Multi Robot Exploration and Map Building with Assured Collision Avoidance

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

by

Sarat Chandra Nagavarapu
(Roll No. 10423004)

Under the guidance of
Prof. Leena Vachhani
and
Prof. Arpita Sinha



IDP in Systems and Control Engineering
INDIAN INSTITUTE OF TECHNOLOGY–BOMBAY
2016

To

My Parents

Gouri Maha Lakshmi

Venkata Krishna Rao

&

My Brother

Siva Rama Krishna

Thesis Approval

The thesis entitled

Multi Robot Exploration and Map Building with Assured Collision Avoidance

by

Sarat Chandra Nagavarapu
(Roll No. 10423004)

is approved for the degree of

Doctor of Philosophy

_____	_____
Examiner	Examiner
_____	_____
Guide	Coguide

_____
Chairman

Date: _____

Place: _____

DECLARATION

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Date :

Name : Sarat Chandra Nagavarapu

Roll No : 10423004

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY, INDIA

CERTIFICATE OF COURSE WORK

This is to certify that **Sarat Chandra Nagavarapu** (Roll No. 10423004) was admitted to the candidacy of Ph.D. degree December 2010, after successfully completing all the courses required for the Ph.D. programme. The details of the course work done are given below.

S.No	Course Code	Course Name	Credits
1	SC 601	Modeling and Identification of Dynamical Systems	6
2	SC 602	Control of Nonlinear Dynamical Systems	6
3	SC 625	System Theory	6
4	SC 628	Guidance Strategies for Autonomous Vehicles	6
5	SC 700	Embedded Control System	6
6	MA 5109	Graph Theory	6
7	SCS 801	Seminar	4
8	HS 699	Communication and Presentation Skills	4
		Total Credits	**44**

IIT Bombay

Date: Dy. Registrar (Academic)

© Copyright by Sarat Chandra Nagavarapu 2016

All Rights Reserved

Abstract

Autonomous exploration and map building of an unknown environment has many applications in search and rescue, surveillance, navigation, etc. This work investigates multi robot map building and inter robot collision avoidance techniques while exploring an unknown environment. Map of the environment is represented as a graph and the task is to explore each and every vertex and edge of the unknown graph. Redundancy minimization, condition for declaring completion, finite time completion and collision avoidance while exploring the graph using multiple robots are the major challenges addressed in this work.

An efficient multi robot graph exploration strategy is presented in which each robot takes independent decision for exploring unvisited portions of the unknown graph. Each robot builds its own map while exploring the environment. An efficient data structure representing the partial map information allows exchange of information between the robots. This work presents a variation of incidence matrix as efficient data structure to minimize the redundancy in exploration. It also provides an easy condition for checking the completion and finite time exploration, which are supported by linear algebra concepts. The proposed technique requires either lesser or equal number of edge traversals compared to the existing strategies for a tree exploration.

A generic framework for multi robot exploration and mapping which is independent of communication method used between robots, initial placement of robots and search strategy is also proposed. The work presents an iterative approach to determine the optimal fleet size to complete the exploration in minimum number of edge traversals using the proposed generalized framework. In multi robot systems, avoiding inter robot collisions is crucial for efficient task completion and increased robustness. A predefined constant speed change approach is proposed to ensure collision avoidance using minimal sensing.

Simulation results verify the performance of the algorithm on various trees and graphs. Experiments with multiple robots show multi robot exploration avoiding inter-robot collision with robustness to robot failures.

Contents

Abstract **i**

List of Tables **v**

List of Figures **vi**

Nomenclature **viii**

1 Introduction **1**

1.1 Motivation 3

1.2 Contributions 5

1.3 Thesis Organization 7

2 Literature Survey **9**

2.1 Exploration and Mapping 9

2.2 Collision Avoidance 12

2.3 Optimal Number of Robots 13

2.4 Remarks 14

3 Preliminaries on Graph Based Exploration and Mapping **15**

3.1 Terminology 15

3.2 Vector Space associated with a Graph 20

3.3 Conclusion 21

4 Improved Multi Robot Depth First Search (IMRDFS) Strategy **22**

4.1 Improved Multi Robot Depth First Search (IMRDFS) Strategy 23

4.2 Analytical Results 27

4.3 Algorithm using Modified Incidence Matrix 34
4.4 Simulation Results 44
4.5 Comparison with Existing Methods 47
4.6 Conclusion 51

5 A Generic Framework for Multi Robot Exploration and Mapping 52
5.1 Proposed Framework for Multi Robot Graph Exploration 53
5.2 Analytical Results 60
5.3 Algorithm for the Generic Framework 62
5.4 Number of Robots 67
5.5 Simulation Results 68
5.6 Comparison with Existing Methods 70
5.7 Conclusion 73

6 Multi Robot Collision Avoidance while Mapping 74
6.1 Collision Detection 75
6.2 Collision Avoidance Strategy 77
6.3 Analytical Results 78
6.4 Algorithm for Collision Avoidance 83
6.5 Conclusion 84

7 Hardware Implementation 85
7.1 Hardware Setup 85
7.2 Experimental Arena 87
7.3 Experimental Results 88
7.4 Robustness to Robot Failures 91
7.5 Conclusion 92

8 Conclusion and Future Work 93
8.1 Future Work 94

References 95

Publications 101

Acknowledgments **102**

List of Tables

5.1 Merge operation 56
5.2 Comparison with analytical method 72
5.3 Reasoning based on K_{sat} and N_{sat} obtained from the proposed method and results from the fast collaborative exploration approach 72

List of Figures

1.1 Multi robot exploration and mapping 2

3.1 Illustration of terminology and proposed strategy 16

4.1 Partially completed graph 28
4.2 Robot edge traversals 30
4.3 Illustration of edge selection and approaching the connected vertex 31
4.4 Robot cyclic traversal 32
4.5 Repetitive edge traversal 33
4.6 Partially explored graph 34
4.7 Flowchart of IMRDFS Strategy 38
4.8 Shortest path to a vertex in partial graph 40
4.9 Example graph traversal with two robots 41
4.10 Tree topologies with 12 edges 45
4.11 Number of edge traversals required to complete exploration Vs. number of robots 46
4.12 Simulated result of a graph exploration using two robots 46
4.13 Illustration of partitioning of a tree in two subtrees G_1 and G_2 47
4.14 Exploration using proposed technique on an example graph 49
4.15 Exploration using proposed method with 5 robots 50
4.16 Exploration using the MRDFS [1] with 5 robots 50
4.17 Number of edge traversals required to complete exploration Vs. number of robots for a square lattice with 100 vertices and 180 edges 51

5.1 Edge and vertex sets of a partially explored graph 54
5.2 Edge set update 57
5.3 Component wise merge operation 58

5.4 Components of an example edge set $\mathcal{E}_A$ and their association with the edge set $\mathcal{E}$ 59
5.5 Arbitrary tree structure with 39 edges 69
5.6 $\%\eta$ computed for MRDFS strategy on the representative graph topologies; iterative values are marked using various markers 70
5.7 $\%\eta$ computed for IMRDFS strategy on the representative graph topologies; iterative values are marked using various markers 70
5.8 $\%\eta$ computed for MRLRV strategy on the representative graph topologies; iterative values are marked using various markers 71
5.9 $\%\eta$ computed for MRBFS strategy on the representative graph topologies; iterative values are marked using various markers 71

6.1 Robot collision detection 76
6.2 Edges with lanes 78
6.3 Collision avoidance using independent decisions 79
6.4 All possibilities of encountering between two robots 81
6.5 Simultaneous collision detections with multiple robots 83

7.1 Spark V Robot 86
7.2 Experiment on Arena 1 using two robots 87
7.3 Experiment on Arena 2 using four robots 87
7.4 Robot path traversals for the experiment on Arena 1 using two robots 88
7.5 Path traversals for Robot 1 on Arena 2 89
7.6 Path traversals for Robot 2 on Arena 2 90
7.7 Path traversals for Robot 3 on Arena 2 90
7.8 Path traversals for Robot 4 on Arena 2 90
7.9 Robot failure while exploring the environment 91

Nomenclature

v_i	A vertex.
v_r	Root vertex.
e_{ij}	An edge between two vertices v_i and v_j.
R_k	k^{th} Robot.
$\mathcal{V}$	Set of vertices.
$\mathcal{E}$	Set of edges.
$G(\mathcal{V}, \mathcal{E})$	Graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$.
E	Number of edges in the graph G.
V	Number of vertices in the graph G.
K	Number of robots exploring the graph G.
θ_{ji}	Incidence angle subtended by an edge e_{ji} at the vertex v_j.
$\mathcal{E}_c^n(R_k)$	Set of completed edges available with the robot R_k after n^{th} update.
$\mathcal{E}_c^n(v_j)$	Set of completed edges stored at the vertex v_j after n^{th} update.
$\mathcal{E}_o^n(R_k)$	Set of out edges available with the robot R_k after n^{th} update.
$\mathcal{E}_o^n(v_j)$	Set of out edges stored at the vertex v_j after n^{th} update.
$\mathcal{E}_u^n(R_k)$	Set of unexplored edges available with the robot R_k after n^{th} update.
$\mathcal{E}_u^n(v_j)$	Set of unexplored edges stored at the vertex v_j after n^{th} update.
$\mathcal{E}_m^n(R_k)$	Set of my-unexplored edges available with the robot R_k after n^{th} update.
$\mathcal{E}_m^n(v_j)$	Set of my-unexplored edges stored at the vertex v_j after n^{th} update.
$G_c^n(R_k)$	Partially completed graph available with the robot R_k after n^{th} update.
$G_c^n(v_j)$	Partially completed graph stored at the vertex v_j after n^{th} update.

$\mathcal{V}_c^n(R_k)$	Set of visited vertices available with the robot R_k after n^{th} update.
$\mathcal{V}_c^n(v_j)$	Set of visited vertices stored at the vertex v_j after n^{th} update.
I	Modified incidence matrix.
$I^n(R_k)$	Incidence matrix available with the robot R_k after n^{th} update.
$I^n(v_j)$	Incidence matrix stored at the vertex v_j after n^{th} update.
$\mathcal{G}_A$	Partially explored graph available with a source A.
$\mathcal{V}_A$	Vertex set of a partially explored graph available with a source A.
$\mathcal{E}_A$	Edge set of a subgraph $\mathcal{G}_A$.
$\mathcal{E}_A^{new}$	Updated edge set $\mathcal{E}_A$.
$\mathcal{E}_{R_k}$	Edge set availabe with a robot R_k.
$\mathcal{E}_{B_i}$	Edge set available with a beacon B_i.
$\mathcal{E}_M$	Merged vector of two vectors.
$\mathcal{E}_{v_j}$	Edge set associated with a vertex v_j.
$\mathcal{E}_{AB}$	Intersection of two edge-sets $\mathcal{E}_A$ and $\mathcal{E}_B$.
e_{ix}	Half edge identified at a vertex v_i.
$\mathcal{W}_\mathcal{E}$	Edge space (set of all the subsets of $\mathcal{E}$).
I_A	Incidence matrix associated with a source A.
I_M	Incidence matrix corresponding to the vector M.
p	Weight of an edge.
η	Performance measure.
N	Number of edge traversals to complete the exploration.
N_{sat}	Least number of edge traversals to complete the exploration.
K_{sat}	Number of robots to achieve N_{sat}.
K_{opt}	Optimal number of robots.
$\mathcal{H}_A(n)$	Number of unexplored half edges of a source A at a particular edge traversal n.

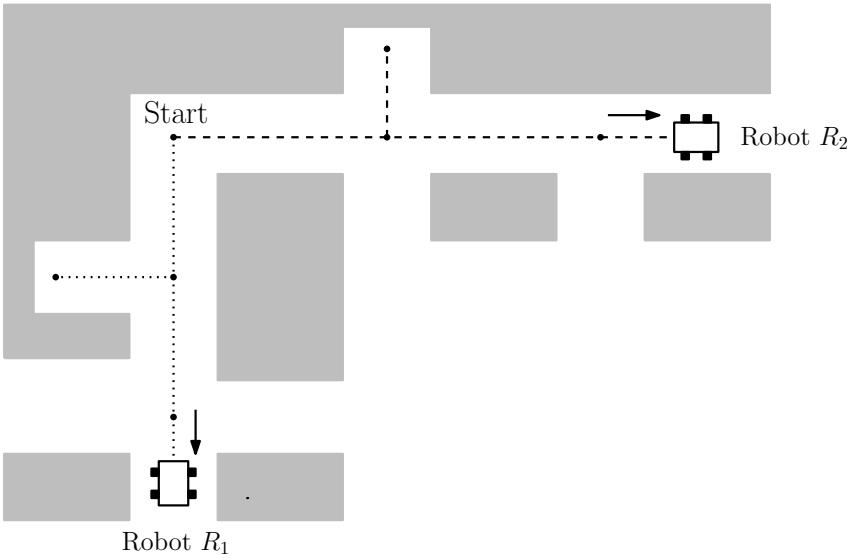
Chapter 1

Introduction

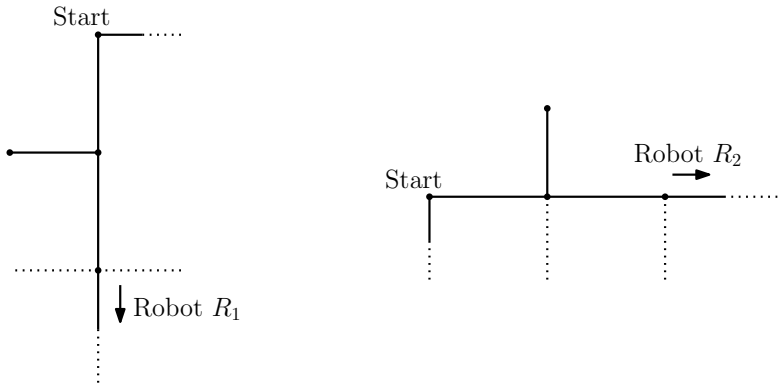
Mobile robots are being increasingly used to perform tasks such as surveillance, search and rescue in disaster areas, wildfire monitoring, demining, coverage, planetary exploration, environmental mapping, searching for harmful chemical substances, etc. These tasks can be performed either by using a single robot or a group of mobile robots working in coordination and require the robot(s) to completely explore the environment. Executing these missions using multiple robots achieves additional benefits of faster task completion, robustness and improved efficiency. The *multi robot exploration and mapping* addresses the systematic and coordinated exploration of an unknown environment by a group of mobile robots.

Robots equipped with sensors gather information about the environment and integrate it into a given representation. The term mapping in robotics is often described by the question *What does the world look like?* Each robot in the group incrementally builds its own partial map of the environment, termed as *local map* and all these local maps acquired by the robots are merged to obtain the *global map* of the environment. Thus, the merging maps procedure integrates the local map information to obtain a global map. The generated global map of the environment can be reused for robot navigation through effective utilization of the resources. Maps are mainly classified into two types [2]: metric and topological. Metric maps are built around the geometry of the area being mapped (e.g. occupancy grid), whereas topological maps are concerned with the connections between key regions in the environment (e.g. graph). This work aims to develop graph based map of an unknown environment.

An example of a multi robot exploration and mapping framework using two robots in an unknown environment is illustrated using Fig. 1.1. In Fig. 1.1(a), the obstacles are shown using gray colored boxes and the traversals of the robots R_1 and R_2 are depicted using the dotted and



(a) Multi robot exploration



(b) Map generated by the robot R_1 (c) Map generated by the robot R_2

Figure 1.1: Multi robot exploration and mapping

dashed lines respectively. The map of the explored environment is represented using a graph. The junction points and the free spaces in the environment are illustrated as the vertices and the edges of the generated graph respectively. No direct communication between the robots is considered for information/map exchange. We assume that the environment is bounded and unknown apriori. Robots R_1 and R_2 begin the exploration from the *Start* point and move into two different portions of the environment. As the exploration progresses, many areas in the environment become visible to the robots and the corresponding maps are generated. The local maps generated by the robots R_1 and R_2 at the instance shown in Fig. 1.1(a) are described in Figs. 1.1(b) and 1.1(c) respectively. For a map generated by a robot, the explored edges of the environment, edges that are chosen by some other robot, the unexplored edges originating from a vertex in the explored region are represented by solid lines, solid-dotted lines, dotted lines

respectively.

Multiple robots maneuvering in an environment must avoid collision with each other in order to guarantee an efficient task completion. The plan of action that the robots seek to evade the incoming collision is known as *collision avoidance*. Collision avoidance has been a challenging problem associated with the coordinated multi robot exploration. Collision between robots while exploring a graph is possible, when the robots follow a common edge or reach to a common vertex. In order to avoid a collision, the robot must have the capability to detect a collision. *Collision detection* is the act of scanning the robots' vicinity to detect the presence (or absence) of a possible collision. The probability of collisions increase with the increase in the number of robots. It is important to address inter robot collision avoidance to ensure efficient completion.

The multi robot exploration task is declared to be complete, only when each and every vertex and edge of the graph is explored. The performance of any autonomous exploration can be assessed using the *completion time*. The time required to complete an exploration task depends primarily on the number of robots deployed to perform exploration. Determining the optimal number of agents (fleet/team size) for exploring a given graph is important, because it contributes to the reduction of completion time and cost/energy consumption of the multi robot system.

1.1 Motivation

Multiple robots working in coordination for an autonomous application have several advantages over single robot executing the same task. The number of steps taken by each robot to complete the task reduces when multiple robots follow a strategy to reduce the redundancy. A multi robot system is more robust when compared to a single robot system as the task can still be executed even if any robot fails. Furthermore, the reliability and efficiency are higher in the case of a multi robot system. However, exploration using multiple robots need to address following major challenges:

- **Declaration of completion:** The exploration task is complete when no portion of the environment is left unexplored. A major challenge in autonomous exploration is to declare completion, which signifies that the entire environment is explored, particularly when the environment is explored using decentralized control of multiple robots.

- **Redundant exploration:** When multiple robots explore an environment, the same region in the environment may be covered by several robots, which results in redundancy. Thus, minimizing the redundancy is important to achieve faster completion.

- **Coordinating the robots:** In order to minimize redundant exploration and reduce the exploration time, it is necessary to establish coordination among the robots. Coordination can be achieved by incorporating an efficient edge selection scheme by sharing the updated partial maps between the robots.

- **Communication scheme:** The mode of communication between robots is also a key factor in determining the effectiveness of the exploration. Therefore, designing an efficient communication framework (for example, centralized, decentralized, broadcasting, etc.) is crucial to establish an organized exchange of partial map information between the robots.

- **Inter robot collision avoidance:** It is challenging to design a collision avoidance technique that works efficiently for a multi robot system with decentralized control. Collision detection and avoidance becomes more difficult with the additional constraint of minimal sensing.

- **Map merging:** The decisions pertaining to edge selection during multi robot exploration depend on the updated map information. So, merging the partial maps from multiple robots is essential to achieve efficient exploration.

An exploration framework is based on many aspects such as search strategy, inter robot communication scheme, initial placement of the robots, edge selection mechanism for exploration of new areas, etc. When an unknown environment is explored using multiple agents, the movements of all the agents must be coordinated using an efficient information exchange method. Therefore, we aimed to answer following questions: When to declare completion? What is the guarantee that the movements of multiple agents are leading to completion in finite time? Can we answer to these queries irrespective of information exchange method, any initial placement of agents and strategy for movement of individual agent? Is there a chance of faster exploration by increasing the number of agents? Therefore, a generic framework for multi robot exploration and mapping is needed to develop any exploration strategy independent of these aspects.

Optimal team size is an evident query in multi robot exploration. With the increase in the number of robots exploring the environment, the completion time decreases exponentially. The completion time of the exploration determines the cost of exploration and energy consumption by the robots. Even with a team of infinite number of robots, the exploration time can not be reduced indefinitely beyond a limit. Therefore, determining the least upper bound for team size to achieve minimum completion time is of interest for minimizing the exploration cost.

1.2 Contributions

This work investigates *decentralized exploration and mapping strategy aiming for efficiency and faster graph exploration by minimizing redundancy and avoiding inter-robot collisions. Objectives also include development of a generalized framework, finding optimal team size and minimum completion time for multi robot exploration without knowing the details of initial placement of each agent, search strategy and information exchange method.* Our contributions in this work are summarized as follows:

- **Decentralized exploration using improved multi robot depth first search (IMRDFS) strategy:** The work proposes an IMRDFS strategy using an efficient data structure. The incidence matrix as data structure is modified to accommodate a systematic and faster multi robot exploration. The modified incidence matrix records the partially explored graph information and represents map of the environment when exploration is complete. Redundancy in edge traversals is minimized using the latest map information that is available with the robots at each update instance. A new condition for declaring completion is obtained. The proposed technique promises completion of exploration in finite number of edge traversals. The total edge traversals required to complete a tree exploration using proposed exploration strategy with multiple robots is at most equal to the number of edges in the tree. The edge traversals required to complete the exploration is always equal or less than that required using MRDFS method [1]. The improvement over MRDFS [1] is guaranteed when difference between radius and half the number of edges of the tree is at least one. The minimum completion time achieved by the proposed IMRDFS strategy has also been verified by comparing it with several other search strategies such as MRDFS [1], BFS [3, 4], LRV [5], etc. Various experimental and simulation results validate the proposed technique.

- **Unified framework for multi robot graph exploration:** A unified framework using the modified incidence matrix data structure is proposed for any initial placement of the agents, search strategy and information exchange method. The key challenge is to coordinate the agents for exploiting the maximum benefit of multiple agents by effective use of the partial map information. A generic approach is developed to obtain the conditions for completion and systematic strategy leading to finite time completion. An iterative approach to find an optimal number of agents and minimum exploration time is also developed and compared with the existing techniques [6], [7]. The optimal fleet size obtained using the proposed iterative approach guarantees a minimum step completion on any graph topology.

- **Multi robot collision avoidance:** Obtaining a collision condition in the absence of inter robot communication using minimal sensing is a major challenge in collision avoidance. It is shown that minimal sensing is sufficient to detect the possibility of collision in a decentralized system. A necessary and sufficient condition for collision is found. When a possible collision is detected, the proposed strategy of changing speeds of the robots to predefined values is proved to avoid collision. Knowledge of possible collision before changing the speed is the only sensing requirement for proposed collision avoidance scheme. Unlike existing work on multi robot collision avoidance [8–11], the proposed strategy suggests speed change for the robots to avoid collisions without knowing the position or velocities of other robots and without communicating the decision of speed change.

