in Figure 5.3(a). However, the weighted sum method modifies its trajectory (specifically the red one) which becomes longer with a slightly increased visibility as presented in Table 5.1.

Finally we ran the methods for 5000 iterations as shown in Figures 5.3(e) and (f). Our method converged to a near optimal non-dominated solution. On the other hand, the Tchebycheff method generated the green trajectory with a slightly increased visibility (0.47 vs 0.53) and a longer path (91 vs 88). The red trajectory generated by our method is very short compared to the trajectory generated by the Tchebycheff method (96 vs 117) with very good visibility cost (0.58). Therefore, we conclude that our method provides a better compromise solution to all the objectives than the weighted sum method which frequently biases towards a particular objective.

Table 5.1: Trajectory Analysis in Terms of Multiple Objectives

	Iteration	Objective	Tchebycheff	Our Multi RRT*
$Vehicle$ 1	500	Visibility	0.62	0.46
		Distance	98	111
	2000	Visibility	0.81	0.46
		Distance	106	111
	5000	Visibility	0.47	0.53
		Distance	91	88
$Vehicle$ 2	500	Visibility	0.80	0.58
		Distance	83	96
	2000	Visibility	0.80	0.58
		Distance	103	96
	5000	Visibility	0.55	0.58
		Distance	117	96

One important property regarding the convergence of the tree can be observed in Figures 5.3(a), (c) and (e) where we see the trajectories are converging very fast and the reconstructions of the paths do not produce any abrupt change in the tree/trajectory structures. This is because the probability of a node to be minimum in terms of multiple costs is less than the single cost which slows down the tree re-connection.

5.11.4 Case Study 3: Adversarial Environment

In Figure 5.4 we present a case with one vehicle, two units and two adversaries. The vehicle starts from the bottom area intending to reach the top yellow region while serving two blue units. Two other adversaries are present who grow the red trees while our vehicle grows the green tree following Algorithm 14. The red trees have a certain range of growth d_e and a fixed distance to attack r_{vis}. Therefore the green tree keeps a safe distance while growing and finally finds a trajectory while maximizing its visibility of the two units.

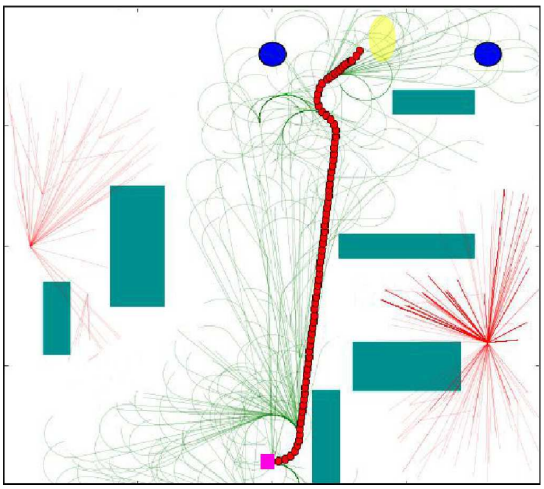


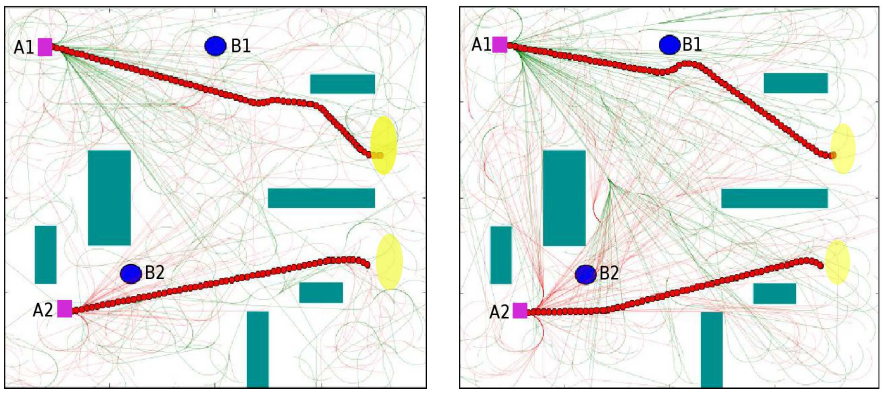
Figure 5.4: Path planning for a vehicle surrounded by two enemies. The objective is not only to avoid enemy's visibility range but also to serve the blue units and reach the yellow goal region safely.