- **Experimental validation:** The experimental validation of the proposed strategy is carried out using a team of Spark V robots exploring the environment. The Xbee transceivers are interfaced to each robot to establish the communication framework among the robots and the vertex beacons. The tree and graph explorations have been validated using two different experiments, one with a tree structure using four robots and the other with a graph structure using two robots. Collision avoidance has also been incorporated into the experimentation by changing the speeds of the robots whenever needed. The robustness of the proposed strategy against the failure of the robots is also validated experimentally using a team of spark V robots exploring an environment with tree structure.

1.3 Thesis Organization

Rest of the thesis is organized as follows:

Next chapter presents an overview of the relevant research in multi robot exploration and mapping, collision avoidance and optimal fleet size. This chapter also details the existing techniques in contrast to the proposed strategy to perform the assigned multi robot task.

Chapter 3 introduces the preliminaries on graph based multi robot exploration and mapping. The elementary concepts of algebraic graph theory and vector space associated with graphs and terminology used in this work are also presented.

In Chapter 4, the proposed improved multi robot depth first search (IMRDFS) strategy is presented with a new condition to declare completion. A modified incidence matrix data structure is proposed to store and update the partial map information. It offers the advantages of systematic and coordinated exploration, reduced redundancy and improved robustness. The finite time completion of the exploration strategy is proved using theoretical analysis. The comparative analysis of the proposed strategy with that of the existing methods is also described. Simulation results verify the performance of the proposed strategy on various tree and graph topologies with varying team size.

Chapter 5 illustrates a generic framework for multi robot exploration and mapping. This generic framework for multi robot exploration and mapping is designed for any given tree/graph topology, initial position of the agents, communication framework and search strategy. This work also presents an iterative method to determine the optimal fleet size for exploring any given graph. A performance measure is proposed, which demonstrates the improvement in multi robot exploration over two robot exploration. The results are analyzed for several search strategies on various tree and graph topologies using simulations.

Chapter 6 describes the proposed collision avoidance scheme using minimal sensing. A condition for collision detection and a speed reduction based approach for collision avoidance are proposed. Collision avoidance is achieved without knowing the position or velocities of other robots and without communicating the decision of speed change. Therefore, the proposed collision avoidance technique with minimal sensing has the advantage of reduced sensor data processing and minimal communication and computation overhead on the robots.

The hardware implementation of the proposed IMRDFS strategy using a team of mobile robots is discussed in Chapter 7. A team of differential drive robots are used on two different

arenas with tree and graph structures and the finite time completion of the proposed strategy is validated. The hardware implementation of the speed reduction based collision avoidance technique and the robustness validation of the proposed IMRDFS strategy are also reported.

Conclusions related to the success of this work along with some recommendations for future directions are presented in Chapter 8.

Chapter 2

Literature Survey

Multi robot exploration in robotics is an interesting field of research which deals with a systematic and coordinated coverage of an unknown environment by a group of mobile robots. If a team of mobile robots have to accomplish the task of exploring an environment, they have to coordinate among themselves by establishing either a centralized or a decentralized communication protocol. The information exchange or map updates determine the effectiveness of the exploration. Multi robot exploration has been advantageous over the single robot exploration in minimizing the exploration time and the average distance traveled by each robot to complete the exploration. The existing literature is discussed in following way: various mapping and graph exploration techniques are discussed in next section. The autonomous multi robot application needs to deal with an additional task of inter robot collision avoidance. Therefore, a survey of existing work on inter robot collision avoidance is presented in section 2.2. This work also investigates the problem of finding an optimal team size to achieve faster completion avoiding redundancy. Section 2.3 discusses the existing results on team size requirement for efficient graph exploration.

2.1 Exploration and Mapping

Communication method between robots is a major factor in designing the multi robot exploration technique. The information is exchanged in existing decentralized approaches for multi robot exploration in several ways. Robots drop book keeping devices (beacons) at important locations and share the partial map information stored at these beacons or through local communication at a vertex [1, 12, 13]. In some cases, robots request a special call for meeting to

share/update the maps [14, 15]. Each robot broadcasts the partial map information in limited range [16] and map information is updated.

A rendezvous based multi robot exploration strategy for undirected graphs has been proposed in [14], where multiple robots start the exploration from a root node and map the environment. Rendezvous is requested after a specific time and the maps are merged and updated. The same strategy with rank based location selection scheme for rendezvous has been proposed in [15]. [17] gives a decentralized cooperative exploration strategy using sensor based random graphs (SRG) in which a local coordination strategy decides the task allocation and collision avoidance. [18] discusses a distributed graph coloring technique for map building where the nodes traversed by a robot are filled with a unique color. For multi robot graph exploration, this coloring scheme avoids the redundancy in exploration. Space decomposition based exploration strategies are well known in the exploration and mapping literature. The environment to be explored is decomposed into cells, polygons, etc. and the exploration strategy direct the robots to achieve the collective exploration.

A systematic strategy is required to complete the exploration. Single and multi robot exploration strategies available in literature are discussed next. Robots in a depth first search (DFS) [1] traverses a tree or graph by starting at the root and explores as far as possible along each branch before backtracking. Breadth first search (BFS) [3, 4] on the other hand, instruct robots to explore the neighbor nodes at the highest levels first, before moving to the next level neighbors. In leat recently visited (LRV) [5] approach, robot deploys sensor nodes and maintains a sensor network, while covering and exploring the environment. At each node, the edge with minimum weight is chosen for next traversal. The MRDFS [1] and BFS [3, 4] strategies have been used for exploring graphs using multiple robots. Several parallel algorithms for graph exploration such as parallel DFS [19] and parallel BFS [20] have been proposed to reduce the exploration time of multi robot exploration strategies. A new approach to implement parallel BFS [20] on multi core CPUs through effective utilization of memory bandwidth has been presented. For graph search in case of massive data sets, another hybrid method for parallel BFS [20] using GPUs has also been proposed. The work on parallel DFS [19] investigates the implementation methodologies of DFS by means of shared memory models, parallel comparison trees, fixed connection networks, etc. to reduce the algorithm runtime. Randomized parallel algorithms for undirected graphs using DFS have also been detailed.

A master-slave based centralized multi robot depth first search exploration strategy has

been presented in [21] in which the search space is decomposed into occupancy grids and the master robot dictates the moves to the slave robots. The maps build by the slave robots are incrementally updated and merged to obtain a global map of the environment. A distributed multi robot exploration strategy for limited communication scenarios has been discussed in [22]. The strategy make the robots to stick together within the communication region and completes the free space exploration. A distributed exploration strategy [23] has been proposed for a group of robots with unknown initial positions. Each robot in the group builds its own map and the map information (position and orientation) is exchanged with other robots and the mapping errors are corrected.

A cell decomposition based exploration technique with square sized cells has been suggested [24] for ant type robots such that each robot in the team builds a spanning tree of the explored cells. These individual spanning trees are merged to reduce repetitive exploration. Static sensors [25] are used as markers to represent the explored portions of the environment. The Back-Tracking Deployment (BTD) algorithm has been an extension to the snake Like Deployment (SLD) algorithm where the marker color represent the unexplored grid cells and the robots back track to the nearest unexplored cell by back tracking the explored cell sequence. [26] presents a detailed mathematical analysis on the complexity of single robot exploration in an unknown environment populated with polygonal obstacles.

An online multi robot exploration strategy called as Brick and Mortar [27] algorithm has been proposed. The environment is decomposed into square grid cells and the exploration decision is made based on the adjacent cell occupancy. The cells are incrementally marked as visited and the exploration continues. A variant of the ant algorithm for multi robot exploration for unknown environments [28] has been developed to explore the cells and the exploration information exchange is made through the cells by writing the exploration data to them.

A depth first search based exploration strategy for a group of robots under limited communication constraints has been proposed in [29]. The robots move together as a pack and explore the map preserving the communication link. The map information is continuously broadcasted and a global map is obtained. A frontier cell based decentralized exploration strategy has been proposed in [30], where the robots exchange the map information at each new frontier cell that they meet. Another variant of frontier cell based multi robot exploration strategy [31] has been presented for robots with limited communication capabilities. A team of robots maintain a continuous communication link and explore the environment by expanding the frontier cells.

Whenever the robots meet, the explored map information is exchanged.

To address the multi robot exploration problem for larger environments with uncertain boundaries, a Voronoi diagram based partitioning technique has been proposed in [32]. The environment is decomposed into Voronoi cells and the robots are dictated to move towards the unknown cells of the Voronoi. Deployment of sensor nodes along the vertices of a Voronoi map [33] has been an efficient way of constructing an information network. Each robot drops a sensor node at a new explored vertex of the constructed Voronoi map. The maps at each sensor node are merged to obtain a complete information network to track and the environment from intruders. Decision theoretic approach [34] has been used to propose distributed multi robot exploration. Each robot in the team individually maps the environment and the exploration frontiers are determined for each of the robots separately and the map merging minimizes repetitive exploration.

2.2 Collision Avoidance

Multi robot exploration problem also has to address collision with other robots. While exploring the graph-like environment using multiple robots, there is a possibility of robots colliding with each other. Multiple robots maneuvering in an environment must avoid collision with each other in order to guarantee an efficient task completion. Especially with the increase in the size of the robot team, the probability of inter robot collisions become imminent. Agent velocity control is one of the key factors to achieve collision avoidance. A dynamic window [8] of velocities around the agent's current velocity has been computed for the given sampling time to avoid collisions. Associating the robots with aggressive levels [35] has been studied as a solution to the multi robot collision avoidance problem. Velocity obstacle cone [36] corresponding to the neighboring agents yield the collision free velocity for every agent in the team.

Collision in multi robot application has been avoided by modifying speed of the robot [8], updating steering of the robot [9, 10, 37, 38] or deciding slaves' action based on that of master [11, 39], when possibility of a collision with other robot(s) is detected. Associating the robots with aggressive levels [35] has been studied as a solution to the multi robot collision avoidance problem. Velocity obstacle cone [36] corresponding to the neighboring agents yield the collision free velocity for every agent in the team. Given the initial state of the agents, the set of collision inputs are determined and an approximate supervisor [40] for collision avoidance

is computed. Maintaining a desired minimum distance of separation among the agents [41] with agent position broadcasting solves the issue of inter robot collisions. Collision has also been avoided by developing Speed-Time Graph (STG) [42]. Exploration and map building using multiple robots also need to deal with issues such as redundant exploration, collision avoidance, coordination and information exchange between the robots [43].

2.3 Optimal Number of Robots

The selection of number of agents is a major factor in an efficient exploration as it contributes to the reduction of completion time and increment in total energy of the multi agent system. A probabilistic approach [44] has been developed for finding the fleet size considering various factors affecting the exploration. The negative exponential relationship exists between completion time and number of robots [7]. An analytical method [7] and polynomial approximation to the relationship [45] have been developed to obtain the maximum number of robots providing maximum reduction on completion time for specific topologies. The asymptotic relationship between number of robots and exploration time [6] has been obtained under global and local communication framework. A competitive ratio [46] has been proposed to validate the performance of an online centralized exploration algorithm with respect to an optimal search algorithm with a team of searchers exploring an unknown graph. However, this work addresses development of a generic and simple non-iterative simulation technique to find the optimal number of agents for exploration.

In [47], optimization of fleet size for frontier based exploration using simulations has been presented. The proposed strategy has been simulated by varying the parameters such as fleet size, robot initial positions, obstacle density, environments etc. and the optimal robot count for a known arena has been determined. An approach towards energy efficient fleet size determination has been proposed in [48]. This strategy considers mobile sensors covering an area populated with obstacles of un-known locations and known density. The speed of the robots has been varied according to the time and energy constraints to complete the exploration. Simulations have been carried out for varying fleet size and a reduced mobile sensor count has been determined. To the best of our knowledge, a generic approach for multi robot exploration is not available in literature.

2.4 Remarks

This chapter presents the relevant literature in the area of multi robot exploration and map building. Multiple robots coordinate with each other by exchanging information using various methods. The method of exchanging information between robots is a major factor in designing an exploration technique. The existing approaches for multi robot exploration use following ways for information exchange: when the robots meet, a special call for meeting, book keeping devices (beacons) at important locations sharing partial map information [13] or location information [1, 12] and broadcast by each robot in limited range [16]. In all these methods, the inter robot communication is established using book keeping devices/sensor nodes/active landmarks (beacons) at the vertices of the graph. The flooding algorithm [13] aims to minimize the overall distance traveled by the robots and allows only one robot to occupy any edge/node at a given instance. The MRDFS [1] and LRV [5] strategies use only the vertex local information to select the next edge. The information required to store at a vertex is minimal, however, it may result in redundant exploration. Therefore, an effective data structure for storing the partial map at the explored vertices avoids redundancy in exploration.

The fleet size of a multi-robot system is a key parameter that determines the cost and the time of execution of any given task. Therefore, finding the optimal number of robots is a key to reduce the completion time and overall system cost. Past work in determining optimal fleet size [6, 7, 44, 45] deals with specific cases of graph exploration. To the best of our knowledge, a unified framework that supports finding the optimal fleet size for any given topology, initial position of robots, communication framework and search strategy using the proposed unified framework is not addressed in literature.

In many robotic applications, collision avoidance is achieved by modifying speed [8, 36] and steering angle [9,10,37,38] of the robots or the master robot deciding slaves' action [11,39]. To avoid collisions, each of these techniques use continuous position and velocity measurements of the target robot in the field of view. Besides this, the master-slave approach [11, 39] is highly dependent on the communication link established between the master and slaves. However, collision avoidance with minimal sensing in a system with decentralized control has the advantage of reduced sensor data processing and computation overhead on the robots.

Next chapter introduces the fundamental concepts of algebraic graph theory that are used in the proposed IMRDFS strategy.

Chapter 3

Preliminaries on Graph Based Exploration and Mapping

Multi robot exploration in unknown environments has been an active area of research in mobile robotics. A topological representation of the environment, referred to as a *map* is needed for safer and efficient navigation of the robots. Any unknown environment populated with obstacles can be modeled as a graph with free paths and junctions represented by the edges and vertices of the graph, respectively. Therefore, representing an unknown environment using a graph is a finest solution to mapping. The work presented in this thesis is based on the algebraic graph theory aspects and vector space associated with graphs. The following sections present the graph theory terminology and the association of vector space with a graph in detail.

3.1 Terminology

Terminology used in this work is illustrated using Fig. 3.1. Obstacles are shown using gray boxes. Let the graph to be explored contains E edges and V vertices. The vertices are represented by v_i for $i \in 1, 2, \ldots, V$ and edges are represented by e_{ij} for $i, j \in 1, 2, \ldots, V$. The graph is explored using K number of robots starting from the same location termed as *root vertex*, v_r. The k^{th} robot is referred as R_k.

Robots move along edges of the graph and construct their local graphs. If a robot visits a vertex, where no beacon exists, it drops a new beacon at that vertex. In particular, a robot drops a beacon on the first visit of a vertex and the robot stores its partially explored graph information at the beacon. Information of a beacon and the robot are updated, when the robot

visits a vertex. Fig. 3.1 shows a graph with 8 vertices and 10 edges representing free space of an environment. Consider, two robots R_1 and R_2 start the exploration from vertex v_1 ($v_r = v_1$) and a particular instance during exploration where the robot R_1 traversed e_{12}, e_{24} and e_{46} edges in sequence while the robot R_2 traversed e_{13}, e_{34} and e_{43} edges in sequence. Edges shown with dotted lines are still unexplored and one shown with dash dotted line (edge e_{78}) is unidentified by any robot.

The terminology of graph theory and exploration used in this work is in the same spirit as in [1, 49]. It is presented here for completeness with the help of Fig. 3.1.

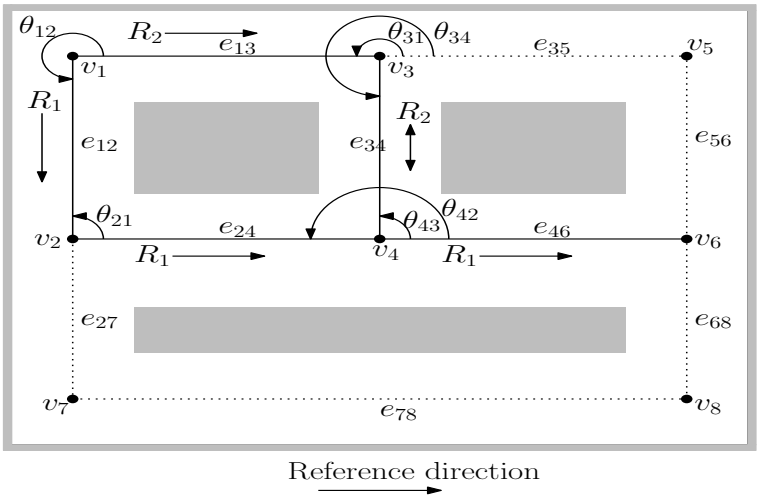


Figure 3.1: Illustration of terminology and proposed strategy

- *Vertex:* A vertex v_i for $i \in 1, 2, \ldots, V$ is a terminal point or an interconnecting point in a graph. Fig. 3.1 shows vertices as v_i for $i \in \{1, 2, 3, \ldots, 8\}$.

- *Edge:* An edge e_{ij} for $i, j \in 1, 2, \ldots, V$ is a connecting path between two vertices v_i and v_j. The edges are represented by e_{ij} for $i, j \in \{1, 2, 3, \ldots, 8\}$ in Fig. 3.1.

- *Graph:* A graph is an ordered pair $G = (\mathcal{V}, \mathcal{E})$ where, $\mathcal{V}$ is the set of vertices whose elements are the vertices of the graph and $\mathcal{E}$ is the set of edges whose elements are the edges of the graph.

- *Subgraph:* A graph $G' = (\mathcal{V}', \mathcal{E}')$ is a subgraph of a graph $G = (\mathcal{V}, \mathcal{E})$, if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$.

- *Cycle:* A cycle is a closed walk in the graph with no repetitions of vertices or edges except the repetition of starting and ending vertex. In Fig. 3.1, the vertex sequence {v_1, v_2, v_4,

v_3, v_1} represents a cycle.

- *Radius:* The radius r of a tree is the maximum distance between the root vertex and any other vertex in the tree.

- *Spanning Tree:* A spanning tree T of a graph G is a tree of G having all the vertices of G. For a graph with n vertices, any spanning tree contains exactly $n-1$ edges.

- *Set of unexplored edges $\mathcal{E}_u^n(.)$:* An edge that has been identified but not yet traversed by any robot is an *unexplored edge*. In Fig. 3.1, edge e_{35} contains the vertex information at one end, but it is not yet traversed. Therefore, it is an *unexplored* edge. This edge is connected at vertex v_3, but the other vertex is unknown at the instance being considered (edge terminology e_{35} is used for referring in Fig. 3.1).

  The set of unexplored edges available with a robot R_k after n^{th} update is referred as $\mathcal{E}_u^n(R_k)$ and that stored at a vertex v_j is referred as $\mathcal{E}_u^n(v_j)$.

- *Set of out edges $\mathcal{E}_o^n(.)$:* An out edge is an edge e_{ij} which is traversed by at least one robot, but only its source vertex information is available (no knowledge of destination vertex). For example, (referring to Fig. 3.1) when robot R_1 takes the decision of traversing edge e_{46} from vertex v_4, the destination vertex information of edge e_{46} at vertex v_4 is unavailable even when the robot reaches at vertex v_6. Therefore, edge e_{46} is an *out edge* for vertex v_4.

  A set $\mathcal{E}_o^n(R_k)$ is the set of out edges available with the robot R_k after n^{th} update and a set $\mathcal{E}_o^n(v_j)$ is the set of out edges stored at a beacon of vertex v_j after n^{th} update.

- *Set of completed edges $\mathcal{E}_c^n(.)$:* An edge e_{ij} is a completed edge when its both the vertices v_i and v_j are known. When the robot R_1 visits vertex v_2 in Fig. 3.1, both the vertices v_1 and v_2 are known for edge e_{12}. Therefore, status of this edge is *completed* at vertex v_2. The sets of completed edges available with a robot R_k and that stored at a vertex v_j after n^{th} edge traversal are the sets $\mathcal{E}_c^n(R_k)$ and $\mathcal{E}_c^n(v_j)$ respectively.

- *Set of completed vertices $\mathcal{V}_c^n(.)$:* A vertex which is visited by at least one robot is termed as completed vertex (for consistency in terminology with completed edge). The sets $\mathcal{V}_c^n(R_k)$ and $\mathcal{V}_c^n(v_j)$ are the sets of completed vertices available respectively with a robot R_k and at a vertex v_j after n^{th} update.

- *Partially completed graph $\mathcal{G}_c^n(.)$:* The sets $G_c^n(R_k)$ and $G_c^n(v_j)$ are ordered pairs $(\mathcal{V}_c^n(R_k), \mathcal{E}_c^n(R_k))$ and $(\mathcal{V}_c^n(v_j), \mathcal{E}_c^n(v_j))$ respectively and represent partially completed graph available with the robot R_k and the vertex v_j respectively after n^{th} update.

- *Ordered set of my-unexplored edges $\mathcal{E}_m^n(.)$:* The my-unexplored edges stored at a vertex v_j are the edges that are connected to vertex v_j and have not been traversed by any robot. However, the my-unexplored edges for a robot R_k are the unexplored edges that are identified by the robot R_k itself while visiting the vertices. Since, the robot and vertex exchange information, both will also have information about the unexplored edges identified by other robots. Therefore, the set of my-unexplored edges are a subset of the set of unexplored edges stored at a vertex or available with the robot. In particular, $\mathcal{E}_m^n(R_k) \subseteq \mathcal{E}_u^n(R_k)$ and $\mathcal{E}_m^n(v_j) \subseteq \mathcal{E}_u^n(v_j)$. The sets $\mathcal{E}_m^n(R_k)$ and $\mathcal{E}_m^n(v_j)$ are the sets of my-unexplored edges of a robot R_k and a vertex v_j respectively. The superscript n corresponds to n^{th} update of these sets.

- *Incidence angle:* It is the angle $\theta_{ji} \in (0, 2\pi]$ subtended by an edge e_{ji} at a vertex v_j in counter clockwise direction measured with respect to a reference direction. Note that the zero angle is represented by 2π. For example, angle θ_{31} shown in Fig. 3.1, is the incidence angle made by edge e_{31} at vertex v_3.

- *Incidence matrix $\mathcal{I}^n(.)$:* The definition of incidence matrix [49] is modified in our proposed strategy in the following way: Each element d_{ij} of the incidence matrix I is defined as,

$$d_{ij} = \begin{cases} -\theta_{ij} & \text{if } e_{ij} \in \mathcal{E}_o(.) \text{ or } e_{ij} \in \mathcal{E}_c(.), \\ +\theta_{ij} & \text{if } e_{ij} \in \mathcal{E}_u(.), \\ 0 & \text{otherwise.} \end{cases}$$

  A column in the incidence matrix represents the edges connected to a vertex and a row represents an explored vertex of the graph. The proposed modification in incidence matrix encodes the information of sets $\mathcal{V}_c(.), \mathcal{E}_c(.), \mathcal{E}_o(.), \mathcal{E}_u(.)$ and $\mathcal{E}_m(.)$. Edge category (unexplored, completed or out) is identified by the column vectors of the modified incidence matrix I. A completed edge is represented by a column with two non-zero elements, an unexplored edge by a column with only one positive element and an out edge by a column with only one negative element.

  The incidence matrices $I^n(R_k)$ and $I^n(v_j)$ are the incidence matrices available with a

robot R_k and stored at a vertex v_j respectively after n^{th} update. Referring to Fig. 3.1, the incidence matrix obtained at vertex v_4 after fourth update is given by,

$$I^4(v_4) = \begin{array}{c} \\ v_4 \\ v_2 \\ v_3 \\ v_1 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_{34} & e_{24} & e_{12} & e_{13} & e_{46} & e_{27} & e_{35} \end{array} \\ \begin{bmatrix} -\pi/2 & -\pi & 0 & 0 & 2\pi & 0 & 0 \\ 0 & -2\pi & -\pi/2 & 0 & 0 & 3\pi/2 & 0 \\ -3\pi/2 & 0 & 0 & -\pi & 0 & 0 & 2\pi \\ 0 & 0 & -3\pi/2 & -2\pi & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The representation of the edges in the graph shown in Fig. 3.1 is redefined to ease the vector space analysis. The edges e_{13}, e_{12}, e_{24}, e_{27}, e_{78}, e_{34}, e_{46}, e_{56}, e_{68} are represented as e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9 respectively. Consider, initial placement of two robots R_1 and R_2 at vertices v_1 and v_8 respectively and a particular instance during exploration where the robot R_1 traversed $\mathrm{e}_2(e_{12})$, $\mathrm{e}_3(e_{24})$ in sequence and now traversing $\mathrm{e}_7(e_{46})$ edge. While the robot R_2 traversed $\mathrm{e}_5(e_{87})$, $\mathrm{e}_4(e_{72})$ in sequence and now traversing $\mathrm{e}_2(e_{21})$ edge.

A subgraph X of a graph $\mathcal{G}$ is an *induced subgraph*, if the following is true for all vertices in X: any two vertices of graph X are adjacent in X if and only if they are adjacent in $\mathcal{G}$. A *component* [49] of a graph $\mathcal{G}$ is an induced subgraph of $\mathcal{G}$ that is maximal and connected. The partially explored graph is a subgraph of the graph $\mathcal{G}$. It consists of one or multiple components. It can have multiple components because robots may have the capability to exchange their partially explored graphs and all robots may not start from the same place.

While exploring the unknown graph, there are edges for which a terminal vertex v_i is known, but the other vertex is unknown, these edges are referred as *half edges* [50], $e_{i\mathrm{x}}$. Incidence matrix is used in this work to store the partially explored graph. A column in the incidence matrix [49] represents an edge connected to the explored vertex and a row represents an explored vertex of the graph. The incidence angle is used to modify elements of an incidence matrix [51]. Incidence angle is the angle $\theta_{ji} \in (0, 2\pi]$ subtended by an edge e_i at a vertex v_j in counter clockwise direction measured with respect to a reference direction. The zero angle is represented by 2π. For example, angle θ_{12} shown in Fig. 3.1, is the incidence angle made by edge e_2 at vertex v_1.

The algebraic graph theory [49] builds the vector spaces [52] of a graph. Next section presents preliminaries for algebraic graph theory used to establish results in this work.

3.2 Vector Space associated with a Graph

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is associated with the edge and vertex spaces [49]. In graph theory [53], the edge space and vertex space of an undirected graph are vector spaces defined in terms of the edge and vertex sets, respectively. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a finite undirected graph. The vertex space $\mathcal{W}_\mathcal{V}$ of $\mathcal{G}$ is the vector space over the finite field of two elements $\mathcal{F}_2 := \{0, 1\}$ of all functions $\mathcal{V} \rightarrow \mathcal{F}_2$. Every element of $\mathcal{W}_\mathcal{V}$ naturally corresponds to the subset of $\mathcal{V}$ which assigns a 1 to its vertices. Also every subset of $\mathcal{V}$ is uniquely represented in $\mathcal{W}_\mathcal{V}$. The set of all the subsets of $\mathcal{E}$, the edge space $\mathcal{W}_\mathcal{E}$ is the vector space generated by the edge set $\mathcal{E}$ over the field $\mathcal{F}_2 = \{0, 1\}$ under ring sum (vector addition) and modulo 2 sum (scalar multiplication) operations. The dimension of the vertex space is thus the number of vertices of the graph, while the dimension of the edge space is the number of edges. Given two subgraphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$, the edge sets $\mathcal{E}_1$ and $\mathcal{E}_2$ represent two vectors in the edge space $\mathcal{W}_\mathcal{E}$. The ring sum $\mathcal{E}_1 \oplus \mathcal{E}_2$ is given by,

$$\mathcal{E}_1 \oplus \mathcal{E}_2 = (\mathcal{E}_1 \cup \mathcal{E}_2) - (\mathcal{E}_1 \cap \mathcal{E}_2). \tag{3.1}$$

Remark 3.1 The ring sum of any two edge sets of $\mathcal{E}$ is a subset of $\mathcal{E}$ (vector addition of two vectors is a vector itself).

Example 3.2 *Let $\mathcal{E}_1$ = (1,1,0,0) and $\mathcal{E}_2$ = (1,0,1,0) be two edge sets of the set $\mathcal{E}$ = (1,1,1,1). The edge set $\mathcal{E}_3$ obtained from the ringsum of the edge sets $\mathcal{E}_1$ and $\mathcal{E}_2$ is given by,*

$$\begin{aligned} \mathcal{E}_3 &= \mathcal{E}_1 \oplus \mathcal{E}_2, \\ &= (1, 1, 0, 0) \oplus (1, 0, 1, 0), \\ &= (0, 1, 1, 0), \\ &\subseteq (1, 1, 1, 1). \end{aligned} \tag{3.2}$$

For $\mathcal{E} = \{\mathrm{e}_1, \mathrm{e}_2, \ldots, \mathrm{e}_E\}$, the subsets $\{\mathrm{e}_1\}$, $\{\mathrm{e}_2\}$, ..., $\{\mathrm{e}_E\}$ constitutes a fundamental basis for $\mathcal{W}_\mathcal{E}$. Also, the dimension of the vector space $\mathcal{W}_\mathcal{E}$ is given by,

$$\dim(\mathcal{W}_\mathcal{E}) = E. \tag{3.3}$$

Without loss of generality, we use $\mathcal{E}_A$ to represent both the edge set of a subgraph $\mathcal{G}_A \subset \mathcal{G}$ as well as a vector in the edge space $\mathcal{W}_\mathcal{E}$. It is worth noting that edge set of any subgraph of

the graph $\mathcal{G}$ represents a subspace of the vector space. An edge set $\mathcal{E}_A$ of $\mathcal{E}$ is a vector in vector space $\mathcal{W}_\mathcal{E}$ represented by the linear combination of its basis elements $\mathcal{E}_A = \sum_{i=1}^{E} f_i \mathrm{e}_i$, where $f_i \in \mathcal{F}_2$ and given by

$$f_i = \begin{cases} 1 & \text{if } \mathrm{e}_i \in \mathcal{E}_A, \\ 0 & \text{otherwise.} \end{cases}$$

There exists 2^E possible vectors in $\mathcal{W}_\mathcal{E}$ including the *zero* vector $\vec{0}$ (each $f_i = 0$ corresponds to the null edge set) and the *one* vector $\vec{1}$ (each $f_i = 1$ corresponds to the edge set $\mathcal{E}$ itself).

3.3 Conclusion

This chapter presented the terminology related to graph theory and vector space used in this work. The concepts from graph theory and incidence matrix are used to develop and analyze the IMRDFS exploration strategy detailed in chapter 4. The association of vector space with a graph is adopted to propose the generic framework for multi robot exploration and mapping discussed in chapter 5.

In order to achieve faster and efficient completion, the movement of robots on unvisited portions of the graph should be coordinated by means of sharing the partial maps of the explored environment. Coordination becomes more difficult if the communication scheme is decentralized. The next chapter proposes a multi robot exploration and mapping strategy, the *improved multi robot depth first search (IMRDFS)* using a decentralized communication framework.

Chapter 4

Improved Multi Robot Depth First Search (IMRDFS) Strategy

Coordinating multiple robots to carry out the mapping and exploration task is challenging. The proposed multi robot exploration and mapping strategy consider that robots are initially located at the root vertex of an unknown undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, formed by V vertices and E edges, where $\mathcal{V}$ and $\mathcal{E}$ represent sets of vertices and edges, respectively. Robots move along the free paths (edges) in the environment and locally construct a graph of their own. The robots drop a beacon at every newly visited vertex (first robot arrival) and the beacon information assists further exploration.

The proposed improved multi robot depth first search (IMRDFS) strategy is the MRDFS algorithm to perform parallel search of an unknown environment by multiple robots. The primary idea is to traverse edges that are explored by self first in depth first manner and to obtain the map of the environment in the form of a modified incidence matrix. An edge assignment scheme has been proposed to distinguish explored graph portions from the unexplored graph portions to minimize redundant exploration. As long as there are unexplored and out edges, the robots select one of those edges to explore. Only if all the edges are completed, the robots signal completion of exploration.

The following assumptions have been made in proposing the strategy:

1. Environment is bounded, but unknown apriori
2. Robots carry unlimited number of beacons
3. A beacon is dropped at each unvisited (new) vertex

4. Robot-robot and beacon-beacon communication is not possible or does not exist

5. Robot-beacon local communication is possible only at a vertex

6. Each robot is able to localize itself with respect to its partial map.

Since, inter robot and inter beacon communication does not exists, the algorithm establishes a decentralized framework for multi robot exploration. The next section details the proposed IMRDFS strategy for exploration and mapping.

4.1 Improved Multi Robot Depth First Search (IMRDFS) Strategy

The robots store the partially explored map information as incidence matrices and these partially developed map information is exchanged through the vertex beacons dropped by a robot visiting a vertex for the first time. The partially explored graph is stored as an incidence matrix. The information of partial graph (represented by $G_c(.)$, $\mathcal{E}_o(.)$, $\mathcal{E}_u(.)$ and $\mathcal{E}_m(.)$ as explained in chapter 3 section 3.1) explored by other robot is exchanged only when a robot visits a vertex having a beacon. The robots visit various vertices at different instances of the exploration, therefore information available with a beacon may not be the latest information. The proposed approach for multi robot graph exploration aims for a systematic exploration providing independent decision power to each robot, when information is exchanged with the beacon at a vertex.

The modified incidence matrix data structure facilitates a systematic exploration by determining the next edge of exploration. Each robot selects an edge based on its priority from it's partial map. The robots first completely explore their own partial graph by traversing the my-unexplored edges $m_u(own)$. Only when all the my-unexplored edges are completed, the edges $m_u(other)$ or m_o are chosen to complete the exploration. The proposed edge selection scheme is responsible for the robot coordination. Once an edge is selected by a robot for exploration, the status of the edge is modified based on its current state. Following status modifications are possible for the edges:

- unexplored edge(e_u) to an out edge(e_o),
- unexplored edge(e_u) to a completed edge(e_c),

- out edge(e_o) to a completed edge(e_c).

The updated edge status for every edge traversal prevent redundant exploration. It also ensures coordination among the robots and accelerates the exploration. If two or more robots are present at any vertex, then the robots select the edges sequentially. If an unexplored edge e_{ij} at a vertex v_i is chosen by a robot R_k, then it's status is changed to an out edge at v_i. So, for the other robots, the edge e_{ij} is no longer an unexplored edge.

A graph $G(\mathcal{V}, \mathcal{E})$ represents an environment populated with obstacles. The paths (free spaces) represent edges and the junctions represent vertices of the graph. Robots drop the beacons at these vertices, which store the partial map information as incidence matrices. The merge of two incidence matrices at a vertex (robot and the vertex), gives the updated partial graph. The explored graph information is updated through merging and a completed graph is obtained in finite edge traversals. As the number of robots exploring the environment increases, the edge traversal count to completion decreases.

The initializations of vertex and edge sets for each robot $R_k, k = 1, \ldots, K$ at root vertex v_r are as follows:

$$G_c^0(R_k) = (\{v_r\}, \emptyset)\,,\ \mathcal{E}_o^0(R_k) = \mathcal{E}_u^0(R_k) = \mathcal{E}_m^0(R_k) = \emptyset.$$

When a robot visits a vertex v_j which is never explored, it drops a beacon with a unique ID (for instance, the robot R_1 gives IDs as $a_1, a_2, a_3, \cdots$, the robot R_2 gives IDs as $b_1, b_2, b_3, \cdots$, and so on) and initialize the sets as follows:

$$G_c^0(v_j) = (\{v_j\}, \emptyset)\,, \mathcal{E}_o^0(v_j) = \emptyset \text{ and}$$

$$\mathcal{E}_m^0(v_j) = \mathcal{E}_u^0(v_j) = \{e_{pq} \mid v_p = v_j\}.$$

A robot identifies edges $e_{pq} \in G$ for which one vertex is the vertex $v_p = v_j$, when it visits vertex v_j. But, the other vertex v_q is not visited. Therefore, when a beacon is initialized for vertex v_j, it is initialized with all the edges connected to vertex v_j as unexplored. The exploration begins at root vertex v_r, therefore first vertex visited by each robot is v_r.

Let n^{th} update instance be the n^{th} edge traversal of graph G by any robot reaching to any vertex. *It is worth noting that the count n differentiates between the updates, it is not required for implementation.* Let the robot R_k visits vertex v_j by traversing an edge e_{ij} at n^{th} update instance. For maintaining consistency in representation, we consider updating each set associated with every robot and vertex at n^{th} update instance. The n^{th} update of each set associated with

a robot R_k ($G_c^n(R_k), \mathcal{E}_m^n(R_k), \mathcal{E}_u^n(R_k)$ and $\mathcal{E}_o^n(R_k)$) is modified when the robot R_k visits any vertex. Similarly, n^{th} update of each set associated with a vertex v_j ($G_c^n(v_j), \mathcal{E}_m^n(v_j), \mathcal{E}_u^n(v_j)$ and $\mathcal{E}_o^n(v_j)$) is modified when any robot visits it. These updates are as follows: The n^{th} updates for robots $R_p, p \neq k$ and vertices $v_i, i \neq j$ are given by (4.1).

$$\begin{aligned} G_c^n(R_p) &= G_c^{n-1}(R_p), \quad & G_c^n(v_i) &= G_c^{n-1}(v_i), \\ \mathcal{E}_o^n(R_p) &= \mathcal{E}_o^{n-1}(R_p), \quad & \mathcal{E}_o^n(v_i) &= \mathcal{E}_o^{n-1}(v_i), \\ \mathcal{E}_u^n(R_p) &= \mathcal{E}_u^{n-1}(R_p), \quad & \mathcal{E}_u^n(v_i) &= \mathcal{E}_u^{n-1}(v_i), \\ \mathcal{E}_m^n(R_p) &= \mathcal{E}_m^{n-1}(R_p) \text{ and} \quad & \mathcal{E}_m^n(v_i) &= \mathcal{E}_m^{n-1}(v_i). \end{aligned} \tag{4.1}$$

Remark 4.1 For the first visit of a vertex v_j at n^{th} update instance, $G_c^{n-1}(v_j) = G_c^0(v_j)$, $\mathcal{E}_o^{n-1}(v_j) = \mathcal{E}_o^0(v_j)$, $\mathcal{E}_u^{n-1}(v_j) = \mathcal{E}_u^0(v_j)$ and $\mathcal{E}_m^{n-1}(v_j) = \mathcal{E}_m^0(v_j)$. In particular, there is no change in the sets because no robot visited vertex v_j till $(n-1)^{th}$ update instance.

The n^{th} updates for the partially completed graphs of the robot R_k and vertex v_j are given by (4.2).

$$\begin{aligned} G_c^n(v_j) &= G_c^n(R_k) \\ &= G_c^{n-1}(R_k) \cup (\{v_i, v_j\}, \{e_{ij}\}) \cup G_c^{n-1}(v_j). \end{aligned} \tag{4.2}$$

where the robot R_k completed an edge e_{ij} of graph G as n^{th} edge traversal. In particular, the robot R_k reaches a vertex v_j by traversing the edge e_{ij} and the edge e_{ij} be the n^{th} edge traversal of the graph G. The subset $G_c^n(.)$ is updated using (4.2). The sets of unexplored $\mathcal{E}_u^n(.)$, $\mathcal{E}_m^n(.)$ and out edges $\mathcal{E}_o^n(.)$ for the robot R_k and vertex v_j are updated in two steps.

In first step, the sets are updated by merging. The robot R_k takes decision and selects an edge for next traversal based on this merged information. The information about selected edge for traversal is updated in the second step. The sets of unexplored and out edges on n^{th} edge traversal of graph G are updated in two steps. In first step, sets are merged using (4.3).

$$\begin{aligned} \mathcal{E}_o^n(v_j) &= \mathcal{E}_o^{n-1}(v_j) \cup \mathcal{E}_o^{n-1}(R_k) \setminus \mathcal{E}_c^n(v_j), \\ \mathcal{E}_u^n(v_j) &= \mathcal{E}_u^{n-1}(v_j) \cup \mathcal{E}_u^{n-1}(R_k) \setminus (\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)), \\ \mathcal{E}_m^n(v_j) &= \mathcal{E}_m^{n-1}(v_j) \setminus (\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)), \text{ and} \\ \mathcal{E}_m^n(R_k) &= \mathcal{E}_m^{n-1}(v_j) + \mathcal{E}_m^{n-1}(R_k) \setminus (\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)) \end{aligned} \tag{4.3}$$

where $+$ represents ordered sum of sets.

The robot takes a decision using edge information in the sets $\mathcal{E}_u^n(R_k), \mathcal{E}_o^n(R_k)$ and $\mathcal{E}_m^n(R_k)$ given by (4.3). The proposed DFS strategy dictates the robot to choose an edge based on the following order of priority:

$$\{\mathcal{E}_m^n(R_k), \mathcal{E}_u^n(R_k), \mathcal{E}_o^n(R_k)\}, \tag{4.4}$$

with highest priority given to an edge from $\mathcal{E}_m^n(R_k)$. The second priority is given to the unexplored edges identified by other robots. These edges are obtained from the set $\mathcal{E}_u^n(R_k) \setminus \mathcal{E}_m^n(R_k)$. This priority assignment enables a systematic exploration of the environment. A robot initially explores its own unexplored portion of the graph by traversing each unexplored edge from $\mathcal{E}_m^n(R_k)$ and then helps other robots by choosing the edges from $\mathcal{E}_u^n(R_k)$ or $\mathcal{E}_o^n(R_k)$ to complete the exploration. In particular, a robot prefers to explore as a single robot DFS strategy initially and then helps others.

In the second step, the edge for next traversal is identified based on the priority described in (4.4). The information about selected edge is updated in second step. Let an edge e_{lm} be selected based on the priority described in (4.4) for next edge traversal, the second step update is described by (4.5).

$$\begin{aligned}
\mathcal{E}_o^n(R_k) &= \mathcal{E}_o^n(v_j) = \mathcal{E}_o^n(v_j) \cup \{e_{lm}\}, \\
\mathcal{E}_u^n(R_k) &= \mathcal{E}_u^n(v_j) = \mathcal{E}_u^n(v_j) \setminus \{e_{lm}\}, \\
\mathcal{E}_m^n(v_j) &= \mathcal{E}_m^n(v_j) \setminus \{e_{lm}\} \text{ and} \\
\mathcal{E}_m^n(R_k) &= \mathcal{E}_m^n(R_k) \setminus \{e_{lm}\}.
\end{aligned} \tag{4.5}$$

If the beacon at a vertex v_j initiates the update, simultaneous update of multiple robots visiting vertex v_j can be addressed. From (4.3), the set $\mathcal{E}_m^n(.)$ is updated with a possible removal of edges, therefore

$$\max\left(|\mathcal{E}_m^n(v_j)|\right) = \left|\mathcal{E}_m^0(v_j)\right|.$$

We also have $\mathcal{E}_u^0(v_j) = \mathcal{E}_m^0(v_j)$. If $e_{ij} \in \mathcal{E}_m^n(v_j)$ then $e_{ij} \in \mathcal{E}_u^n(v_j)$, therefore,

$$\mathcal{E}_m^n(v_j) \subseteq \mathcal{E}_u^n(v_j). \tag{4.6}$$

When a robot R_k visits a vertex v_j, we have $\mathcal{E}_u^n(v_j) = \mathcal{E}_u^n(R_k)$. Therefore, from (4.6), we get

$$\mathcal{E}_m^n(v_j) \subseteq \mathcal{E}_u^n(R_k).$$

We also have $\mathcal{E}_m^0(R_k) \subseteq \mathcal{E}_u^0(R_k)$. Therefore, from (4.5), we get

$$\mathcal{E}_m^n(R_k) \subseteq \mathcal{E}_u^n(R_k). \tag{4.7}$$

The primary goal of any multi robot exploration strategy is to guarantee finite time exploration completion. Efficient coordination, edge selection and partial map updates helps to achieve this objective. The next section present analytical results on the finite time completion of the proposed IMRDFS strategy.