5.11.5 Case Study 4: Cooperative Motion Planning

A case is presented with two vehicles A_1, A_2 and two units B_1, B_2 in Figure 5.5. A_1 is assigned to monitor B_1 and A_2 is assigned to monitor B_2. Additionally, an extra cooperative cost $l_c(x_1, x_2)$ is assigned that allows a reward ω to the costs of A_1 and A_2 when they are visible to each other. Otherwise it incurs a penalty ρ on the costs of the states x_1, x_2 (See lines $8-14$ of Algorithm 15). In Figure 5.5(a), we see that the paths of the two vehicles attract each other while keeping visibility to their respective blue units. In contrast, Figure 5.5(b) is the outcome of MultiObjectiveRRT* where the vehicles only keep proximity to their assigned blue units and finish the calculated paths without cooperation.

5.12 Summary

In this chapter, we formulated the problem of a group of units that need to monitor a group of targets in a contested environment as a multi-objective optimal motion



(a) (b)

Figure 5.5: (a) Cooperative path generation using MultiObjectiveRRT* Algorithm; (b) MultiObjectiveRRT* path generation without cooperation.

planning problem. We presented modifications to optimal sampling-based planning algorithms to include multiple objectives and non-additive costs. Additionally, we proposed algorithms that can handle both adversarial and cooperative missions based on the ideas in [KF10a]. We found that our proposed system can generate better paths than the weighted sum and Tchebycheff model for certain types of motion planning problems. Our study of avoiding adversarial objects in an environment was able to generate safe paths while serving friendly units. Several interesting directions are left for future work.

Multi-optimality problems in motion planning appear naturally in several practical domains. The modifications of algorithm 11 should work for other motion planning problems. An immediate goal would be testing the performance of the multi-objective addition to RRT* on benchmark problems in manipulation of an articulated robot body to see its performance.

We also want to extend the possible set of multi-objective missions in contested environments. Simple extensions will include modeling moving units as unmanned aerial vehicles (UAVs) that want to maintain a connected visibility network. Fol-

lowing such units to cover them makes the problem more complex where a tuning among velocity, safety, and monitoring is required. We believe that the proposed system can be a useful aid to calculate a feasible solution in these complex scenarios.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

In this thesis, we have studied the complexities of optimal relay placement problems, re-planning of relay robots in LoS based systems, safety quantification of a robotic construction job and robotic path planning for the multi-objective optimization problems. We have proposed robotic motion planning and autonomous system based solutions that are capable of dealing with most variations of the problems.

In the relay placement problem, where a chain formation of the relay robots is required, we have developed a layered graph from a discretization of the world model. The number of layers in the graph is equivalent to the maximum available relays. We propose a modified breadth first search algorithm to create a tree from the graph, rooted at the operator node and this tree is used as a communication map. The tree building algorithm is proven to be polynomial and only needs to run once for a fixed operator position. Afterward, in the cases of relocation of the remote unit, we only need to extract a new plan from the tree instead of re-computing the entire plan that was proposed in the best-known solution found in literature. Accordingly, a comparison table of running times between our method and the existing solution has been provided in Chapter 2 to show the improvement. We have also proposed a cost optimal min-arborescence tree computation algorithm, in cases when we need to serve multiple remote units. Such a tree spans over the operator, remote units, and intermediate relays and guarantees that the total communication cost is minimal.

We have tested our ideas using a custom communication cost function which includes the effects of common signal barriers such as building, obstacle, and terrain. Future research should target of testing our system for different communication modalities using the real world cost functions. We also developed a prototype hardware system using the mobile servo robots in an indoor grid environment. For

further extension, we suggest to use other vehicles in an outdoor setup to test the impact of different motion dynamics on signal strength.

In Chapter 3, we study the problem of setting up a visibility based fully connected network among several mobile units and autonomous robot relays. The robot relays act as servicing vehicles as they are able to provide coverage, computational power, and command to the remote units. Each of the nodes in the system, whether servicing vehicle or mobile unit must be in the visibility polygon of another servicing vehicle in order to establish a fully operational LoS based relay network. Accordingly, we propose two different polynomials-time algorithms, one centralized and one distributed to check whether the current setup is communication-valid or not. In the centralized algorithm, we have used algebraic graph theory technique and computed Laplacian matrix to check its second smallest eigenvalue that indicates the connectivity status of a graph. Our proposed distributed algorithm relies on a message passing system where any servicing vehicle can initiate a query about the system's visibility based connectivity status. Then other vehicles respond with their coverage information and the requester vehicle compiles the messages to get the status of the network.