4.2 Analytical Results

The completion of graph exploration using proposed technique is shown through following lemma and theorem.

Lemma 4.2 *Partially completed graph at any vertex v_j, $G_c^n(v_j)$ for any n, is a connected subgraph of G.*

Proof Let a set $\mathcal{G}^n$ be defined using (4.8).

$$\mathcal{G}^n = \{G_c^n(v_j)\} \ \forall\, j = 1, 2, \ldots, V^1. \tag{4.8}$$

We prove that each element of $\mathcal{G}^n$ for $n = 0, 1, 2, \ldots$ is connected subgraph of G. Let a robot R_k reached vertex v_i at p^{th} update instance and selects to traverse an edge e_{ij} at vertex v_i. The p^{th} update of partially completed graph at vertex v_i is obtained using (4.2). Therefore, $G_c^p(R_k) = G_c^p(v_j)$. Let the edge e_{ij} be completed by the robot R_k at $n^{th}(n > p)$ update instance (other robots would have completed $(n - p)$ edge traversals). Therefore, the robot R_k reaches vertex v_j with

$$G_c^{n-1}(R_k) = G_c^p(v_i). \tag{4.9}$$

From (4.9) and (4.2), each element of $\mathcal{G}^n$ is updated using (4.10).

$$G_c^n(v_j) = \begin{cases} G_c^p(v_i) \cup (\{v_i, v_j\}, \{e_{ij}\}) \cup G_c^{n-1}(v_j) \ \text{ when} \\ \text{an edge } e_{ij} \text{ is completed by any robot} \\ G_c^{n-1}(v_j) \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{4.10}$$

When the exploration begins, none of the edge is completed. Therefore,

$$\mathcal{G}^0 = \{(\{v_1\}, \emptyset), (\{v_2\}, \emptyset), \ldots, (\{v_V\}, \emptyset)\} \tag{4.11}$$

We prove that $G_c^n(v_j)$ is a subgraph and it is connected using induction.

[1]Even though beacons are not placed at unvisited vertices at the beginning of the exploration, they are initialized with the sets at $n = 0$ at the first visit of a respective vertex.

Let $n = 0$, (4.11) gives $G_c^0(v_j) = (\{v_j\}, \emptyset)$ for all $j = 1, \ldots, V$. Since $\{v_j\} \in \mathcal{V}$ and $\emptyset \in \mathcal{E}$, $G_c^0(v_j)$ is a subgraph. Also, $G_c^0(v_j)$ represents a set with single vertex, it is a graph with no edges. Therefore, each element of $\mathcal{G}^0$ is connected.

When $n = 1$, $G_c^1(v_j)$ is updated using (4.10). We have following two possibilities:

- If $G_c^1(v_j) = G_c^0(v_j)$, then $G_c^1(v_j)$ is a connected subgraph because $G_c^0(v_j)$ is a connected subgraph.

- If $G_c^1(v_j) = G_c^0(v_i) \cup (\{v_i, v_j\}, e_{ij}) \cup G_c^0(v_j)$, (because an edge e_{ij} is traversed), then $v_i \in G_c^0(v_i)$, $v_j \in G_c^0(v_j)$ and an edge e_{ij} connects vertices v_i and v_j. Since $G_c^0(v_i)$ and $G_c^0(v_j)$ are connected, $G_c^1(v_j)$ is connected. Since $e_{ij} \in \mathcal{E}$, $G_c^0(v_i)$ and $G_c^0(v_j)$ are subgraphs, therefore $G_c^1(v_j)$ is a subgraph.

Therefore, each element of $\mathcal{G}^1$ is a connected subgraph, because each element of $\mathcal{G}^0$ is a connected subgraph. Similarly, each element of $\mathcal{G}^2$ is a connected subgraph, because each element of $\mathcal{G}^0$ and $\mathcal{G}^1$ is a connected subgraph. Following the same reasoning, each element of $\mathcal{G}^n \; \forall \; n = 1, 2, \ldots$ is a connected subgraph, because each element of $\mathcal{G}^p \; \forall \; p = 0, 1, \ldots, (n-1)$ is a connected subgraph.

■

Example 4.3 *In Fig. 4.1, the graph $G_c^4(R_1)$ (with edges shown using solid and dotted lines) represents the partial map obtained by the robot R_1 after four edge traversals.*

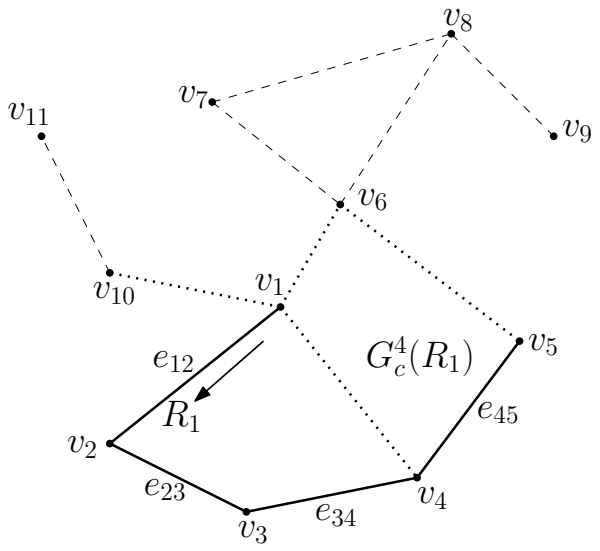


Figure 4.1: Partially completed graph

The partially completed graph $\mathcal{G}^p$ contains at least one unexplored/out edge. The graph is said to be completely explored, if all the edges in the graph are explored. Theorem 4.4 presented next establishes the condition for exploration completion.

Theorem 4.4 *Exploration is complete if and only if*

$$\mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k) = \emptyset \textit{ for any } k = 1, \ldots, K.$$

Proof Necessary part:
From the definitions of $\mathcal{E}_u^n(R_k)$, $\mathcal{E}_o^n(R_k)$ and $\mathcal{E}_c^n(R_k)$, an edge

$$\begin{aligned} e_{ij} &\in \{\mathcal{E}_u^n(R_k), \mathcal{E}_o^n(R_k)\} \text{ if either } v_i \text{ or } v_j \text{ is unexplored,} \\ e_{ij} &\in \{\mathcal{E}_c^n(R_k)\} \text{ if both } v_i \text{ or } v_j \text{ are explored.} \end{aligned}$$

Let $\mathcal{E}_x^n(R_k)$ be the set of edges $\{e_{ij}\}$ (unknown to R_k) for which both the connecting vertices v_i and v_j are unexplored. Therefore,

$$\mathcal{E} = \{\mathcal{E}_c^n(R_k), \mathcal{E}_o^n(R_k), \mathcal{E}_u^n(R_k), \mathcal{E}_x^n(R_k)\}. \tag{4.12}$$

Let $\mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k) = \emptyset$ holds, we have

$$\mathcal{E} = \{\mathcal{E}_c^n(R_k), \mathcal{E}_x^n(R_k)\}. \tag{4.13}$$

When $\mathcal{E} = \emptyset$, $\mathcal{E}_c^0(R_k) = \emptyset$ and $\mathcal{E}_x^0(R_k) = \emptyset$ from (4.13). An empty set of edges in a connected non-empty graph has one vertex and the exploration is complete at this vertex.
When $\mathcal{E} \neq \emptyset$, we claim that $\mathcal{E}_c^n(R_k) \neq \emptyset$ and $\mathcal{E}_x^n(R_k) \neq \emptyset$ can not happen simultaneously. We prove this by contradiction.
Let $\mathcal{E}_c^n(R_k) \neq \emptyset$ and $\mathcal{E}_x^n(R_k) \neq \emptyset$. Since the graph is connected, $\exists\ e_{ij} \in \mathcal{E}_x^n(R_k)$ and $e_{kl} \in \mathcal{E}_c^n(R_k)$ such that one of the following conditions is true:

$$v_i = v_k, v_i = v_l, v_j = v_k, v_j = v_l.$$

Since v_i and v_j are unexplored and v_k and v_l are explored, it contradicts the assumption $\mathcal{E}_c^n(R_k) \neq \emptyset$ and $\mathcal{E}_x^n(R_k) \neq \emptyset$.
Therefore, either $\mathcal{E}_c^n(R_k) \neq \emptyset$ or $\mathcal{E}_x^n(R_k) \neq \emptyset$ is true. In particular, we have two possibilities:

- *Case 1:* $\mathcal{E}_x^n(R_k) \neq \emptyset \implies \mathcal{E}_c^n(R_k) = \emptyset$. From (4.13), we get $\mathcal{E}_x^n(R_k) = \mathcal{E}$ which is true only at start of the exploration.

- *Case 2:* $\mathcal{E}_c^n(R_k) \neq \emptyset \implies \mathcal{E}_x^n(R_k) = \emptyset$. From (4.13), we get $\mathcal{E}_c^n(R_k) = \mathcal{E}$ which shows that all the edges are explored.

Case 2 shows that the exploration is complete. *Case 1* only occurs at start of the exploration when $\mathcal{E}_u^0(.) = \mathcal{E}_o^n(.) \neq \emptyset$ for a graph with more than one vertex, therefore the condition $\mathcal{E}_u^n(R_k) = \mathcal{E}_o^n(R_k) = \emptyset$ is true only when all the edges are completed.

Sufficient part: Assume that the exploration is complete. All the edges are explored, therefore $\mathcal{E}_x^n(R_k) = \emptyset$ and $\mathcal{E}_c^n(R_k) = \mathcal{E}$. From (4.12), we have $\mathcal{E}_u^n(R_k) = \mathcal{E}_o^n(R_k) = \emptyset$. ■

Corollary 4.5 *$\mathcal{E}_u^n(v_j) \cup \mathcal{E}_o^n(v_j) = \emptyset$ for any $j = 1, \ldots, V$ only if $\mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k) = \emptyset$ for at least one robot k, $k = 1, \ldots K$.*

Proof From (4.2), we know that $G_c^n(v_j) = G_c^n(R_k)$, when the robot R_k visits the vertex v_j. Therefore,

$$\mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k) \neq \emptyset \implies \mathcal{E}_u^n(v_j) \cup \mathcal{E}_o^n(v_j) \neq \emptyset.$$

■

Example 4.6 *Corollary 4.5 is demonstrated using Fig. 4.2, where robots R_1 and R_2 start the exploration from vertex v_1. Sequences of vertices visited by the robots R_1 and R_2 are $v_1 - v_2 - v_3 - v_4 - v_1 - v_3$ and $v_1 - v_6 - v_5 - v_4 - v_5 - v_6$ respectively. After 5 edge traversals, robot R_1 reaches vertex v_3 with a partial graph $G_c^5(R_1)$ that represents the complete graph. Hence,*

$$\mathcal{E}_u^5(R_1) \cup \mathcal{E}_o^5(R_1) = \emptyset.$$

At vertex v_3, robot R_1 updates the vertex incidence matrix with the complete map. Therefore,

$$\mathcal{E}_u^5(v_3) \cup \mathcal{E}_o^5(v_3) = \emptyset.$$

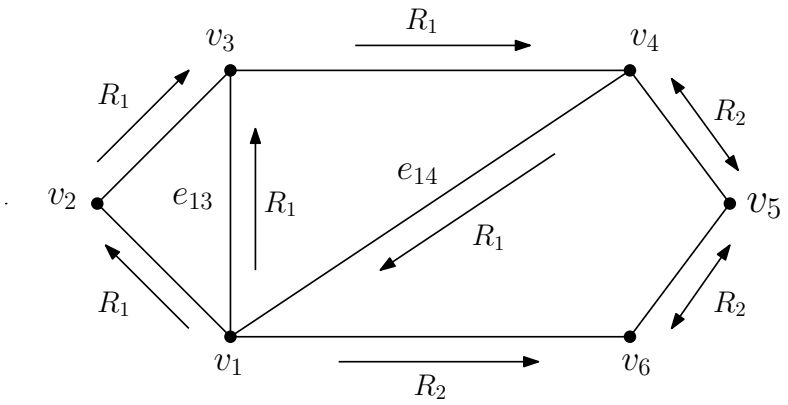


Figure 4.2: Robot edge traversals

The proposed algorithm systematically explores a given environment and minimizes redundancy using the edge selection scheme. For a graph $G(\mathcal{V}, \mathcal{E})$ with finite $|\mathcal{V}|$ and $|\mathcal{E}|$, the robots completely explore the graph in finite edge traversals. As the environment to be explored is of finite size/dimensions, the edge lengths of the graph associated with the environment are also finite. Hence, any robot takes finite time to traverse any edge in the graph. Therefore, the terms finite time and finite steps are used interchangeably to prove the exploration completion. Theorem 4.7 describes the exploration completion in finite edge traversals.

Theorem 4.7 *The exploration is complete in finite number of edge traversals (updates), if next edge traversal e_{ij} for a robot R_k is selected from* (4.4) *at every update instance n.*

Proof From (4.4), we have $\{e_{ij} \mid e_{ij} \in \mathcal{E}_m^n(R_k) \cup \mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k)\}$. From (4.6), $\mathcal{E}_u^n(R_k) = \emptyset \implies \mathcal{E}_m^n(R_k) = \emptyset$.

We claim that $\mathcal{E}_c^n(R_k)$ increments by not more than $|\mathcal{V}_c^n(R_k)|$ edge traversals. Let a robot R_k reaches a vertex v_{curr} at n^{th} update instance and selects an edge e_{ij} for next edge traversal such that $e_{ij} \in \mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k)$ ($e_{ij} \notin \mathcal{E}_c^n(R_k)$). Since $G_c^n(R_k)$ is connected, there exists a path from v_{curr} to v_i passing through not more than $|\mathcal{V}_c^n(R_k)|$ vertices (including v_{curr} and v_i). Fig. 4.3 depicts an illustration of connected vertices. Let the robot R_k reaches a vertex v_m along the path connecting v_{curr} and v_i at q^{th} update instance, where $q > n$. If $e_{ij} \in \mathcal{E}_c^q(v_m)$, the

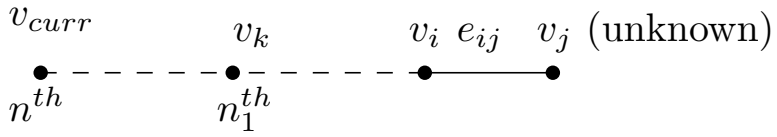


Figure 4.3: Illustration of edge selection and approaching the connected vertex

subgraph $G_c^n(R_k)$ is obtained using (4.10). and $e_{ij} \in \{\mathcal{E}_c^{n+n_1}(R_k) \mid n_1 \leq |\mathcal{V}_c^n(R_k)| - 1\}$. Otherwise, the edge e_{ij} is traversed in atmost $|\mathcal{V}_c^n(R_k)|$ edge traversals.

Since $\mathcal{E}_u^n(R_k) \cup \mathcal{E}_o^n(R_k) \neq \emptyset$ until exploration is complete, the proposed strategy selects an edge e_{ij} for next traversal till exploration is complete. Each edge selection ($e_{ij} \notin \mathcal{E}_c^n(R_k)$) results in $e_{ij} \in \mathcal{E}_c^{n+n_1}(R_k)$ with $n_1 \leq |\mathcal{V}_c^n(R_k)|$. From (4.12), we have

$$|\mathcal{E}_u^n(R_k)| + |\mathcal{E}_o^n(R_k)| \leq E - |\mathcal{E}_c^n(R_k)|\,.$$

We also have $|\mathcal{E}_u^n(R_k)| + |\mathcal{E}_o^n(R_k)| = 0$ when $|\mathcal{E}_c^n(R_k)| = E$ from Theorem 4.4. Since $|\mathcal{E}_c^n(R_k)|$ increments by one in finite number of edge traversals, it reaches to the value E in finite number of edge traversals. ■

For any finite graph, the exploration algorithm guarantees a finite time completion. Also the strategy ensures that none of the robots fall in interminable cyclic traversals.

Corollary 4.8 *A robot is never captured in unending cyclic traversal, if selection of next edge traversal is such that* $e_{ij} \notin \mathcal{E}_c^n(R_k)$ *for every* n.

Proof Let $C(\mathcal{V}_{cy}, \mathcal{E}_{cy}) = \{v_1, e_{12}, v_2, \ldots, v_x, e_{x1}, v_1\}$ be a cycle in graph G. Let the vertex and edge sets of the cycle be $\mathcal{V}_{cy} = \{v_1, v_2, \ldots, v_x\}$ and $\mathcal{E}_{cy} = \{e_{12}, e_{23}, \ldots, e_{x1}\}$. Let us suppose a robot R_k start to traverse the cycle through the edge e_{12} and completed the cycle by reaching v_1 through the back edge e_{x1}. At every vertex, robots update the incidence matrix and choose an

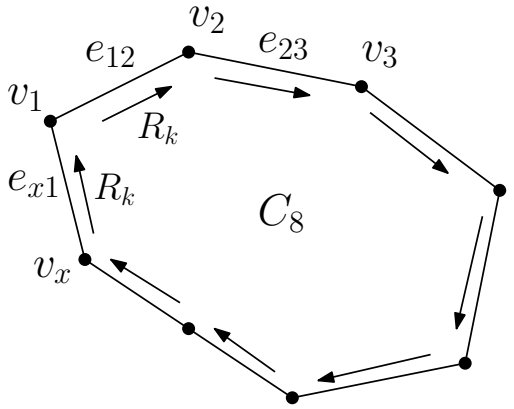


Figure 4.4: Robot cyclic traversal

edge based on (4.4).

After $(k-1)^{th}$ traversal, the status of all the edges of the cycle C are updated to completed edges. i.e. $\mathcal{E}_{cy} \in \mathcal{E}_c^n(R_k)$. The strategy allows the robot to re-traverse the edges of the cycle C only if there exists at least one vertex $v_y \in \{v_2, v_3, \ldots, v_x\}$ with at least one unexplored or out edge. So, the next edge chosen by the robots $e_{ij} \notin \mathcal{E}_c^n(R_k)$ for every n. Thus no robot traverses a cycle in the graph unendingly. ■

Lemma 4.9 *An edge may be traversed by multiple robots more than once in either direction.*

Proof Let R_1 and R_2 be the two robots exploring a graph as shown in Fig. 4.5. Let $t(e_{ij})$ represents the number of times an edge e_{ij} is traversed by the robots. The completed and unexplored (graph portion G_1) edges of the graph are shown using solid and dotted lines respectively.

Let the robot R_1 reaches to the vertex v_2 by traversing e_{12} as shown in Fig. 4.5. So, $t(e_{12}) = 1$. At vertex v_2, R_1 selects the edge e_{16} from $I^p(v_1)$ (from my-unexplored edges list)

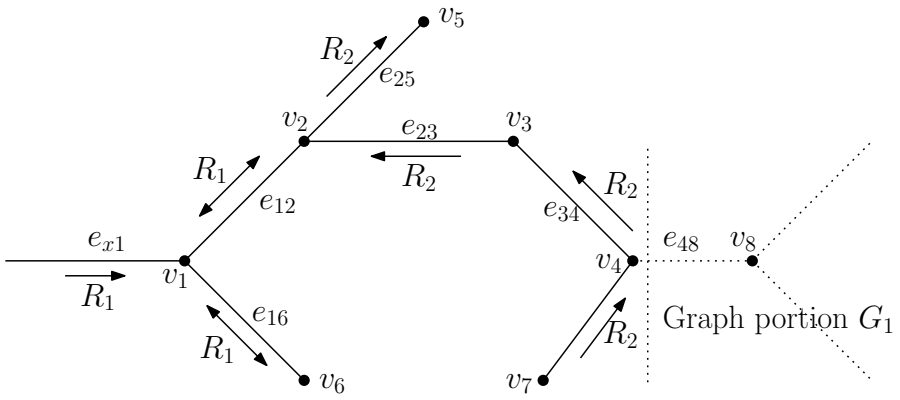


Figure 4.5: Repetitive edge traversal

and traverses back to vertex v_1 through e_{12}. Now that the edge e_{12} has been traversed by R_1 in both the directions, thus $t(e_{12}) = 2$. After completing the edge e_{16}, R_1 selects the edge e_{48} from $I^q(v_1)$. So, R_1 moves through the edge e_{12} to further explore the graph portions. Therefore,

$$t(e_{12}) = 3.$$

Let $\mathcal{G}_1$ be the portion of the graph containing the unexplored edges as shown in the Fig. 4.5. So, traversing to the nearest neighbor containing at least one unexplored edge, may require the robots to move back and forth between the graph portion $\mathcal{G}_1$ and the explored region of the graph. So, depending on the topology of the graph robots may traverse any edge more than twice in either direction. ■

Lemma 4.10 *A partially explored graph with at most $|\mathcal{V}| - 1$ vertices will have the incidence matrix $I^n(v_i)$ containing at least one column vector representing unexplored or out edge.*

Proof Let $G_c^n(R_k)$ be a partially explored graph obtained by a robot R_k. Since $|\mathcal{V}_c^n(R_k)| < |\mathcal{V}|$, there exist at least one vertex v_x in G which is not present in G_c^n. So, there exist at least one edge e_{yx} placed rightmost in the incidence matrix $I^n(v_i)$ that connects the vertex v_x to the partially explored graph G_c^n. This edge can either be an e_u or e_o. Therefore,

$$|\mathcal{E}_u + \mathcal{E}_o| \neq 0.$$

■

Example 4.11 *Fig. 4.6 depicts a scenario where two robots R_1 and R_2 explore a graph G starting from vertex v_1. After 3 edge traversals, robots R_1 and R_2 reach the vertices v_4 and v_3 with partial maps $G_c^3(R_1)$ and $G_c^3(R_2)$ respectively. From the exploration scenario shown in*

Fig. 4.6, it is evident that the vertex v_6 in G is not present in either of the partial maps generated by R_1 and R_2. The incidence matrices $I^3(v_3)$ and $I^3(v_4)$ associated with the vertices v_3 and v_4 consist of one unexplored edge e_{16} (present at vertex v_1). Therefore,

$$|\mathcal{E}_u(v_3) + \mathcal{E}_o(v_3)| \neq 0,$$

and

$$|\mathcal{E}_u(v_4) + \mathcal{E}_o(v_4)| \neq 0.$$

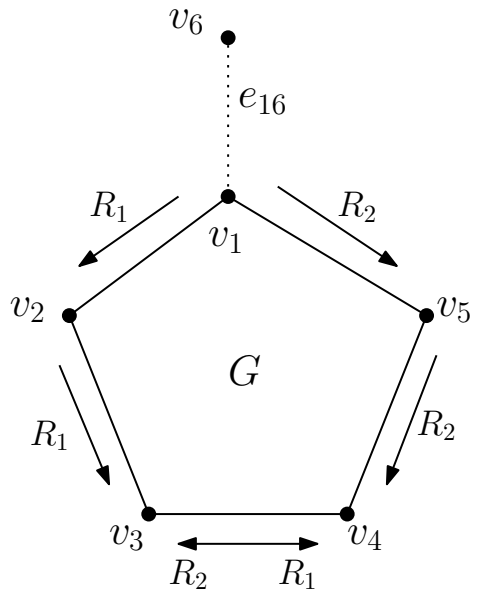


Figure 4.6: Partially explored graph

The next section 4.3 presents proposed strategy based on the modified incidence matrix data structure.