Many complex and dynamic systems where the mobile units are frequently in motion causes regular disconnection. Such cases may not be solved by a single vehicle relocation and we require a new setup of the servicing vehicles. We have found that the optimal placements of the relay vehicles are NP-Hard by relating it to well known TSP with neighbor problem. Therefore, we decompose the environment using the visibility polygon intersections of the mobile units and apply the greedy set cover algorithm to extract the minimal number of polygons that collectively cover all the units. Placing one vehicle per polygon and one vehicle for patrolling among the polygons solve the problem approximately. Finally, we have simulated

this approach in ROS and Gazebo environment and conduct experiments on an outdoor setup using a remote controlled Rover (a modified RC car).

An interesting extension of the proposed solution is to analyze whether the solution can be improved further that is currently being approximated with an $O(\log n)$ ratio at best. Secondly, the assumption about a known world may not be the case when a mission is conducted in an unknown environment. In such cases, an explicit control algorithm is required, which guides the vehicles to keep continuous visibility in any event of movement. However, recovering a mobile unit in an unknown environment is still a challenge. Therefore, several ideas from gap navigation approaches may be helpful in such cases where the vehicles navigate towards the shadow regions [LaV06b] in the environment, or follow the units that are on the move. Also, a feedback based planner can be used to collect the information about the units that intend to go out of sight. Then an information space can be utilized to take actions in subsequent time steps of the mission.

Chapter 4 solves the problem of communication aware safe project planning where we investigate critical safety issues of a building construction jobsite. Here we design an automated planning model that optimizes different attributes such as safety, duration, cost of a project. Therefore, we propose a simulation tool based on time-driven Discrete Event Simulation (DEVS) methodologies that enables the planning managers to investigate the safety metric of a selected construction plan. This model also guides the managers with alternate approaches of a selected plan in order to minimize potential hazards. Generally, a construction plan is given as a CPM graph format which is a precedence constrained directed graph. We use all possible topological sorting algorithm to extract different and equivalent alternate plans from this graph. Each of the activities in a particular plan is then simulated

using our activity scheduler. Also, the atomic events in an activity is simulated using the DEVS-based event scheduler algorithms.

Motion planning algorithms come into effect during the planning of trajectories for workers and equipment. Generalized Voronoi Diagram as a maximum clearance roadmap has been created for the workers' waypoints while sampling-based motion planners generate the trajectories for heavy equipment. A space-time coordination system is then used to avoid collisions among the human workers and the equipment by employing the STOP and MOVE actions.

An immediate future direction of research is to incorporate the stochastic nature of the workplace instead of our deterministic model. Therefore, the positions of the workers and the equipment can be estimated using a probability density function and a number of possible time intervals can be extracted when the probability of collision is calculated to be significantly higher.

The proposed system can also be used for employee training in other complex workplaces such as manufacturing and product assembly line. These automated systems need safe colocation of human and robots and a proper sequence of activity planning is required to maximize safety. As we have shown that the safety can be maximized without incurring significant cost and delay, our system can easily be used for other industrial automation projects to generate a better work plan.

Finally, in Chapter 5, we have developed a sampling-based motion planning algorithm (based on RRT*) that optimizes multiple objectives instead of a single one, compared to the conventional motion planners in literature. Sampling-based planners expand on the free space by randomly selecting configurations and connecting them with the existing tree based on a single cost metric. Instead, we use a collection of independent cost functions in the form of a cost vector and normalize them during tree expansion (predecessor selection and successor update process). Addi-

tionally, we also propose algorithms to solve path planning problems in cooperative and non-cooperative scenarios in multi-robotic systems where a path is required for each of the robots. This is achieved by defining a mutual penalty (non-cooperative case) or reward (cooperative case) functions during individual tree expansion.

Several immediate future improvements are feasible based on the proposed modification. Our idea of multi-cost vector, cost normalization, and dominating/non-dominating classification can be used in other sampling-based motion planners such as RRT, PRM and PRM*. Another immediate extension to the work is to use gradient descent method that will try to bias the trajectories towards the specific goal functions. Also designing good reward function can help to select best parent nodes during expansion of the random tree. Furthermore, a Monte-Carlo method can provide further improvement in the proposed methodology by incorporating random variables and their statistical estimations during the tree construction process.

BIBLIOGRAPHY

[AEGPS10] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, page 51. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[AH11] Homam AlBahnassi and Amin Hammad. Near real-time motion planning and simulation of cranes in construction: Framework and system architecture. *Journal of Computing in Civil Engineering*, 26(1):54–63, 2011.

[AMP91] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proceedings of 3rd Canadian Conference on Computational Geometry*, 1991.

[BB10] S. Bhattacharya and T. Başar. Graph-theoretic approach for connectivity maintenance in mobile networks in the presence of a jammer. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3560–3565. IEEE, 2010.

[BDH+09] O. Burdakov, P. Doherty, K. Holmberg, J. Kvarnström, and P. Olsson. Positioning unmanned aerial vehicles as communication relays for surveillance tasks. In *Robotics: Science and Systems*, 2009.

[BDH+10] O. Burdakov, P. Doherty, K. Holmberg, J. Kvarnström, and P. Olsson. Relay positioning for unmanned aerial vehicle surveillance. *The international journal of robotics research*, 29(8):1069–1087, 2010.

[BF10] N. Bezzo and R. Fierro. Tethering of mobile router networks. In *IEEE Proceedings of the American Control Conference*, 2010.

[BG95] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

[BG08] Priyadarshi Bhattacharya and Marina L Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[BHEH02] R. H. Byrne, J. J. Harrington, S. E. Eskridge, and J. E. Hurtado. Cooperative system and method using mobile robots for testing a cooperative search controller, June 18 2002. US Patent 6,408,226.

[BKV10] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. Sampling-based motion planning with temporal goals. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2689–2696. IEEE, 2010.

[BL89a] J. Barraquand and J. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, pages 340–347. IEEE, 1989.

[BL89b] J. Barraquand and J. C. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *IEEE International Symposium on Intelligent Control*, 1989.

[BLS14] Carmenate T Bobadilla L, Mostafavi A and Bista S. Predictive assessment and proactive monitoring of struck by safety hazards in construction sites: An information space approach. *15th International Conference on Computing in Civil and Building Engineering*, 2014.

[BMCH07] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson. Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints. *Robotics, IEEE Transactions on*, 23(1):47–59, 2007.

[BMG+12] L. Bobadilla, F. Martinez, E. Gobst, K. Gossman, and S. M. LaValle. Controlling wild mobile robots using virtual gates and discrete transitions. In *Proceedings IEEE American Control Conference*, 2012.

[BMPC08] K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej. Using RSSI value for distance estimation in wireless sensor networks based on zigbee. In *15th IEEE International Conference on Systems, Signals and Image Processing*, 2008.

[bmw] https://techcrunch.com/2017/03/16/bmws-self-driving-car-will-aim-for-full-level-5-autonomy-by-2021/.

[BO99] T. Basar and G. Jan. Olsder. *Dynamic Noncooperative Game Theory*, volume 23. Siam, 1999.

[BTI11] D. Bhadauria, O. Tekdas, and V. Isler. Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*, 28(3):388–404, 2011.

[CA09] Jaesik Choi and Eyal Amir. Combining planning and motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, 2009.

[CHI11] T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[CKS14] D. H. Choi, S. H. Kim, and D. K. Sung. Energy-efficient maneuvering and communication of a single uav-based relay. *IEEE Transactions on Aerospace and Electronic Systems*, 50(3):2320–2327, 2014.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (2nd Ed.)*. MIT Press, Cambridge, MA, 2001.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[Cor09] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[CP07] Altannar Chinchuluun and Panos M Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50, 2007.

[CPR13] Leading causes of fatal and non-fatal injuries in construction. *http://stopconstructionfalls.com/wp-content/uploads/2013/07/Leading-Causes-of-Fatal-and-Nonfatal-Injuries-in-Construction-2013-update.pdf*, 2013.

[CT13] T. Cheng and J. Teizer. Real-time resource location data collection and visualization technology for construction safety and activity monitoring applications. *Automation in Construction*, 34:3–15, 2013.

[DCIVR06] M. Dunbabin, P. Corke, and Daniela I. Vasilescu Rus. Data muling over underwater wireless sensor networks using an autonomous underwater

vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2091–2098. IEEE, 2006.

[DD14] R. M. S. Dean and Charles A. D. Robotic collaborative technology alliance: an open architecture approach to integrated research. In *SPIE Defense Security*, pages 90960M–90960M, 2014.

[DeB10] W. M. DeBusk. Unmanned aerial vehicle systems for disaster relief: Tornado alley. In *AIAA Infotech@ Aerospace Conference, AIAA-2010-3506, Atlanta, GA*, 2010.

[DFS+92] J. S. Dyer, P. C. Fishburn, R. E. Steuer, J. Wallenius, and S. Zionts. Multiple criteria decision making, multiattribute utility theory: the next ten years. *Management science*, 38(5):645–654, 1992.

[DM01] A. Dumitrescu and J. SB. Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 38–46. Society for Industrial and Applied Mathematics, 2001.

[DMC+14] L. E. Davis, M. J. McNerney, J. Chow, T. Hamilton, S. Harting, and D. Byman. Armed and dangerous? uavs and us security. Technical report, DTIC Document, 2014.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[dVS] da Vinci System. Intuitive surgical. *https://www.intuitivesurgical.com/products/davinci_surgical_system/da-vinci-single-site/*.