4.3 Algorithm using Modified Incidence Matrix

The modified incidence matrix described in Section 4.1 renders a data structure that encodes the information associated with sets $\mathcal{V}_c^n(.), \mathcal{E}_c^n(.)$, $\mathcal{E}_o^n(.), \mathcal{E}_u^n(.)$ and $\mathcal{E}_m^n(.)$ in a matrix form. An incidence matrix of size $\hat{V} \times \hat{E}$, where $\hat{V} = 0, 1, \ldots, V$ and $\hat{E} = 0, 1, \ldots, E$, represents edges as column vectors (identification of an edge as completed, out or unexplored is described in section 4). The proposed implementation suggests maintaining an ordered column vectors in the incidence matrix. For example, column vectors are ordered from left to right corresponding to edges in following sequence: completed, out, unexplored. Various operations on the proposed data structure are as follows:

- *Initialize :* Let a vertex v_j be connected with E_j number of edges. Initializations of incidence matrices for a robot and vertex are given by (4.14).

$$\begin{aligned} I^0(R_k) &= \begin{bmatrix} 0 \end{bmatrix} \ \forall\ k = 1, 2, \ldots, K, \\ I^0(v_j) &= \begin{bmatrix} \theta_{j1} & \ldots & \theta_{jE_j} \end{bmatrix}. \end{aligned} \tag{4.14}$$

  The dimensions of incidence matrices for a robot and vertex at initialization are 0×0 and $1 \times E_j$ respectively.

- *Merge :* Two ordered incidence matrices I_1 and I_2 of dimensions $V_1 \times E_1$ and $V_2 \times E_2$ respectively are merged by eliminating copies of same vertex and edge. Each row of an incidence matrix is associated with a unique vertex tag. The function *vertex_tag* returns the tag associated with a row and *non-zero_element_count* function returns the number of non-zero elements in a column. Pseudo-code for merging two incidence matrices is described in Algorithm 1. An element (i, j), i^{th} row and j^{th} column of matrix I are referred as $I[i, j]$, $I[i, :]$ and $I[:, j]$ respectively. A submatrix from i_1 to i_2 row and j_1 to j_2 of matrix I is referred as $I[i_1 : i_2, j_1 : j_2]$. Extraction of elements from single row and column extraction of matrix I are referred by $I[i_1, j_1 : j_2]$ and $I[i_1 : i_2, j_1]$ respectively. Merging operation preserves ordering of individual matrices, but removes copies of common edges and vertices.

- *Order :* The merged incidence matrix is reordered in this operation. The columns of the two incidence matrices to be merged are individually ordered based on the type of an edge i.e. completed, out and unexplored edges. The completed edges are identified by finding the columns corresponding to two number of non-zero elements. If the completed edges are eliminated then there is only one non-zero element in the columns. The out edge is identified by finding the column with negative sign of non-zero element in the column. The remaining columns after removing columns corresponding to completed and out edges in matrix I are unexplored edges. The functions *completed*, *out* and *unexplored* extract columns corresponding to completed, out and unexplored edges of matrix I respectively. Same type of edges from two portions corresponding to disjoint matrices I_1 and I_2 are clubbed together in this operation. The function *concatenate* adds the columns in sequence. Pseudo-code for ordering the merged incidence matrix is described in Algorithm 2.

Algorithm 1 *Merge_Matrices*

Input: I_1 of dimension $V_1 \times E_1$ and I_2 of dimension $V_2 \times E_2$

Output: Merged incidence matrix I of dimension $\hat{V} \times (\hat{E}_1 + \hat{E}_2)$

Functions: *vertex_tag*(), *delete_row*(), *delete_column*(), *non-zero_element_count*()

$I = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}$; $\hat{V} = V_1 + V_2$; $\hat{E}_1 = E_1$; $\hat{E}_2 = E_2$; $del_j = []$; $del_i = []$.

for each pair (i_1, j_1), where integer $i_1 \in [1, V_1]$ and integer $j_1 \in [1, E_1]$ **do**

for each pair (i_2, j_2), where integer $i_2 \in [V_1{+}1, V_1{+}V_2]$ and integer $j_2 \in [E_1{+}1, E_1{+}E_2]$ **do**

if *vertex_tag*($I[i_1, :]$)=*vertex_tag*($I[i_2, :]$) (common vertex) & $|I[i_1, j_1]| = |I[i_2, j_2]|$ (common edge) **then**

if status of edge information is different ($\text{sgn}(I[i_1, j_1]) \neq \text{sgn}(I[i_2, j_2])$) **then**

$I[i_1, j_1] = I[i_2, j_2] = -|I[i_1, j_1]|$.

if column j_2 corresponds to a completed edge (*non-zero_element_count*($I[V_1{+}1 : V_1 + V_2, j_2]) = 2$) **then**

Record j_1 in list del_j (for deleting j_1 column later).

$\hat{E}_1 = \hat{E}_1 - 1$.

else if column j_1 corresponds to a completed edge(*non-zero_element_count*($I[1 : V_1, j_1]) = 2$) **then**

Record j_2 in list del_j (for deleting j_2 column later).

$\hat{E}_2 = \hat{E}_2 - 1$.

else

Record j_1 in list del_j (for deleting any one column later).

$\hat{E}_1 = \hat{E}_1 - 1$.

Copy edge details of j_2 at column j_1 preserving the details of other vertex, if it does not correspond to a complete edge (

if (*non-zero_element_count*($I[V_1 + 1 : V_1 + V_2, j_1] < 2$) **then**

$I[V_1 + 1 : V_1 + V_2, j_1] = I[V_1 + 1 : V_1 + V_2, j_1] + I[V_1 + 1 : V_1 + V_2, j_2])$.

Copy edge details of j_1 at column j_2 preserving the details of other vertex, if it does not correspond to a complete edge (

if (*non-zero_element_count*($I[1 : V_1, j_2] < 2$) **then**

$I[1 : V_1, j_2] = I[1 : V_1, j_2] + I[1 : V_1, j_1].)$

Record i_1 in list del_i (for deleting i_1 row later).

Delete rows recorded in list del_i and columns recorded in list del_j.

Algorithm 2 *Order_Matrix*

Input: Merged incidence matrix I of dimension $\hat{V} \times (\hat{E}_1 + \hat{E}_2)$

Output: Ordered incidence matrix $I_{ordered}$, Number of completed edges E_c in $I_{ordered}$.

Functions: *concatenate*(), *completed*(). *out*(), *unexplored*(), *size*()

$I_1 = I[:, 1 : \hat{E}_1], I_2 = I[:, \hat{E}_1 + 1 : \hat{E}_1 + \hat{E}_2]$.

$I_{ordered}$= *concatenate*(*completed*(I_1), *completed*(I_2), *out*(I_1), *out*(I_2), *unexplored*(I_1), *unexplored*(I_2)).

E_c=*size*(*completed*(I_1))+*size*(*completed*(I_2)).

When the exploration begins, each robot *initializes* its incidence matrix using (4.14). When a robot reaches a new vertex v_j, it drops a beacon with initialized incidence matrix for vertex v_j. Let the robot R_k visits a vertex v_j by traversing edge e_{ij} at the n^{th} update instance. The incidence matrices of the robot R_k and vertex v_j are updated using Algorithm 3.

Algorithm 3 *First_Step_on_Vertex_Visit*

Input: $I^{n-1}(v_j)$, $I^{n-1}(R_k)$ and I', the column vector corresponding to a completed edge

Output: Updated incidence matrices $I^n(R_k)$ and $I^n(v_j)$

Functions: *Merge_Matrices*(), *Order_Matrix*()

Assign $I^{n-1}(v_j) = I^0(v_j)$.

I'' =*Merge_Matrices*($I', I^{n-1}(R_k)$)[2].

I = *Merge_Matrices*($I'', I^{n-1}(v_j)$).

$I^n(R_k) = I^n(v_j)$ = *Order_Matrix*(I).

The robot R_k decides the next edge traversal using Algorithm 4. When the robot R_k selects an edge from the rightmost column of its updated incidence matrix $I^n(R_k)$, the selected edge may not be an edge of the current vertex. Therefore, shortest path to the known vertex is identified and corresponding edges are entered in a queue. The robot merges the latest incidence matrix at each vertex while traversing already explored edges to reach the selected edge, but does not change decision before it finds that the selected edge is a completed edge. Next we present the flowchart of the proposed IMRDFS strategy.

[2]The column vector I' corresponding to a completed edge is added at the left side of the matrix in Step 2, therefore ordering operation is not required.

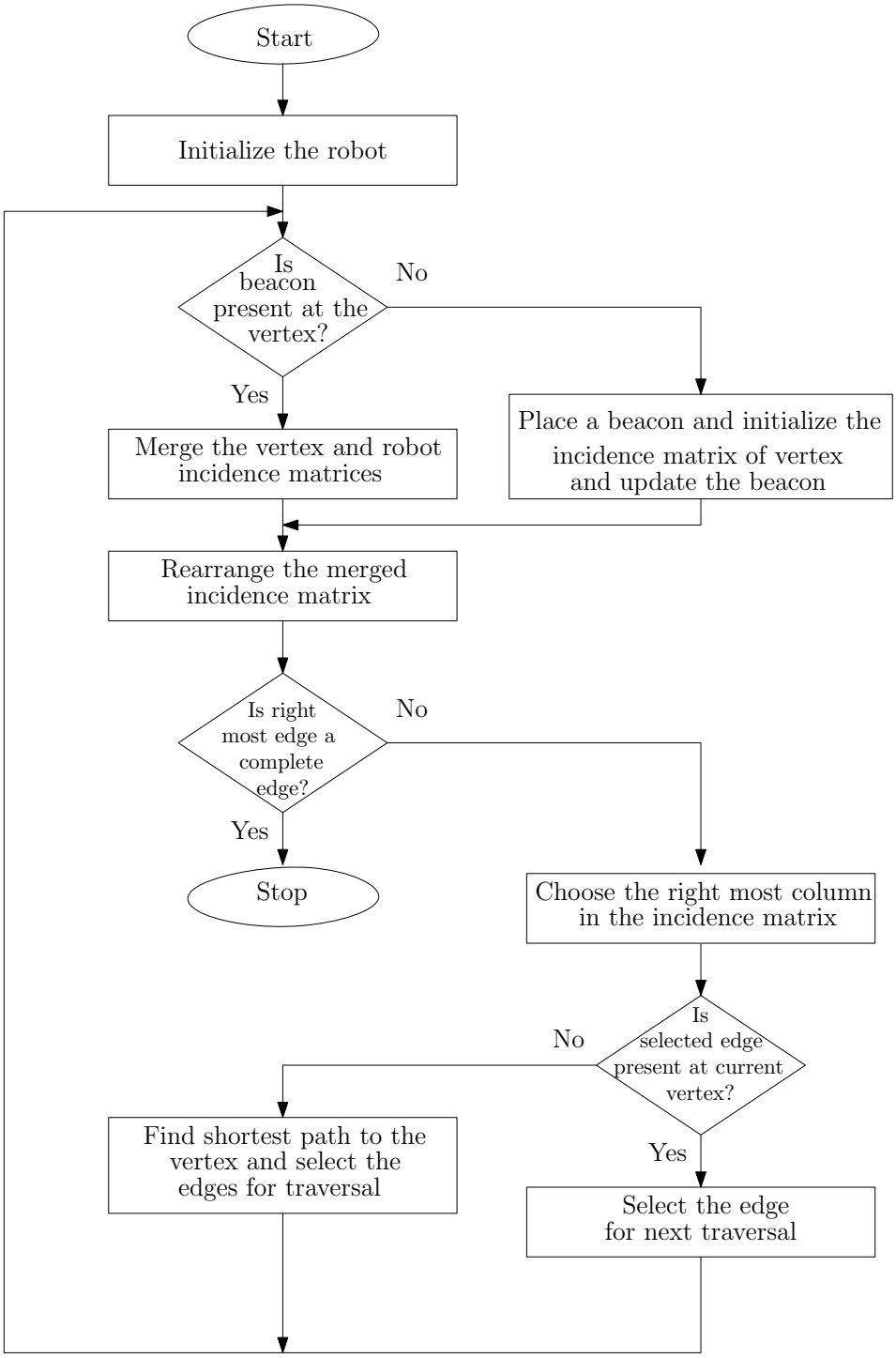


Figure 4.7: Flowchart of IMRDFS Strategy

Algorithm 4 *Second_Step_on_Vertex_Visit*

Input: $I^n(R_k)$ of dimension $\hat{V} \times (\hat{E}_1 + \hat{E}_2)$

Output: next edge for traversal

Functions: *non-zero_element_count*(), *non-zero_element_sign*()

Initialize a queue Q as empty.

if *non-zero_element_count*$(I^n(R_k)[:, \hat{E}_1 + \hat{E}_2]) = 2$ **then**

Stop.

else if (queue Q is empty or last entry (selected edge) of Q is a completed edge in $I^n(R_k)$) **then**

Select the edge corresponding to $I^n(R_k)[:, \hat{E}_1 + \hat{E}_2]$ (rightmost column).

Find shortest path from $I^n(R_k)$ to reach to the known vertex of selected edge.

Add edges from the shortest path following the traversal order in the queue Q.

Add selected edge in the queue Q.

Assign next edge for traversal as the first entry of Q.

if *non-zero_element_sign*$(I^n(R_k)[:, \hat{E}_1 + \hat{E}_2])$ = *+ve* **then**

$I^n(R_k)[i, \hat{E}_1 + \hat{E}_2] = -I^n(R_k)[i, \hat{E}_1 + \hat{E}_2] \; \forall \; i = 1, \ldots, \hat{V}$.

Circularly shift-right $I^n(R_k)[:, E_c + 1 : \hat{E}_1 + \hat{E}_2]$ by one column.

$I^n(v_j) = I^n(R_k)$.

Remark 4.12 The set of my-unexplored edges identified by a robot R_k ($\mathcal{E}_m(R_k)$) are the edges of a vertex v_j ($I^0(v_j)$), when the robot R_k visits vertex v_j. The column vectors corresponding to these are added at the right side of merged incidence matrix in the first step of Algorithm 3. Therefore, it gets highest priority while selecting an edge in the first step of Algorithm 4. *In Fig. 4.6, the edges e_{23} and e_{45} are the my-unexplored edges identified by the robots R_1 and R_2 at vertex v_1 after one edge traversal. These edges are added to the right side of the corresponding merged incidence matrices and are given higher priority while selecting the next edge.*

Remark 4.13 The incidence matrix is ordered, therefore termination condition needs to be checked with just the rightmost column vector. From Algorithm 2, always the unexplored and out edges are placed on the rightmost columns of $I^n(R_k)$. If the rightmost column is an element of $\mathcal{E}_c$, then it sure that there are no unexplored and out edges left in the graph. Henceforth, checking the rightmost column vector is sufficient enough to identify exploration completion.

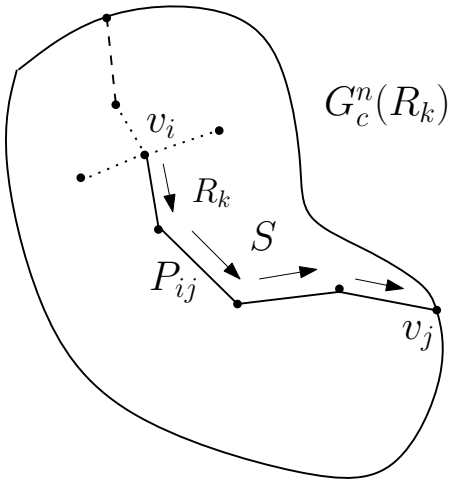


Figure 4.8: Shortest path to a vertex in partial graph

Remark 4.14 The shortest path identified from $I^n(R_k)$ to reach the known vertex of selected path is explored, because $G_c^n(R_k)$ is connected (from Lemma 4.2).

In Fig.4.8, the completed, unexplored and unidentified edges are shown using solid, dotted and dashed lines respectively. Let S be a spanning tree of $G_c^n(R_k)$ and P_{ij} be the shortest path from a vertex v_i to another vertex v_j, $\forall\ v_i, v_j \in \mathcal{V}_c^n(R_k)$. Since, $\mathcal{V}(P_{ij}) \subseteq \mathcal{V}(S)$ and $\mathcal{E}(P_{ij}) \subseteq \mathcal{E}(S)$, the path P_{ij} is a sub graph of S. Also $\mathcal{E}(S) \subseteq \mathcal{E}_c^n(R_k)$, therefore the shortest path to reach any vertex containing the rightmost vector in $I^n(R_k)$ is explored.

Example 4.15 *The incidence matrix initialize, merge and order operations have been illustrated through this example. The paths traversed by the robots R_1 and R_2 are shown using dashed and dash-dotted lines respectively. The robots R_1 and R_2 start the exploration from the vertex v_1 and reach the vertex v_3 after two edge traversals. The operations for the robot traversals shown in Fig. 4.9 are:*

1. Initialize: *The incidence matrices for the vertices and the robots are initialized as,*

$$\begin{aligned}
I^0(v_1) &= \begin{bmatrix}\theta_{12}\ \theta_{14}\end{bmatrix}, \\
I^0(v_2) &= \begin{bmatrix}\theta_{21}\ \theta_{23}\ \theta_{25}\end{bmatrix}, \\
I^0(v_3) &= \begin{bmatrix}\theta_{32}\ \theta_{34}\ \theta_{36}\end{bmatrix}, \\
I^0(v_4) &= \begin{bmatrix}\theta_{41}\ \theta_{43}\end{bmatrix}, \\
I^0(v_5) &= \begin{bmatrix}\theta_{52}\end{bmatrix}, \\
I^0(v_6) &= \begin{bmatrix}\theta_{63}\end{bmatrix}, \\
I^0(R_1) &= I^0(R_2) = \begin{bmatrix}0\end{bmatrix}.
\end{aligned}$$

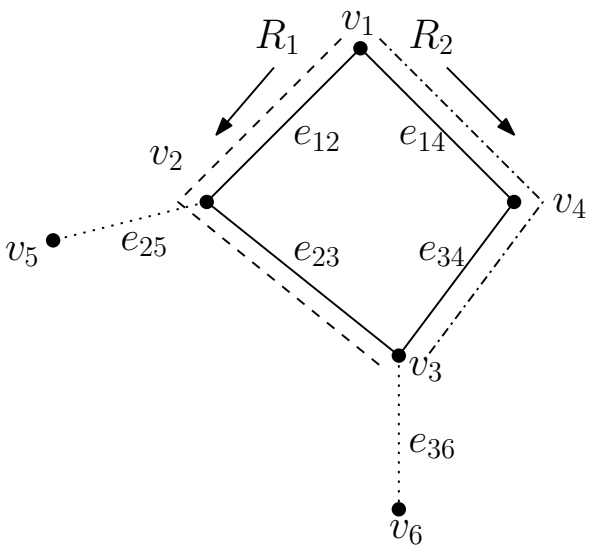


Figure 4.9: Example graph traversal with two robots

2. Merge: *The two ordered incidence matrices associated with the robots R_1 and R_2 of dimensions 3×6 and 3×5 respectively (upon arriving at the vertex v_3) are given by,*

$$I^2(R_1) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 \\ -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} \end{bmatrix} \end{array},$$

$$I^2(R_2) = \begin{array}{c} \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccc} e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \\ 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \end{bmatrix} \end{array}.$$

The vertex tags of incidences matrices $I^2(R_1)$ and $I^2(R_2)$ are,

$$\begin{aligned} vertex_tag(I^2(R_1)) &= [v_1,\ v_2,\ v_3], \\ vertex_tag(I^2(R_2)) &= [v_1,\ v_4,\ v_3]. \end{aligned}$$

The merge operation is initialized by constructing the merged matrix I as,

$$I = \begin{bmatrix} I^2(R_1) & 0 \\ 0 & I^2(R_2) \end{bmatrix}.$$

Therefore,

$$I = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \end{bmatrix} \end{array},$$

Using the $vertex_tag$ information, the common vertices in $I^2(R_1)$ and $I^2(R_2)$ are identified and the elements in the corresponding rows are checked for duplicate entries. If the entries in the rows with common vertex_tag are the same, then the corresponding columns are copied into each other. Once the values are copied, one copy of the duplicate columns and rows are identified and removed.