[DW09] D. Delling and D. Wagner. Pareto paths with SHARC. In *Experimental Algorithms*, pages 125–136. Springer, 2009.

[ECA04] R. Elvik, P. Christensen, and A. Amundsen. Speed and road accidents. *An evaluation of the Power Model. TØI report*, 740:2004, 2004.

[EGA81] H. El Gindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.

[Ehr00] M. Ehrgott. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, 7(1):5–31, 2000.

[EL11] L. Erickson and S. M. LaValle. An art gallery approach to ensuring that landmarks are distinguishable. In *Proceedings Robotics: Science and Systems*, 2011.

[FBDT99] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *Association for the Advancement of Artificial Intelligence AAAI*, 1999.

[FMW00] S. P. Fekete, J. S.B. Mitchell, and K. Weinbrecht. On the continuous weber and k-median problems. In *Proceedings of the sixteenth annual ACM symposium on Computational geometry*, 2000.

[For92] S. Fortune. Voronoi diagrams and delaunay triangulations. *Computing in Euclidean geometry*, 1:193–233, 1992.

[FP09] Andreas Fabri and Sylvain Pion. Cgal: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 538–539. ACM, 2009.

[Fuj96] K. Fujimura. Path planning with multiple objectives. *Robotics & Automation Magazine, IEEE*, 3(1):33–38, 1996.

[GBL01] H. González-Baños and J.-C. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM, 2001.

[GBLT] S. Gillies, A. Bierbaum, K. Lautaportti, and O. Tonnhofer. Shapely. *URL: http://toblerity.org/shapely*.

[GCB06] A. Ganguli, J. Cortés, and F. Bullo. Maximizing visibility in nonconvex polygons: nonsmooth analysis and gradient algorithm design. *SIAM Journal on Control and Optimization*, 45(5):1657–1679, 2006.

[GFMG04] C. Galindo, Juan-Antonio Fernandez M., and J. Gonzalez. Improving efficiency in mobile robot task planning through world abstraction. *Robotics, IEEE Transactions on*, 20(4):677–690, 2004.

[GGST86] H. N Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[GJ79] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[GLL+97] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. In F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, editors, *WADS '97 Algorithms and Data Structures (Lecture Notes in Computer Science, 1272)*, pages 17–30. Springer-Verlag, Berlin, 1997.

[GNT04] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufman, San Francisco, CA, 2004.

[GV09] P. R. Giordano and M. Vendittelli. Shortest paths to obstacles for a polygonal dubins car. *Robotics, IEEE Transactions on*, 25(5):1184–1191, 2009.

[HP11] S. Har-Peled. *Geometric approximation algorithms (Chapter 17)*, volume 173. American mathematical society Providence, 2011.

[HT73] J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[htt] http://msl.cs.uiuc.edu/msl/. Motion strategy library.

[IK95] C. Icking and R. Klein. Searching for the kernel of a polygon- a competitive strategy. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 258–266. ACM, 1995.

[IKH11] J. T. Isaacs, D. J. Klein, and J. P. Hespanha. Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle. In *American Control Conference (ACC), 2011*, pages 1704–1709. IEEE, 2011.

[int] https://www.youtube.com/watch?v=pfzsabyppy8.

[JDH+06] J. C Juarez, A. Dwivedi, A. Roger Hammons, Steven D Jones, Vijitha Weerackody, and Robert A Nichols. Free-space optical communications for next-generation military networks. *Communications Magazine, IEEE*, 44(11):46–51, 2006.

[JSB+15] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, et al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[KFa] S. Karaman and E. Frazzoli. sampling-based motion planning library for dynamical systems. *https://svn.csail.mit.edu/smp*.

[KFb] S. Karaman and E. Frazzoli. Sampling-based motion planning (smp) library. In *https://svn.csail.mit.edu/smp*. MIT.

[KF10a] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Algorithmic Foundations of Robotics IX*, pages 71–87. Springer, 2010.

[KF10b] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems*, 2010.

[KF11] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[KH04] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[KHB02] I. Kacem, S. Hammadi, and P. Borne. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation*, 60(3):245–276, 2002.

[Kir83] D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[KL90] Nabil A Kartam and Raymond E Levitt. Intelligent planning of construction projects. *Journal of computing in civil engineering*, 4(2):155–176, 1990.

[KL00] J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[KM01] Vineet R Kamat and Julio C Martinez. Visualizing simulated construction operations in 3d. *Journal of Computing in Civil Engineering*, 15(4):329–337, 2001.

[KMadH11] P. Kling and F. Meyer auf der Heide. Convergence of local communication chain strategies via linear transformations: or how to trade locality for speed. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 159–166, 2011.

[KS74] Donald E Knuth and Jayme L Szwarcfiter. A structured program to generate all topological sorting arrangements. *Information Processing Letters*, 2(6):153–157, 1974.

[KSLO96] 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 & Automation*, 12(4):566–580, June 1996.

[KWP+11a] S. Karaman, M. R Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1478–1483. IEEE, 2011.

[KWP+11b] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1478–1483. IEEE, 2011.

[KY14] M. Khan and M. Yuksel. Maintaining a free-space-optical communication link between two autonomous mobiles. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 3154–3159. IEEE, 2014.

[KZ86] Kamal Kant and Steven W Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[LaV06a] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[LaV06b] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[LCH+09] H. Li, N. Chan, T. Huang, H. Guo, W. Lu, and M. Skitmore. Optimizing construction planning schedules by virtual prototyping enabled resource analysis. *Automation in construction*, 18(7):912–918, 2009.

[LD81] D. T. Lee and R. L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10:73–87, 1981.

[LH98a] S. M. LaValle and S. A. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *International Journal of Robotics Research*, 17(1):19–42, January 1998.

[LH98b] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925, 1998.

[LK99] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[LK01] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[LL03] Ming Lu and Heng Li. Resource-activity critical-path method for construction planning. *Journal of construction Engineering and Management*, 129(4):412–420, 2003.

[LOC16] P. Ladosz, H. Oh, and W. Chen. Optimal positioning of communication relay unmanned aerial vehicles in urban environments. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1140–1147, 2016.

[LW08] S. Liu and C. Wang. Resource-constrained construction project scheduling model for profit maximization considering cash flow. *Automation in Construction*, 17(8):966–974, 2008.

[LWW+14] Yuanshan Lin, Di Wu, Xin Wang, Xiukun Wang, and Shunde Gao. Lift path planning for a nonholonomic crawler crane. *Automation in Construction*, 44:12–24, 2014.

[LX05] X. Li and J. Xiao. Robot formation control in leader-follower motion using direct lyapunov method. *International Journal of Intelligent Control and Systems*, 10(3):244–250, 2005.

[Mar96] J. C. Martinez. Stroboscope: State and resource based simulation of construction processes. *Doctoral dissertation*, 1996.

[MAZ+15] A. Monfared, M. Ammar, E. Zegura, D. Doria, and D. Bruno. Computational ferrying: Challenges in deploying a mobile high performance computer. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–6. IEEE, 2015.

[Mer94] R. Merris. Laplacian matrices of graphs: a survey. *Linear algebra and its applications*, 197:143–176, 1994.

[Mit00] J. SB. Mitchell. Geometric shortest paths and network optimization. *Handbook of computational geometry*, 334:633–702, 2000.

[Mit13] J. SB. Mitchell. Approximating watchman routes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 844–855. SIAM, 2013.

[MMCH05] T. Muppirala, R. Murrieta-Cid, and S. Hutchinson. Optimal motion strategies based on critical events to maintain visibility of a moving target. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3837–3842, 2005.

[MPA92] J. S.B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. In *Proceedings of the thirty-third annual ACM symposium on Foundations of Computer Science*, 1992.

[MS84] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984.

[MVSW12] R. J. Meuth, J. L. Vian, E. W. Saad, and D. C. Wunsch. Adaptive multi-vehicle area coverage optimization system and method, September 4 2012. US Patent 8,260,485.

[NS09] J. Nagy and Z. Székely. Police robots and the prüm convention. *Romania*, 2009.

[NX13] Lam K. C. Ning X. Costsafety trade-off in unequal-area construction site layout planning. *Automation in Construction*, 32:96–103, 2013.

[OAJRK14] B. K. Oleiwi, R. Al-Jarrah, H. Roth, and Bahaa I. K. Multi objective optimization of trajectory planning of non-holonomic mobile robot in dynamic environment using enhanced ga by fuzzy motion control and a*. In *Neural Networks and Artificial Intelligence*, pages 34–49. Springer, 2014.

[OC08] K. J. Obermeyer and Contributors. The VisiLibity library. `http://www.VisiLibity.org`, 2008. R-1.

[OOD12] K. J. Obermeyer, P. Oberlin, and S. Darbha. Sampling-based path planning for a visual reconnaissance unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 35(2):619–631, 2012.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.

[O'R04] J. O'Rourke. Visibility. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 643–663. Chapman and Hall/CRC Press, New York, 2004.

[OSH] https://www.osha.gov/SLTC/etools/construction/struckby/mainpage.html.