In row 1 of the incidence matrix I, the columns 1 & 9 corresponding to edge e_{12} and columns 3 & 7 corresponding to edge e_{14} are the duplicate entries of the same edge. So, the entries of columns 1 & 9, 3 & 7 are copied into each other and columns 9, 3 & row 1 are recorded for removal. The incidence matrix resulted from these operations is,

$$I = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & -\theta_{21} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} & 0 & 0 & 0 & 0 & 0 \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & 0 & -\theta_{41} & 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \end{bmatrix} \end{array},$$

Similarly, in row 3 of I, the entries in columns 2 & 10 corresponding to edge e_{23}, columns 5 & 8 corresponding to edge e_{34} and columns 6 & 11 corresponding to edge e_{36} are the duplicate entries of the same edge. These column entries are copied into each other and columns 10, 5, 6 & rows 3 are recorded for removal. The resulting incidence matrix after

these set of operations is,

$$I = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & -\theta_{21} & -\theta_{23} & 0 \\ 0 & -\theta_{32} & 0 & 0 & -\theta_{34} & \theta_{36} & 0 & 0 & 0 & -\theta_{32} & \theta_{36} \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & 0 & -\theta_{41} & 0 & -\theta_{43} & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & -\theta_{34} & \theta_{36} & 0 & -\theta_{34} & 0 & -\theta_{32} & \theta_{36} \end{bmatrix} \end{array},$$

After removing the recorded rows (1, 3) and columns (9, 3, 10, 5, 6) from I, the final merged incidence matrix (un-ordered) is given by,

$$I = \begin{array}{c} \\ v_2 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{23} & e_{25} & e_{14} & e_{34} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & \theta_{25} & 0 & 0 & 0 \\ -\theta_{12} & 0 & 0 & -\theta_{14} & 0 & 0 \\ 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 \\ 0 & -\theta_{32} & 0 & 0 & -\theta_{34} & \theta_{36} \end{bmatrix} \end{array}.$$

3. Order: *The ordered incidence matrix $I_{ordered}$ obtained by reordering the merged incidence matrix I is given by,*

$$I_{ordered} = \begin{array}{c} \\ v_2 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{23} & e_{14} & e_{34} & e_{25} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & 0 & 0 & \theta_{25} & 0 \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 \\ 0 & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 \\ 0 & -\theta_{32} & 0 & -\theta_{34} & 0 & \theta_{36} \end{bmatrix} \end{array}.$$

We next show that the incidence matrix based implementation is also applicable to graph with cycles.

Proposition 4.1 *If $\mathcal{E}_u^n(.) \cup \mathcal{E}_o^n(.) = \emptyset$ for any robot or vertex, then dimension of its incidence matrix is $V \times E$.*

Proof From *Case 2* of Theorem 4.4, we get $\mathcal{E}_c^n(.) = \mathcal{E}$, if $\mathcal{E}_u^n(.) \cup \mathcal{E}_o^n(.) = \emptyset$, or

$$|\mathcal{E}_c^n(.)| = E. \tag{4.15}$$

Let dimension of incidence matrix be $\hat{V} \times \hat{E}$ for a robot or vertex. We have

$$\hat{E} = |\mathcal{E}_c^n(.)| + |\mathcal{E}_o^n(.)| + |\mathcal{E}_u^n(.)|. \tag{4.16}$$

From (4.15), we get

$$\hat{E} = E.$$

Since $e_{ij} \in \mathcal{E}_c^n(.) \; \forall \; i, j$, both v_i and v_j are explored $\forall \; i, j$. Therefore, all vertices are explored, or

$$\hat{V} = V.$$

The converse claim suggests that the dimension of the incidence matrix for a robot or vertex is $V \times E$. From (4.16), we have

$$E = |\mathcal{E}_c^n(.)| + |\mathcal{E}_o^n(.)| + |\mathcal{E}_u^n(.)|.$$

We show that there exists an edge $e_{ij} \in \mathcal{E}_u^n(.) \cup \mathcal{E}_o^n(.)$ for explored vertices v_i and v_j for a graph with cycle. Since the number of rows in incidence matrix is V, each and every vertex is explored. The explored vertices v_i and v_j and unexplored edge e_{ij} happens only if there exists more than one path between vertices v_i and v_j. This implies existence of cycle in the graph. Therefore, converse claim is true only for a tree or graph without cycle. ■

4.4 Simulation Results

The efficiency of the proposed strategy is shown by comparing the results with that of the MRDFS, LRV and BFS approaches. For trees, the proposed strategy is better than any other exploration strategy. On graphs, it is equally efficient when compared to the other strategies.

To illustrate the advantage of the proposed algorithm over the existing strategies, the strategy is simulated in MATLAB 2014b for various tree and graph topologies and the results are presented. Fig. 4.10 depicts tree topologies with 12 edges used for simulation. In Fig. 4.10, several tree topologies with varying width and depth are considered to demonstrate the performance of the proposed strategy on any tree topology. The proposed technique is simulated for each topology shown in Fig. 4.10 using various number of robots between 2 to 10. Number of edge traversals required for completing the exploration by multiple robots for each tree topology in *worst case* is presented in Fig. 4.11 (exploration time for a particular tree topology using

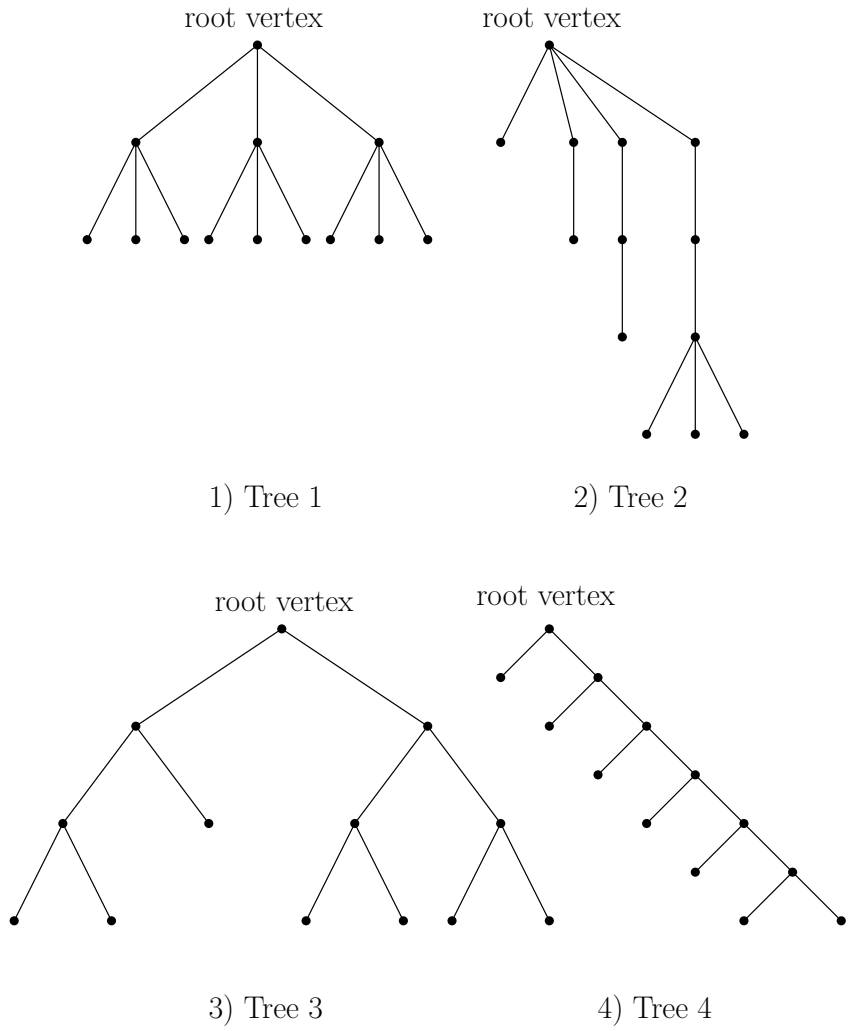


Figure 4.10: Tree topologies with 12 edges

given number of robots depends on the division of number of robots along edges, at a vertex therefore worst case exploration time is reported).

Fig. 4.10 also shows results using MR-DFS technique [1] for comparison. Comparison is further discussed in Section 4.5. Fig. 4.11 shows that the total edge traversals required to complete the exploration depends on the tree topology and the number of robots exploring the tree. The tree topology Tree 4 in Fig. 4.10 takes maximum number of edge traversals to complete exploration with any number of robots.

The proposed multi robot exploration strategy is also simulated for a graph topology as shown in Fig. 4.12. This topology consists of 10 vertices and 12 edges. Each edge is considered being traversed in same time.

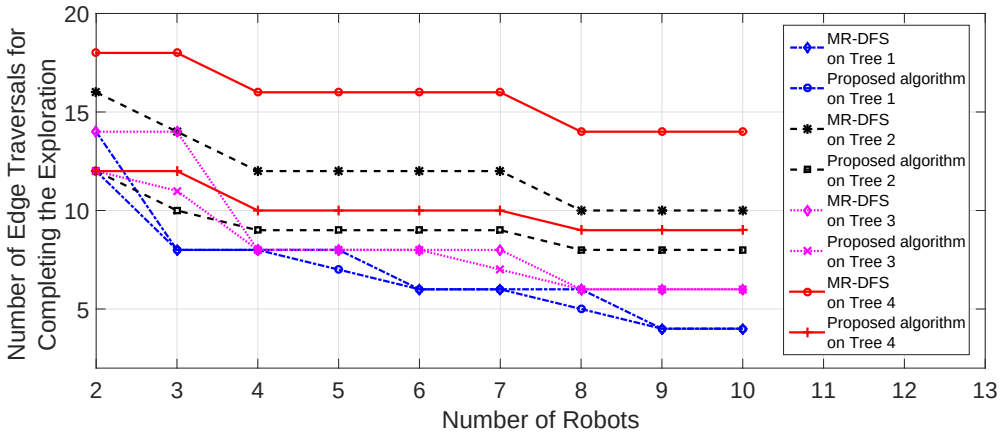


Figure 4.11: Number of edge traversals required to complete exploration Vs. number of robots

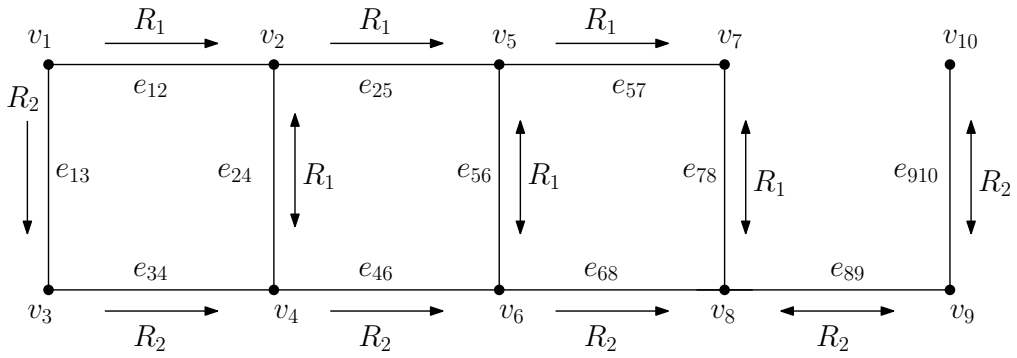


Figure 4.12: Simulated result of a graph exploration using two robots

Fig. 4.12 shows that two robots R_1 and R_2 start the exploration from the root vertex v_1. Sequences of vertices visited by the robots R_1 and R_2 are $v_1-v_2-v_4-v_2-v_5-v_6-v_5-v_7-v_8$ and $v_1-v_3-v_4-v_6-v_8-v_9-v_{10}-v_9-v_8$ respectively. The updated incidence matrix at an intermediate step (10^{th} update at vertex v_6 when robot R_1 visits vertex v_6) is obtained as,

$$I^{10}(v_6) = \begin{array}{c} \\ v_6 \\ v_5 \\ v_4 \\ v_3 \\ v_2 \\ v_1 \end{array} \begin{array}{c} \begin{array}{ccccccccc} e_{56} & e_{46} & e_{34} & e_{13} & e_{25} & e_{24} & e_{12} & e_{68} & e_{57} \end{array} \\ \begin{bmatrix} -\pi/2 & -\pi & 0 & 0 & 0 & 0 & 0 & 2\pi & 0 \\ -3\pi/2 & 0 & 0 & 0 & -\pi & 0 & 0 & 0 & 2\pi \\ 0 & -2\pi & -\pi & 0 & 0 & -\pi/2 & 0 & 0 & 0 \\ 0 & 0 & -2\pi & -\pi/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2\pi & -3\pi/2 & -\pi & 0 & 0 \\ 0 & 0 & 0 & -3\pi/2 & 0 & 0 & -2\pi & 0 & 0 \end{bmatrix} \end{array}$$

The final updated incidence matrix obtained at vertex v_8 has dimension 10×12 and the exploration is completed in eight edge traversals at vertex v_8. The proposed IMRDFS strategy is

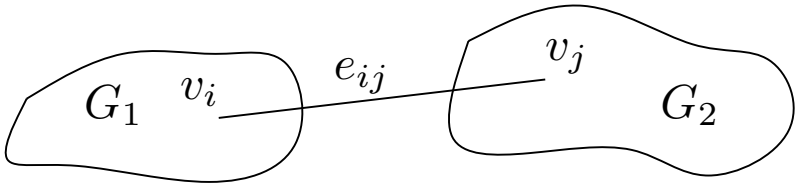


Figure 4.13: Illustration of partitioning of a tree in two subtrees G_1 and G_2

compared with the existing methods and the results are presented in the next section.

4.5 Comparison with Existing Methods

We compare proposed technique with MRDFS [1] and Flooding [13] approaches based on the number of edge traversals required to complete the exploration. The proposed approach reduces the number of edge traversals for completion due to two reasons: one, new condition obtained in Theorem 4.4 is capable of declaring completion without reaching to the root vertex and two, efficient ordering of edges. Efficient ordering of edges gives preference to my-unexplored edges identified by the robot. The robot R_k traverses back through parent vertex (vertices) to reach known vertex of an edge from $\mathcal{E}_m^n(R_k)$. This minimizes extra travel time for each robot to reach to a vertex from which unexplored edge is connected.

The number of edge traversals required to complete a tree exploration for MRDFS [1] and Flooding [13] using two robots is $E + r$, where r is radius of the tree. We show benefit of proposed method using following proposition.

Proposition 4.2 *The number of edge traversals required for completing the exploration of a tree using proposed technique is at most E.*

Proof We assume that time to traverse each edge is same. Let two robots R_1 and R_2 start the exploration of a tree G at root vertex v_r ($K = 2$). Since the robots are exploring a tree (graph without cycle), there exists only one path from vertex v_r to v_i. Let $G = \{G_1, G_2, \{\{v_i, v_j\}, \{e_{ij}\}\}\}$ where $G_1 = \{(\mathcal{V}_1, \mathcal{E}_1) \mid v_i \in \mathcal{V}_1\}$ and $G_2 = \{(\mathcal{V}_2, \mathcal{E}_2) \mid v_j \in \mathcal{V}_2\}$ are subtrees such that $\mathcal{V}_1 \neq \mathcal{V}_2$. We get $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ for a tree. Subtrees G_1 and G_2 are illustrated in Fig. 4.13.

A robot R_1 selects the edge e_{ij} at vertex v_i, if $e_{ij} \in \mathcal{E}_u^n(R_1) \cup \mathcal{E}_o^n(R_1)$. According to the proposed strategy, if $e_{ij} \in \mathcal{E}_o^n(R_1)$, then $\mathcal{E}_u^n(R_1) = \emptyset$ and $G_c^n(R_1) = G_1$. Also, $e_{ij} \in \mathcal{E}_o^n(v_i)$ implies that the robot R_2 has not returned back. Therefore,

$$|\mathcal{E}_1| < |\mathcal{E}_2|\,.$$

If e_{ij} is the last edge explored before completion, then

$$|\mathcal{E}_1| = \begin{cases} |\mathcal{E}_2| + 1, & \text{if } E \text{ is odd, or} \\ |\mathcal{E}_2|, & \text{otherwise.} \end{cases}$$

But, $E = |\mathcal{E}_1| + |\mathcal{E}_2| + 1$. Therefore, exploration is completed in exactly E number of edge traversals using two robots, $K = 2$. In particular, exploration is completed in at most E number of edge traversals for $K \geq 2$.

Single robot DFS needs $2E$ edge traversals for completing the exploration and multi robot DFS using K robots is expected to complete the exploration in $2E/K$ edge traversals. The longest distance traveled by at least one robot for exploring a tree with radius r is $r + 1$ using the proposed method, because the robot needs to return to one level up in the tree to merge the information on partially explored tree. Therefore, the lower bound for number of edge traversals required to complete exploration with the IMRDFS is

$$E_l = \max(2E/K, r + 1) \tag{4.17}$$

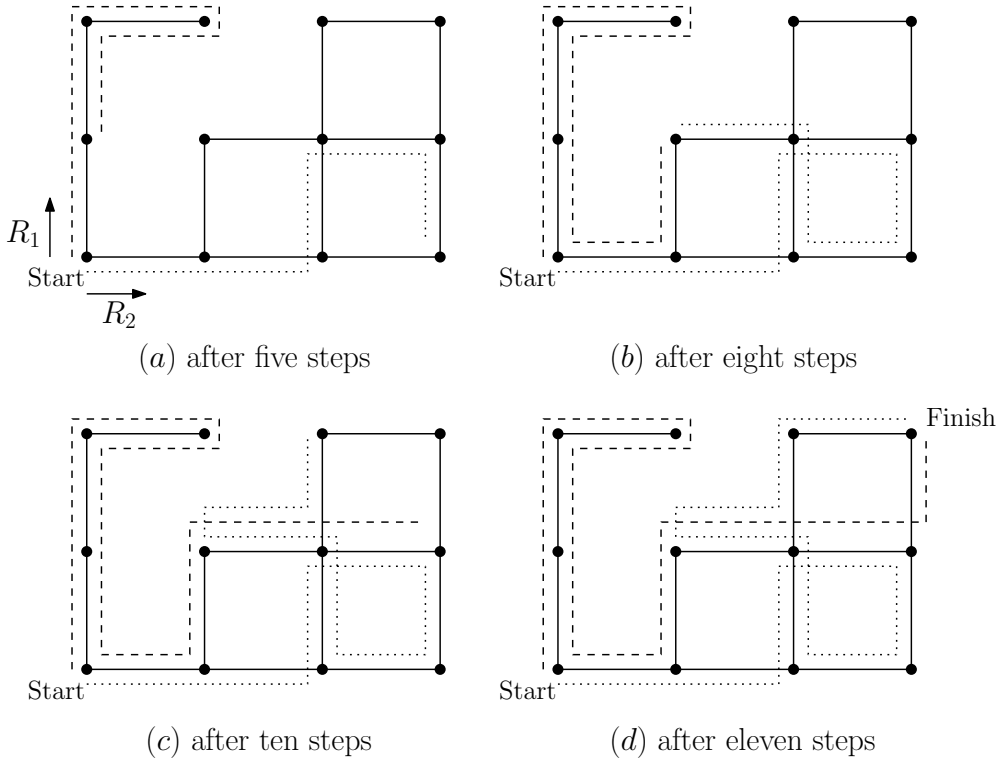
as compared to $\max(2E/K, 2r)$ for MRDFS [1].

We also compare the upper bound on exploration. For a few tree topologies such as balanced tree, the robots return back to the root vertex and declare completion when explored using the proposed MRDFS method. Therefore, the worst case analysis for finding the exploration time in [1] using excess multiplicity of edges is applicable to the proposed MRDFS method as well. Now, upper bound for number of edge traversals required to complete exploration using the IMRDFS strategy (from Proposition 4.2 and analysis on excess multiplicity of edges [1]) is given by

$$E_u = \min\left(E, \frac{2E}{K} + \frac{2}{K}\left(1 + \frac{K}{r}\right)^{K-1} \frac{2}{K!}\, r^{K-1}\right). \tag{4.18}$$

While exploring using sufficient number of robots (large K) for $r > E/2 + 1$, the proposed method completes the exploration in less than $2r$ edge traversals as the robot declares the completion other than the root vertex. Comparison of (4.17) and (4.18) with corresponding lower and upper bounds for MRDFS [1] also show that the proposed IMRDFS strategy gives larger benefit in total edge traversals required to complete the exploration as compared to the existing MRDFS strategy [1] when $r > E/2 + 1$.

The benefit is verified by exploring tree topologies (shown in Fig. 4.11) using various number of robots. Fig. 4.11 conveys that the number of edge traversals required for completing



(a) after five steps (b) after eight steps

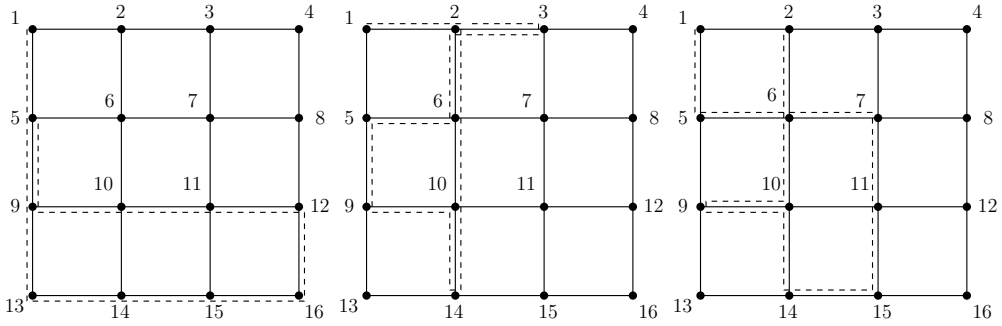
(c) after ten steps (d) after eleven steps

Figure 4.14: Exploration using proposed technique on an example graph

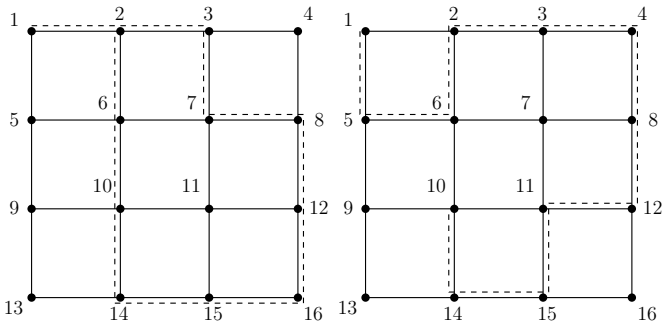
the exploration using proposed technique is always less or equal to that using existing MR-DFS [1]. Efficient ordering of sets and information exchange using proposed method gives benefit on graph exploration as well. The proposed multi robot exploration strategy is also compared with the MR-DFS algorithm for an example graph from [1] with 12 vertices and 14 edges.

Exploration steps using proposed technique for two robots R_1 and R_2 are shown in Fig. 4.14. The paths traversed by the robots R_1 and R_2 are shown using dashed and dotted lines respectively. For both the strategies, the worst case exploration time are compared. The MRDFS method [1] with two robots takes 16 edge traversals to complete the exploration, whereas the proposed method takes 11 edge traversals to complete exploration of same graph.