[OZLL14] J. Ouyang, Y. Zhuang, M. Lin, and J. Liu. Optimization of beamforming and path planning for uav-assisted wireless relay networks. *Chinese Journal of Aeronautics*, 27(2):313–320, 2014.

[PKV10] E. Plaku, L. E Kavraki, and Moshe Y. V. Motion planning with dynamics by a synergistic combination of layers of planning. *Robotics, IEEE Transactions on*, 26(3):469–482, 2010.

[PPBGC17] Ana Puig-Pey, Yolanda Bolea, Antoni Grau, and Josep Casanovas. Public entities driven robotic innovation in urban areas. *Robotics and Autonomous Systems*, 2017.

[PTDM12] Milica P., Z. Tafa, G. Dimi, and V. Milutinovi. A survey of military applications of wireless sensor networks. In *Mediterranean conference on embedded computing (MECO)*, 2012.

[RBD+09] A. Ruckelshausen, P. Biber, M. Dorna, H. Gremmes, R. Klose, A. Linz, F. Rahe, R. Resch, M. Thiel, D. Trautz, et al. Bonirob–an autonomous field robot platform for individual plant phenotyping. *Precision agriculture*, 9(841):1, 2009.

[RBM+16] M. M. Rahman, L. Bobadilla, A. Mostafavi, T. Carmenate, and S. Zanlongo. An automated methodology for worker path generation and safety assessment in construction projects. *IEEE Transactions on Automation Science and Engineering*, 2016.

[RBRa] M. M. Rahman, L. Bobadilla, and B. Rapp. Establishing line-of-sight communication via autonomous relay vehicles. *2016 IEEE Military Communications Conference (MILCOM)*.

[RBRb] M. M. Rahman, L. Bobadilla, and B. Rapp. Sampling-based planning algorithms for multi-objective missions. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE.

[RCB+] M. M. Rahman, T. Carmenate, L. Bobadilla, S. Zanlongo, and A. Mostafavi. A coupled discrete-event and motion planning methodology for automated safety assessment in construction projects. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[RCBM] M M Rahman, T Carmenate, L Bobadilla, and A Mostafavi. Ex-ante assessment of struck-by safety hazards in construction projects: A motion-planning approach. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 277–282.

[RCM04] P. Renaud, E. Cervera, and P. Martiner. Towards a reliable vision-based mobile robot formation control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[Rim17] M. Rimmer. Intellectual property and self-driving cars: Waymo vs uber. 2017.

[RN09] S Russell and P Norvig. The chapter 3 in . *Artificial Intelligence–A Modern Approach, Prentice Hall, Inc*, 2009.

[Rob] Construction Robotics. Sam-100 mason robot. *http://www.nashvillepost.com/business/nashville-post-magazine/article/20834843/build-it-again-sam*.

[RRPS14] D. A. Richie, J. A. Ross, S. J. Park, and D. R. Shires. A monte carlo method for multi-objective correlated geometric optimization. Technical report, DTIC Document, 2014.

[RS08] W. Ren and N. Sorensen. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333, 2008.

[sam] http://www.foxnews.com/tech/2017/03/30/this-construction-robot-can-lay-bricks-6-times-faster-than-can.html.

[SAZ08] S. R. Saunders and A. Aragon-Zavala. *Antennas and propagation for wireless communication systems*. 2008.

[Sim] Simpy: A process-based discrete-event simulation framework.

[SJK08] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1525–1530. IEEE, 2008.

[Tar07a] Z. Tarapata. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2):269–287, 2007.

[Tar07b] Z. Tarapata. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *In-*

ternational Journal of Applied Mathematics and Computer Science, 17(2):269–287, 2007.

[TILT09] O. Tekdas, V. Isler, J.H. Lim, and A. Terzis. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22, 2009.

[Van] Hans Vangheluwe. The discrete event system specification (devs) formalism. Technical report.

[Van01] Hans Vangheluwe. Discrete event modelling and simulation;http://www.cs.mcgill.ca/ hv/classes/ms/discreteevent.pdf. 2001.

[VR81] Yaakov L. Varol and Doron Rotem. An algorithm to generate all topological sorting arrangements. *The Computer Journal*, 24(1):83–84, 1981.

[War87] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[WTK11] N. Watthanawisuth, A. Tuantranont, and T. Kerdcharoen. Design for the next generation of wireless sensor networks in battlefield based on zigbee. In *IEEE Defense Science Research Conference and Expo (DSR)*, pages 1–4, 2011.

[WTM09] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *IEEE Communications Surveys & Tutorials*, 11(2):3–20, 2009.

[WWBB13] D. Watel, M. Weisser, C. Bentz, and D. Barth. Steiner problems with limited number of branching nodes. In *International Colloquium on Structural Information and Communication Complexity*, pages 310–321, 2013.

[YA01] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Elsvier Ad Hoc Networks*, 6(4):329–337, 2001.

[YBZ13] Z. Yu, R. J. Baxley, and G. T. Zhou. Multi-user miso broadcasting for indoor visible light communication. In *Acoustics, Speech and Signal*

Processing (ICASSP), 2013 IEEE International Conference on, pages 4849–4853. IEEE, 2013.

[YGS15] D. Yi, M. A. Goodrich, and K. D. Seppi. Morrf: sampling-based multi-objective motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1733–1739. AAAI Press, 2015.

[YL04] A. Yershova and S. M. LaValle. Deterministic sampling methods for spheres and SO(3). In *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[ZAH10] C Zhang, H AlBahnassi, and A Hammad. Improving construction safety through real-time motion planning of cranes. In *Proceedings of International Conference on Computing in Civil and Building Engineering*, pages 105–115, 2010.

[Zei84] Bernard P Zeigler. *Multifacetted modelling and discrete event simulation*. Academic Press Professional, Inc., 1984.

[Zen13] M. Zenko. *Reforming US drone strike policies*. Number 65. Council on Foreign Relations, 2013.

[ZEP11] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas. Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540, 2011.

[ZHB11] Cheng Zhang, Amin Hammad, and Jamal Bentahar. Multi-agent-based approach for real-time collision avoidance and path re-planning on construction sites. In *28th Int. Symp. on Autom. & Robotics in Const., Seoul*, 2011.

[ZL07] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.

[ZLLZ08] J. Zhao, G. Liu, Y. Liu, and Y. Zhu. Research on the application of a marsupial robot for coal mine rescue. *Intelligent Robotics and Applications*, pages 1127–1136, 2008.

VITA

MD MAHBUBUR RAHMAN

	Born, Bangladesh
	B.S., Computer Science and Engineering BUET Dhaka, Bangladesh
2015	M.S., Computer Science Florida International University Miami, Florida
05/2015–07/2015	Robotic Engineer Intern TE Connectivity Harrisburg, Pennsylvania
08/2012–04/2017	Graduate Assistant Florida International University Miami, Florida
05/2017–07/2017	Dissertation Year Fellow Florida International University Miami, Florida
02/2017	Patent No. US9576359 B2 United States Patent and Trademark Office Published on February 21, 2017.

PUBLICATIONS AND PRESENTATIONS

[[1]] An Automated Methodology for Worker Path Generation and Safety Assessment in Construction Projects. M M Rahman, L Bobadilla, A Mostafavi, T Carmenate, S Zanlongo. IEEE Transaction on Automation Science (T-ASE), 2016.

[[2]] Establishing Line-of-Sight Communication Via Autonomous Relay Vehicles. M M Rahman, L Bobadilla, B Rapp. 2016 IEEE Military Communication Conference (MILCOM), Baltimore, MD.

[[3]] Sampling-Based Planning Algorithms for Multi-Objective Missions. M M Rahman, L Bobadilla, B Rapp. 2016 IEEE Conference on Automation Science and

Engineering, Fort Worth, TX.

[[4]] A Coupled Discrete-Event and Motion Planning Methodology for Automated Safety Assessment in Construction. M M Rahman, T Carmenate, L Bobadilla, A Mostafavi, S Zanlongo. 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA.

[[5]] Ex-Ante Assessment of Struck-by Safety Hazards in Construction Projects: A Motion Planning Approach. M M Rahman, T Carmenate, L Bobadilla, A Mostafavi. 2014 IEEE International Conference on Automation Science and Engineering, Taipei, Taiwan.

[[6]] Modeling and analyzing occupant behaviors in building energy analysis using an information space approach. T Carmenate, M M Rahman, D Liante, L Bobadilla, A Mostafavi. 2015 IEEE Conference on Automation Science, Sweden.

[[7]] Multi-Robot Planning for Non-Overlapping Operator Attention Allocation. S Zanlongo, M M Rahman, F Abodo and L Bobadilla. IEEE Robotic Computing, Taiwan, 2017.

[[8]] Context Based Algorithmic Framework for Identifying and Classifying Embedded Images of Follicle Units. M M Rahman, S S Iyengar, W Zeng, B Nusbaum, P Rose. 2014 SPIE Medical Imaging, San Diego, CA.

[[9]] Hybrid Motion Planning/Discrete-Event Simulation Approach Construction Safety Planning (2 pages). M M Rahman, T Carmenate, L Bobadilla, A Mostafavi. 5th Workshop on Formal Methods for Robotics and Automation in 2014 RSS, Berkeley, CA.