We also compare results on square lattice graph with 16 vertices and 24 edges with 5 robots by recording the step-wise exploration using proposed and MRDFS methods. These exploration steps are shown in Figs. 4.15 and 4.16. The MRDFS algorithm [1] with 5 robots takes 16 edge traversals to complete the exploration, whereas the proposed strategy takes 11 steps to complete the exploration. Figs. 4.15 and 4.16 show that the partial explored graph information stored in incidence matrices at explored vertices helps in selecting an unexplored edge connected to the current vertex or parent vertices (my-unexplored edge) for next edge

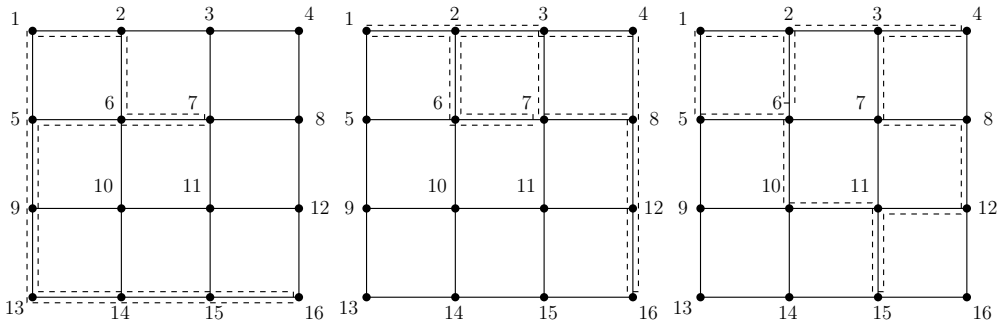


(a) Robot R_1 (b) Robot R_2 (c) Robot R_3

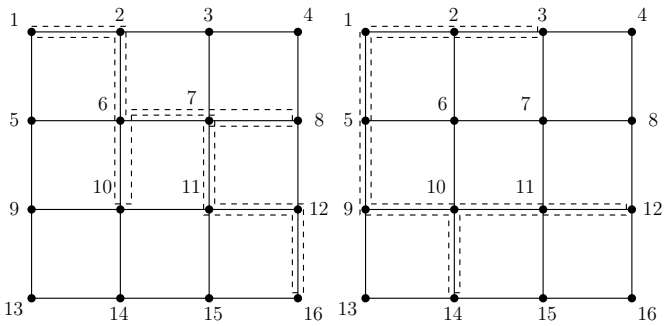


(d) Robot R_4 (e) Robot R_5

Figure 4.15: Exploration using proposed method with 5 robots



(a) Robot R_1 (b) Robot R_2 (c) Robot R_3



(d) Robot R_4 (e) Robot R_5

Figure 4.16: Exploration using the MRDFS [1] with 5 robots

traversal. The proposed approach declares completion at a vertex other than the root vertex, which further reduces the total number of edge traversals.

The proposed method is also compared with BFS [3,4], LRV [5] and MRDFS [1] methods on a square lattice graph with 100 vertices and 180 edges. The simulation results shown in Fig. 4.17 present the total edge traversals required to complete the exploration for various values of K. Fig. 4.17 shows the benefit obtained in total edge traversals using IMRDFS.

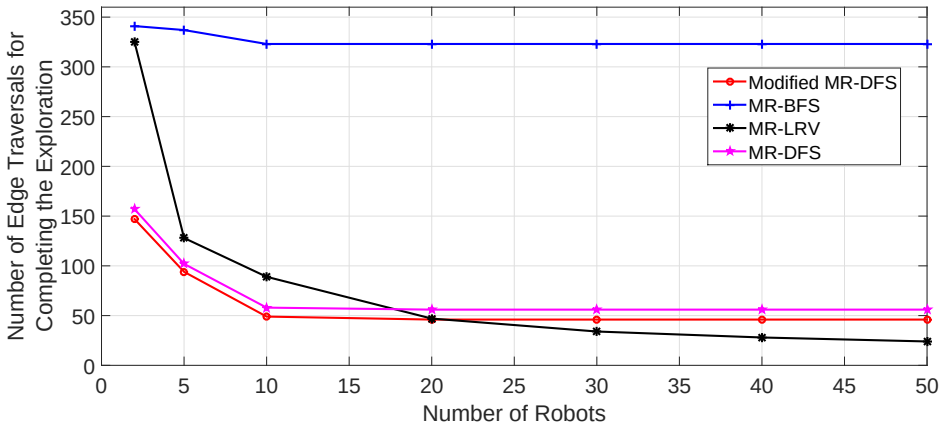


Figure 4.17: Number of edge traversals required to complete exploration Vs. number of robots for a square lattice with 100 vertices and 180 edges

4.6 Conclusion

The finite time completion of exploration for a bounded environment explored by multiple robots using IMRDFS strategy is shown. The proposed modification in incidence matrix data structure facilitates a systematic and coordinated exploration avoiding redundancy in exploration. Each robot updates the beacon at the visited vertex with its partial map and these updates provide coordination among the robots. The proposed strategy takes atmost E edge traversals to complete the exploration on trees, and guarantees finite time completion on graphs. The simulation results indicate the efficiency of the proposed approach over the existing techniques.

The multi robot exploration and map building strategy typically depend on communication scheme, search strategy and the initial position of robots. However, a generic framework for exploration and mapping can be used to develop any existing/new strategies. The next chapter proposes a unified framework for multi robot exploration and mapping with an iterative approach to obtain the optimal fleet size.

Chapter 5

A Generic Framework for Multi Robot Exploration and Mapping

Multi robot mapping and exploration has been widely studied in the field of mobile robotics. Several techniques are present in the literature to address the multi robot coordinated exploration and mapping. Various search strategies, such as DFS [1, 51], BFS [3, 4], LRV [5], and various information exchange methods such as placing book keeping devices at important places [1, 12, 13, 51], broadcasting in limited range [16] have been used for multi robot graph exploration. All these strategies are distinct in terms of the robot coordination, edge selection and declaring completion. Therefore, there is a definite need for a generic multi robot exploration framework that works efficiently for any initial robot positions, search strategy and communication scheme which facilitates the design of new techniques and implementation of existing exploration strategies. This work presents a unified framework for multi robot exploration and mapping strategy for bounded environments. Incase of unbounded environments, the exploration is performed using BFS strategy where the agents keep returning to the root vertex. The proposed generic framework is also applicable to the multi agent graph exploration using BFS. Knowing the necessary and sufficient conditions for declaring completion and finite time completion supports development of an efficient strategy and suggest possible modifications in the method of exchanging information.

The key challenge is to coordinate the robots for exploiting the maximum benefit of multiple robots by effective use of the partial map information. Therefore, the modified incidence matrix data structure (introduced in section 3.1) is adhered to propose a unified framework to address each of these strategies. This chapter details the exploration strategies revised using

the modified incidence matrix and the vector space analysis for the multi robot graph exploration. Modified incidence matrix is used for organized exchange of information and accessible selection of edges supporting systematic completion of the exploration. Further, relationship between exploration time and number of robots (team size, fleet size) is investigated for various combinations of search strategy and communication method. A generic approach to find an optimal number of robots and minimum exploration time are developed. Simulation results show that a significant reduction in the completion time-team size product is possible using proposed technique. The proposed generic framework for multi robot exploration is discussed next.

5.1 Proposed Framework for Multi Robot Graph Exploration

This section discusses a complete and scalable multi robot exploration and mapping strategy. A generic multi robot exploration framework that works efficiently with any initial robot positions, search strategy and communication scheme is essential to design new techniques and compare the performance of existing exploration strategies. A unified framework presented in this work is an attempt to facilitate a generic exploration strategy leading to completion with the help of organized information exchange. The description of proposed framework for exploring unknown graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is as follows:

The graph exploration is initiated by multiple robots. Each robot builds its partially explored graph. Various methods can be used to communicate the partially explored graph of each robot to other robots. A few examples of information exchange method are broadcasting, through centralized station, through beacons at various places in the environment and exchange at the meeting place. The proposed framework considers exchanging the partially explored graph between various sources. These sources include the robots, beacons, centralized station, etc. The partially explored graph $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A)$ of a source A also constitutes half edges. Referring to section 3.1, a half edge $e_{j\mathrm{x}}$ incident at vertex v_j is one among the E' number of edges incident at the vertex v_j. In particular, $e_{j\mathrm{x}} \in \{\mathrm{e}_{j_1}, \mathrm{e}_{j_2}, \ldots, \mathrm{e}_{j_{E'}}\}$.

Remark 5.1 The edge set $\mathcal{E}_A$ uniquely describes the partially explored graph $\mathcal{G}_A$, because explored vertex (includes half edges)/vertices of each edge in $\mathcal{E}_A$ define the vertex-set $\mathcal{V}_A$.

Example 5.2 *For the graph exploration shown in Fig. 5.1, the completed, unidentified and half edges are shown using solid, dashed and dotted lines respectively. The edge and vertex sets*

associated with source A are,

$$\mathcal{E}_A = \{\mathrm{e}_1, \mathrm{e}_2, e_{2x}\} \; and \; \mathcal{V}_A = \{v_1, v_2, v_3\}.$$

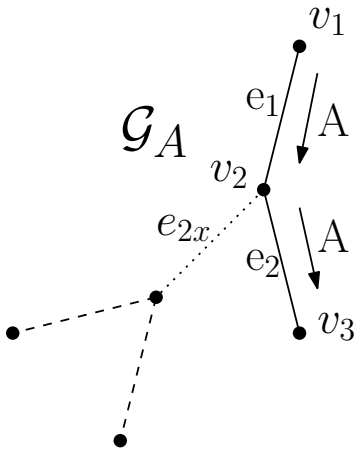


Figure 5.1: Edge and vertex sets of a partially explored graph

Each half edge can also be associated with additional information (representing weight of a half edge), for example, depth for MRDFS [1, 51], breadth for MRBFS [3, 4], number of visited robots for MRLRV [5], shortest path from the current location for flooding algorithm [13] etc. This information (weight) is captured in positive integer p explained as follows: A vector $\mathcal{E}_A \in \mathcal{W}_\mathcal{E}$ corresponding to the partially explored edge set $\mathcal{E}_A$ of a source A is expressed as

$$\mathcal{E}_A = \sum_{i=1}^{E} f_i \mathrm{e}_i \;\text{ or }\; \mathcal{E}_A = (f_1, f_2, \ldots, f_E), \tag{5.1}$$

where each

$$f_i = \begin{cases} 0 & \text{if } \mathrm{e}_i \notin \mathcal{E}_A, \\ 1^{\pm p} & \text{if } \mathrm{e}_i \in \mathcal{E}_A \text{ is an arbitrarily oriented (sign} \\ & \text{describes the orientation) half edge,} \\ 1 & \text{otherwise.} \end{cases} \tag{5.2}$$

Since the scalar value $1^{\pm p}$ provides additional information to that of scalar value 1, the scalar addition (modulo 2 sum) [54] for $1^{\pm p}$ with scalars $0, 1$ and $1^{\pm p'}$, where p' may or may not be equal to p, are same as that with scalar 1. However, the added information is preserved. For example; scalar addition of $1^{\pm p}$ with 0 is same as scalar addition of 1 with 0. In particular, since the scalar addition of 1 and 0 is 1, the scalar addition of $1^{\pm p}$ and 0 is also 1 with the information with variable p preserved. Therefore, the result of scalar addition of $1^{\pm p}$ and 0 is $1^{\pm p}$. Similarly,

the scalar addition of 1^{+p} and $1^{+p'}$ is treated as scalar addition of 1 and 1, which results in 0. Now, the scalar addition with $1^{\pm p}$ is described by

$$\begin{aligned} 1^{+p} + 1^{+p'} &= 0 \\ 1^{-p} + 1^{-p'} &= 0 \\ 1^{\pm p} + 1 &= 1 + 1^{\pm p} = 0 \\ 1^{+p} + 0 &= 0 + 1^{+p} = 1^{+p} \\ 1^{-p} + 0 &= 0 + 1^{-p} = 1^{-p} \end{aligned} \tag{5.3}$$

No addition is performed for the scalar addition of different signs of superscript ($1^{+p} + 1^{-p'} = 1^{-p'} + 1^{+p}$) because half edges are oriented in different directions. The scalar addition is commutative and associative, therefore any further scalar addition with $(1^{+p} + 1^{-p'})$ are performed as described by (5.3).

Let vectors $\mathcal{E}_A$ and $\mathcal{E}_B$ in vector space $\mathcal{W}_\mathcal{E}$ be the vectors corresponding to partially explored edge sets of two sources A and B respectively. Let vector $\mathcal{E}_{AB}$ be the vector corresponding to the intersection of edge sets $\mathcal{E}_A$ and $\mathcal{E}_B$ (edge set $\mathcal{E}_A \cap \mathcal{E}_B$). Since $\mathcal{E}_A \cap \mathcal{E}_B \subseteq \mathcal{E}$, we have $\mathcal{E}_{AB} \in \mathcal{W}_\mathcal{E}$. Now, we define merge operation $\boxplus$ of vectors $\mathcal{E}_A$ and $\mathcal{E}_B$ as follows:

$$\mathcal{E}_M = \mathcal{E}_A \boxplus \mathcal{E}_B = \mathcal{E}_A \oplus \mathcal{E}_B \oplus \mathcal{E}_{AB}. \tag{5.4}$$

Since the vector $\mathcal{E}_M \in \mathcal{W}_\mathcal{E}$, the merge operation is closed. Let vectors $\mathcal{E}_B$ and $\mathcal{E}_{AB}$ be given by

$$\mathcal{E}_B = \sum_{i=1}^{E} g_i \mathrm{e}_i, \text{ and} \tag{5.5}$$

$$\mathcal{E}_{AB} = \sum_{i=1}^{E} h_i \mathrm{e}_i. \tag{5.6}$$

Table 5.1 describes the merge operation for each combination of i^{th} element f_i and g_i corresponding to the vectors $\mathcal{E}_A$ and $\mathcal{E}_B$ respectively. It is worth noting that a particular combination of f_i and g_i describes h_i (corresponding to the i^{th} element of vector $\mathcal{E}_{AB}$). Scalar addition is associative and commutative, therefore combination resulting from applying these properties are not mentioned in the table. Now, the vector corresponding to the combined partially explored graph, whenever communications from sources A and B are available, is given by (5.4).

Remark 5.3 The vector $\mathcal{E}_A \boxplus \mathcal{E}_B$ corresponds to the edge set $\mathcal{E}_A \cup \mathcal{E}_B$.

case	f_i	g_i	$f_i + g_i$	h_i	$f_i + g_i + h_i$
1	0	0	0	0	0
2	0	1	1	0	1
3	0	1^{+p}	1^{+p}	0	1^{+p}
4	0	1^{-p}	1^{-p}	0	1^{-p}
5	1	1	0	1	1
6	1	1^{+p}	0	1	1
7	1	1^{-p}	0	1	1
8	1^{+p}	$1^{+p'}$	0	$1^{+max(p,p')}$	$1^{+max(p,p')}$
9	1^{+p}	$1^{-p'}$	$1^{+p} + 1^{-p'}$	0	$1^{+p} + 1^{-p'}$
10	1^{-p}	$1^{-p'}$	0	$1^{-max(p,p')}$	$1^{-max(p,p')}$

Table 5.1: Merge operation

Example 5.4 *Let $\mathcal{E}_A$ = (1,1,0,0) and $\mathcal{E}_B$ = (1,0,1,0) be two edge sets associated with two sources A and B respectively. Then, the merge operation between these two edge sets is given by,*

$$\begin{aligned}
\mathcal{E}_M &= \mathcal{E}_A \boxplus \mathcal{E}_B, \\
&= \mathcal{E}_A \oplus \mathcal{E}_B \oplus \mathcal{E}_{AB}, \\
&= (1,1,0,0) \oplus (1,0,1,0) \oplus (1,0,0,0), \\
&= (0,1,1,0) \oplus (1,0,0,0), \\
&= (1,1,1,0), \\
&= \mathcal{E}_A \cup \mathcal{E}_B. \qquad (5.7)
\end{aligned}$$

The vector associated with a robot, $\mathcal{E}_A$ is also added with that of a vertex v_j when it reaches the vertex v_j. The vector $\mathcal{E}_{v_j}$ associated with each vertex v_j is obtained by collecting the information of its E' number of incident edges and given by $\mathcal{E}_{v_j} = \sum_{i=1}^{E} l_i \mathrm{e}_i$, where

$$l_i = \begin{cases} 1^{\pm 0} & \text{if } \mathrm{e}_i \in \{\mathrm{e}_{j_1}, \mathrm{e}_{j_2}, \ldots, \mathrm{e}_{j_{E'}}\}, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, the robot updates its vector $\mathcal{E}_A$ at the arrival of a vertex v_j by traversing an edge

$\mathrm{e}_k \in \{\mathrm{e}_{j_1}, \mathrm{e}_{j_2}, \ldots, \mathrm{e}_{j_{E'}}\}$ and the updated vector $\mathcal{E}_A^{new}$ shown in Fig. 5.2 is obtained using (5.8).

$$\mathcal{E}_A^{new} = \mathcal{E}_A \boxplus \mathrm{e}_k \boxplus \mathcal{E}_{v_j}. \tag{5.8}$$

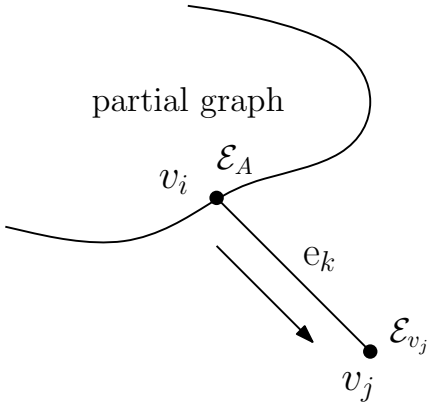


Figure 5.2: Edge set update

For example, if the robots R_1 and R_2 as shown in Fig. 3.1 communicate using beacons at the vertices and beacons are updated at the robot's visit, then the edge sets $\mathcal{E}_{R_1}$ and $\mathcal{E}_{R_2}$ at the time of traversing edges $\mathrm{e}_7(e_{46})$ and $\mathrm{e}_2(e_{12})$ by robots R_1 and R_2 respectively are given by,

$$\mathcal{E}_{R_1} = (1^{+0}, 1, 1, 1^{+0}, 0, 1^{+0}, 1^{+1}, 0, 0, 0),$$
$$\mathcal{E}_{R_2} = (1^{+0}, 1, 1^{+1}, 1, 1, 0, 0, 0, 1^{+0}, 0).$$

The edges $\mathrm{e}_1(e_{13}), \mathrm{e}_6(e_{34})$ and $\mathrm{e}_9(e_{68})$ are the half edges $e_{1\mathrm{x}}$, $e_{4\mathrm{x}}$ and $e_{8\mathrm{x}}$ recorded by the robots at the vertices v_1, v_4 and v_8 respectively. ($\mathcal{E}_{R_2}$ is updated using beacon at v_2 that was visited by robot R_1 at a previous instance.)

The edge sets with the beacons, $\mathcal{E}_{\mathcal{B}_k}$ for $k = 1, \ldots, 5$ placed at visited vertices v_1, v_2, v_4, v_7 and v_8 respectively at the instance shown in Fig. 3.1 are given by,

$$\mathcal{E}_{\mathcal{B}_1} = (1^{+0}, 1^{+1}, 0, 0, 0, 0, 0, 0, 0, 0),$$
$$\mathcal{E}_{\mathcal{B}_2} = (1^{+0}, 1, 1^{+1}, 1, 1, 0, 0, 0, 1^{+0}, 0),$$
$$\mathcal{E}_{\mathcal{B}_3} = (1^{+0}, 1, 1, 1^{+0}, 0, 1^{+0}, 1^{+1}, 0, 0, 0),$$
$$\mathcal{E}_{\mathcal{B}_4} = (0, 0, 0, 1^{+1}, 1, 0, 0, 0, 1^{+0}, 0),$$
$$\mathcal{E}_{\mathcal{B}_5} = (0, 0, 0, 0, 1^{+1}, 0, 0, 0, 1^{+0}, 0).$$

There may exist multiple components in the graph $\mathcal{G}_A$ obtained from edge set $\mathcal{E}_A$ (refer to Remark 5.1) of a source A. For example, multiple components in the graph $\mathcal{G}_A$ would exist when

the robots begin exploration from different places and communicate by broadcasting. Partial graph explored by each robot in multi robot exploration refers to a component in the graph obtained by edge set $\mathcal{E}_A$ of any source A, therefore we provide a component-wise perspective of merge operation described by (5.4) that supports implementation using incidence matrix described in Section 4.3.

Let the number of components present in the graphs obtained by edge set $\mathcal{E}_A$ and $\mathcal{E}_B$ (refer Remark 5.1) be a and b respectively. Let vectors $\mathcal{E}_A$ given by (5.1) and $\mathcal{E}_B$ given by (5.5) be also described by

$$\mathcal{E}_A = A_1 \boxplus A_2 \boxplus \ldots \boxplus A_a, \tag{5.9}$$

$$\mathcal{E}_B = B_1 \boxplus B_2 \boxplus \ldots \boxplus B_b, \tag{5.10}$$

where vector A_j for $j = 1, \ldots, a$ corresponds to a component in the graph obtained from $\mathcal{E}_A$ and vector B_j for $j = 1, \ldots, b$ corresponds to that from $\mathcal{E}_B$. Now, (5.4), (5.9) and (5.10) render

$$\mathcal{E}_M = A_1 \boxplus A_2 \boxplus \ldots \boxplus A_a \boxplus B_1 \boxplus B_2 \boxplus \ldots \boxplus B_b. \tag{5.11}$$

Fig. 5.3 shows the construction of iterative component-wise perspective of merge operation. In

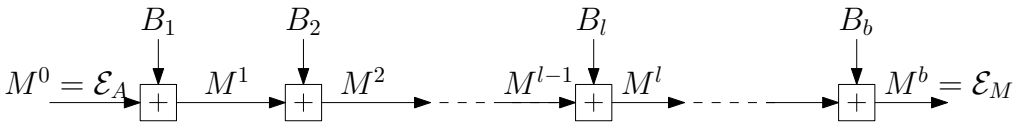


Figure 5.3: Component wise merge operation

Fig. 5.3, we consider

$$M^l = M^{l-1} \boxplus B_l \text{ for } l = 1, \ldots, b \text{ with} \tag{5.12}$$

$$M^0 = \mathcal{E}_A = A_1 \boxplus A_2 \boxplus \ldots \boxplus A_a.$$

From (5.11) and (5.12), we have

$$\mathcal{E}_M = M^b. \tag{5.13}$$

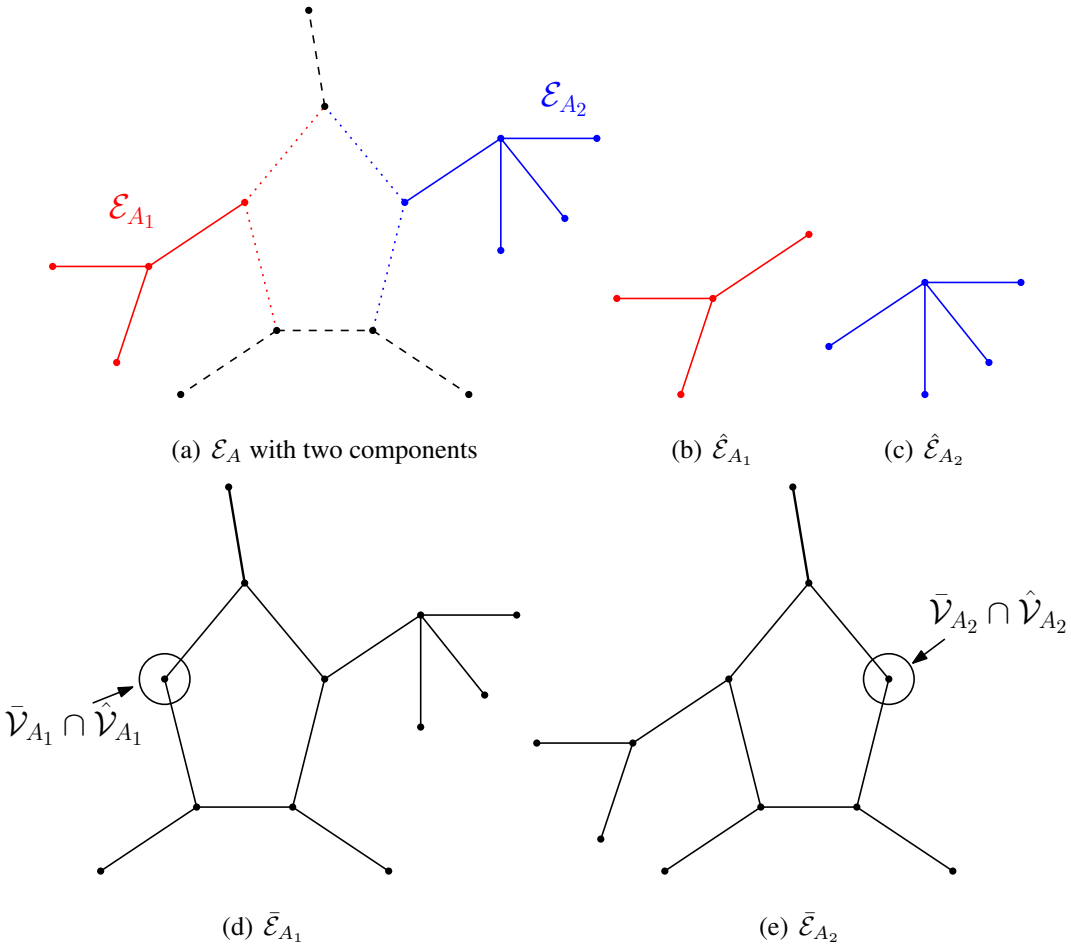
Now, we analyse the components of edge set M^1. Using (5.12), we have

$$M^1 = A_1 \boxplus A_2 \boxplus \ldots \boxplus A_a \boxplus B_1. \tag{5.14}$$

Let the edge set B_1 has at least one common edge (or common vertex) with k components of edge set $\mathcal{E}_A$, where $0 \leq k \leq a$. Therefore, the number of components, m^1 in the graph obtained

by edge set M^1 is $a+1-k$. In particular, $1 \leq m^1 \leq a+1$. Proceeding in the similar manner, for the m^b components present in the graph obtained by edge set M^b, we have $1 \leq m^b \leq a+b$.

We define a vector $\hat{\mathcal{E}}_A$ obtained from vector $\mathcal{E}_A$ given by (5.1) by substituting $f_i = 0$ for each $f_i = 1^{\pm p}$ for any p and illustrate using Fig. 5.4(a). The numbers of components of the graphs obtained by edge sets $\mathcal{E}_A$ and $\hat{E}_A$ are equal. Let the components present in the graph $\hat{E}_A$ be $\hat{E}_{A_j}$ for $j = 1, \ldots, a$. Fig. 5.4(a) shows an example edge set $\mathcal{E}_A$ having two components $\mathcal{E}_{A_1}$ and $\mathcal{E}_{A_2}$. The half edges of the edge set $\mathcal{E}_A$ are shown using dotted lines and dashed lines show the unidentified edges in edge set $\mathcal{E}$. Figures 5.4(b) and 5.4(c) show two components present in the graph obtained by corresponding edge set $\hat{E}_A$.



(a) $\mathcal{E}_A$ with two components (b) $\hat{\mathcal{E}}_{A_1}$ (c) $\hat{\mathcal{E}}_{A_2}$

(d) $\bar{\mathcal{E}}_{A_1}$ (e) $\bar{\mathcal{E}}_{A_2}$

Figure 5.4: Components of an example edge set $\mathcal{E}_A$ and their association with the edge set $\mathcal{E}$

Next section presents the analytical results on the necessary and sufficient conditions for finite time completion.

5.2 Analytical Results

The conditions for declaring completion and finite time exploration are obtained using following lemma and theorems using components and the edge sets $\hat{E}_{A_j}$ for $j = 1, \ldots, a$ as explained.

Lemma 5.5 *The edge set corresponding to each component in the graph obtained by $\mathcal{E}_A$ given by* (5.1) *has at least one half edge, if exploration is incomplete.*

Proof Let edge set $\bar{\mathcal{E}}_{A_j}$ for $j = 1, \ldots, n_A$ be given by $\bar{\mathcal{E}}_{A_j} = \mathcal{E} \setminus \hat{\mathcal{E}}_{A_j}$, where operation $\setminus$ denote set difference. Formation of edge sets $\bar{\mathcal{E}}_{A_j}$ for $j = 1, 2$ of the example graph are shown in Figs. 5.4(d) and 5.4(e).

Since $\mathcal{G}$ is connected and exploration is incomplete, the corresponding vertex-sets $\bar{\mathcal{V}}_{A_j}$ and $\hat{\mathcal{V}}_{A_j}$ (vertex-set is obtained by considering both the vertices of each edge in the edge set) has following property: $\bar{\mathcal{V}}_{A_j} \cap \hat{\mathcal{V}}_{A_j} \neq \emptyset$ for each $j = 1, \ldots, a$. Common vertices in vertex-sets $\bar{\mathcal{V}}_{A_j}$ and $\hat{\mathcal{V}}_{A_j}$ for $j = 1$ and 2 are shown using circles in Figs. 5.4(d) and 5.4(e) respectively. Since each common vertex of $\bar{\mathcal{V}}_{A_j}$ and $\hat{\mathcal{V}}_{A_j}$ is explored, the incident edges in $\bar{\mathcal{E}}_{A_j}$ at each common vertex are half edges in $\mathcal{E}_{A_j}$.

Theorem 5.6 *For any vector $\mathcal{E}_A \in \mathcal{W}_{\mathcal{E}}$ described by* (5.1)*, $f_i = 1 \; \forall \; i = 1, \ldots, E$ if and only if $f_i \neq 1^{\pm p}$ for each i and for any p after initialization.*

Proof When $f_i = 1 \; \forall \; i = 1, \ldots, E$ for a vector $\mathcal{E}_A$, we have $f_i \neq 1^{\pm p}$ for any i. Now, we show that the converse is also true. In particular, when $f_i \neq 1^{\pm p} \; \forall \; i = 1, \ldots, E$, we show that $f_i \neq 0 \; \forall \; i = 1, \ldots, E$.

From Lemma 5.5, the condition $f_i \neq 1^{\pm p} \; \forall \; i = 1, \ldots, E$ for any p implies either vertex-set $\mathcal{V}_A$ obtained from edge set $\mathcal{E}_A$ is null or the exploration is complete. This implies either vector $\mathcal{E}_A = \vec{0}$ or $\mathcal{E}_A = \mathcal{E}$.

The vector $\mathcal{E}_A = \vec{0}$ is true only before initialization and $\mathcal{E}_A = \mathcal{E}$ implies $f_i = 1 \; \forall \; i = 1, \ldots, E$. In particular, $f_i \neq 0 \; \forall \; i = 1, \ldots, E$ after initialization, if $f_i \neq 1^{\pm p} \; \forall \; i = 1, \ldots, E$ for any p.

Corollary 5.7 *For any vector $\mathcal{E}_A$ given by* (5.1)*, the condition $f_i \neq 1^{\pm p}$ for any $i = 1, \ldots, E$ and any p, is the necessary and sufficient condition for completion.*

Proof From the proof of Theorem 5.6, we have $f_i \neq 1^{\pm p} \; \forall \; i$ implies either $\mathcal{E}_A = \vec{0}$ or $\mathcal{E}_A = \vec{1}$. $\mathcal{E}_A = \vec{0}$ holds before the initialization and $\mathcal{E}_A = \mathcal{E} = \vec{1}$ holds at the completion.

When exploration is performed using multiple robots, it is important to design a systematic strategy that ensures completion in finite time. At the arrival of a vertex, the strategy selects an edge to be explored next. Searching a path to reach this new edge to be explored must avoid problems such as infinite loop in the path of a single robot, unable to find paths to explore new edges particularly in the presence of multiple components, etc. Following theorem establishes condition for completing exploration in finite time.

Theorem 5.8 *The exploration of an unknown graph $\mathcal{G}$ is complete in finite time if the next edge to be explored is selected from the half edges of a component which contains the edge currently explored.*

Proof Let a subgraph of the graph $\mathcal{G}_A$ obtained from edge set $\mathcal{E}_A$ be described by the edge set $\hat{\mathcal{E}}_A$ (refer to remark 5.1) and the corresponding subspace be $\mathcal{W}_{\hat{\mathcal{E}}_A}$. Let fundamental basis β_i of the subspace $\mathcal{W}_{\hat{\mathcal{E}}_A}$ be

$$\beta_i = \{\mathrm{e}_j \mid f_j = 1 \; \forall \; j = 1, \ldots, E\}. \tag{5.15}$$

Clearly, we have $\dim(\mathcal{W}_{\hat{\mathcal{E}}_A}) = E_A$, where E_A is the number of edges (excluding half edges) in edge set $\mathcal{E}_A$. From Theorem 5.6, we have $E_A < E$ if exploration is incomplete. When next edge to be explored is a half edge from connected component, the path from the robot's current location to the origin of selected half edge is known. Let the selected half edge be e_k. Now, the vector $\mathcal{E}_A$ is updated using following two ways:

1. When the robot explores the edge e_k. The vector $\mathcal{E}_A$ is updated using $\mathcal{E}_A^{new}$ given by (5.8).

2. When the vector $\mathcal{E}_A$ is updated using (5.4) with $\mathcal{E}_B$ given by (5.5) having $g_k = 1$ and assigning $\mathcal{E}_A = \mathcal{E}_M$.

Since one among above two conditions occur in finite time, the selected half edge is explored ($f_k = 1$ in (5.1)) in finite time. Let the subspace associated with the subgraph obtained by edge set $\hat{\mathcal{E}}_A^{new}$ as in (5.15) be $\mathcal{W}_{\hat{\mathcal{E}}_A}^{new}$. Now, we have $\dim(\mathcal{W}_{\hat{\mathcal{E}}_A}^{new}) \geq \dim(\mathcal{W}_{\hat{\mathcal{E}}_A}) + 1$ in finite time. Since, the updated edge set $\mathcal{E}_A$ is $\mathcal{E}_A^{new}$, we have $\dim(\mathcal{W}_{\hat{\mathcal{E}}_A}) = E$ ($\mathcal{E}_A = \mathcal{E}$) in finite time as E is finite, if selection for next edge to be explored is a half edge from connected component in the graph obtained by edge set $\mathcal{E}_A$.

It is worth noting that each and every vertex $v_j \; \forall \; j = 1, \ldots, V$ is explored if $\dim(\mathcal{W}_{\mathcal{E}_A}) = E$ for a connected graph $\mathcal{G}$ (despite unknown V and E). Next section describes the generic algorithm

for various multi robot exploration and mapping techniques such as DFS [1, 51], LRV [5], BFS [3, 4] and Flooding algorithm [13] using modified incidence matrix.

5.3 Algorithm for the Generic Framework

Incidence matrix described in Section 3.1 renders a data structure that encodes the information associated with the subgraph $\mathcal{G}_A$. An incidence matrix associated with source A, I_A of size $V_A \times E_A$ represents edges as column vectors. An edge e_j connected at vertex v_i has non-zero entry d_{ij} in incidence matrix I_A. Each element of incidence matrix [51], d_{ij} also accumulates weight, p associated with each half edge. Therefore, each element d_{ij} of incidence matrix I_A is given by,

$$d_{ij} = \begin{cases} -(\theta_{ij} + 2p\pi) & \text{if } f_j = 1^{\pm p} \text{ with } p > 0 \text{ or } f_j = 1, \\ +\theta_{ij} & \text{if } f_j = 1^{\pm 0}, \\ 0 & \text{otherwise.} \end{cases}$$

Each source A performs following operations in sequence at the arrival of a vertex or any communication with any source after initialization as described in [51]: *Merge, Order, Next_edge_selection*. However, work in [51] describes the operations for a specific application where graph obtained by edge set $\mathcal{E}_A$ from any source A can have only one component. The generalized framework presented in this work addresses multiple components in the graph obtained by edge set $\mathcal{E}_A$. Therefore, the corresponding incidence matrix I_A is rearranged to perform merge operation described by (5.12). The merge operation described by (5.12) is performed iteratively on vectors M^{l-1} and B_l, for $l = 1, \ldots, b$. The vector B_l for each $l = 1, \ldots, b$ represents a component in the graph obtained by edge set $\mathcal{E}_B$, however the vector M^{l-1} can have more than one components. The merge operation using incidence matrices is performed on incidence matrices I_M^{l-1} (corresponding to vector M^l) and I_l (corresponding to the vector B_l). It is worth noting that the incidence matrix I_M^{l-1} can be represented as block diagonal form given by,

$$I_M^{l-1} = \begin{bmatrix} I_{M_1}^{l-1} & & \mathbf{0} & \\ & I_{M_2}^{l-1} & & \\ & & \ddots & \\ & \mathbf{0} & & I_{M_{m^{l-1}}}^{l-1} \end{bmatrix},$$

where m^{l-1} is the number of components in the graph obtained by edge set M^{l-1} and each block $I^{l-1}_{M_k}$ for $k = 1, \ldots, m^{l-1}$ corresponds to each component M^{l-1}_k. The incidence matrix I^{l-1}_M is rearranged such that the components connected with B_l are stacked at lower part of the diagonal. The rearrangement of the incidence matrix for Merge operation (described in [51]) is presented using Algorithm 5. The function *vertex_set()* returns the set of vertices in the graph associated with incidence matrix used in its argument. The deletion of a copy of an edge is same

Algorithm 5 *Rearrange*

Input: I^{l-1}_M having m^{l-1} blocks and I_l

Output: Rearranged incidence matrix I^{l-1}_M and number of component in next iteration m^l

Functions: *vertex_set*().

1: $k = 1$, $j = m^{l-1}$, $\mathcal{V}_1 =$ *vertex_set*(I_l).
2: **do**
3: $\quad \mathcal{V}_2 =$ *vertex_set*$(I^{l-1}_{M_k})$.
4: $\quad$ **if** $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ **then**
5: $\quad\quad k = k + 1$.
6: $\quad$ **else**
7: $\quad\quad$ Interchange blocks $I^{l-1}_{M_k}$ and $I^{l-1}_{M_j}$
8: $\quad\quad j = j - 1$.
9: **while** $k < j$ **do**
10: $m^l = j + 1$.

as that described in Merge operation in [51]. Merge operation preserves ordering of individual matrices, but removes copies of common edges and vertices. The connected half edges are placed at the rightmost side of the merged incidence matrix if the assignment is as follows:

$$(I_1, I_2) = \begin{cases} (I_A, I_{v_j}), & \text{when updated using (5.8),} \\ (I_B, I_A), & \text{when updated using (5.4) and} \\ & \mathcal{E}^{new}_A = E_m, \\ (I_A, I_B), & \text{when updated using (5.4) and} \\ & \mathcal{E}^{new}_B = E_m. \end{cases} \tag{5.16}$$

Ordered assignment for updating the incidence matrix as described in (5.16) ensures that the highest priority is given to the half edges connected to the current vertex. Further, half edges

are ordered based on the weight p. Proposed ordering of the edges of a component in the graph represented by vector $\mathcal{E}_A$ is as follows: $\{\mathrm{e}_i, e_j \mid f_i = 1, f_j = 1^{\pm p}\}$. In particular, priority is based on half edges at current vertex and the weight p. Let the function *ordered_half*() returns the ordered columns of half edges based on the decreasing order of p such that the half edges at the current vertex is given highest priority. Pseudo code to order edges of a component is described using Algorithm 6.

Algorithm 6 *Order*

Input: Merged incidence matrix I of dimension $\hat{V} \times (\hat{E}_1 + \hat{E}_2)$.

Output: Ordered incidence matrix I^{new}.

Functions: *concatenate*(), *explored*(), *ordered_half*(), *size*().

1: $I_1 = I[:, 1 : \hat{E}_1], I_2 = I[:, \hat{E}_1 + 1 : \hat{E}_2]$.

2: I^{new}= *concatenate*(*explored*(I_1), *explored*(I_2), *ordered_ half*(I_1), *ordered_half*(I_2)).

The algorithm describing the sequence of operations for component-wise merge is described using Algorithm 7. Line 6 of the Algorithm 7, maintains the order of only lowest block of each incidence matrix I_M^l for $l = 1, \ldots, b$, because this block corresponds to the connected component of block I_l.

Algorithm 7 *Component-merge*

Input: I_A having a blocks and I_B having b blocks given by I_l, $l = 1, 2, \ldots, b$

Output: Merged incidence matrix I_M having m blocks

Functions: *Rearrange*(), *Order*(), *Number_of_rows*()x, *Number_of_columns*() .

1: $I_M^0 = I_A$, $m^0 = a$

2: **for** $l = 1$ to b **do**

3: $[I_M^{l-1}, m^l] =$ *Rearrange*$(I_M^{l-1}, m^{l-1}, I_l)$

4: $I_M^l =$ *Merge*(I_M^{l-1}, I_l)

5: $\hat{V}$ = *Number_of_rows*$(I^l_{M_{m^l}})$, $\hat{E}_2$ = *Number_of_ columns*(I_l) and $\hat{E}_1$ = *Number_of_columns*$(I^l_{M_{m^l}}) - \hat{E}_2$

6: $I^l_{M_{m^l}} =$ *Order*$(I^l_{M_{m^l}}, \hat{V}, \hat{E}_1, \hat{E}_2)$

7: $I_M = I_M^b$ and $m = m^b$.

Some modifications are made in the *Merge* algorithm to incorporate Cases 8 and 10 in Table 5.1. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be the vertex_tags of the incidence matrices $I^n(R_1)$ and $I^n(R_2)$

respectively and $\mathcal{V}_{12}$ be the set of common vertices. While communicating using beacons at each visited vertex, the problem of matching vertex and edge can be solved by assigning unique ID to each vertex. For example robot R_1 assigns a_1, a_2, a_3, etc., robot R_2 assigns b_1, b_2, b_3, etc. and so on. The rows of the incidence matrices corresponding to the common vertices are placed at the middle of the merged incidence matrix I. The rows corresponding to the distinct vertices in $I^n(R_1)$ and $I^n(R_2)$ are placed at the top and bottom of the rows with common vertex_tags respectively. Thus, this incidence matrix merging arrangement requires only column deletion (does not require any row deletion) to remove duplicate edges. The proposed arrangement for the merged incidence matrix (I) is represented as,

$$I = \begin{bmatrix} I^n(R_1)[\mathcal{V}_1 \setminus \mathcal{V}_{12}, ;] & 0 \\ I^n(R_1)[\mathcal{V}_{12}, ;] & I^n(R_2)[\mathcal{V}_{12}, ;] \\ 0 & I^n(R_2)[\mathcal{V}_2 \setminus \mathcal{V}_{12}, ;] \end{bmatrix}. \tag{5.17}$$

From the merged incidence matrix I, the duplicate columns in the rows corresponding to the common vertex_tags ($\mathcal{V}_{12}$) are identified and the entries in each of these columns are copied to preserve the edge information. After copying the values, one of the duplicate columns is deleted. Example 5.9 demonstrates the proposed incidence matrix *Merge* operation.

Example 5.9 *Let us consider the robot graph traversal scenario shown in Fig. 4.9 with robot incidence matrices $I^2(R_1)$ and $I^2(R_2)$ as,*

$$I^2(R_1) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 \\ -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} \end{bmatrix} \end{array},$$

$$I^2(R_2) = \begin{array}{c} \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccc} e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \\ 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \end{bmatrix} \end{array}.$$

The vertex_tags $\mathcal{V}_1$, $\mathcal{V}_2$ associated with the robot incidence matrices $I^2(R_1)$, $I^2(R_2)$ and the set of common vertices $\mathcal{V}_{12}$ are,

$$\mathcal{V}_1 = \{v_1, v_2, v_3\},\ \mathcal{V}_2 = \{v_1, v_4, v_3\} \text{ and } \mathcal{V}_{12} = \{v_1, v_3\}.$$

The merged incidence matrix obtained by arranging the two ordered incidence matrices $I^2(R_1)$ and $I^2(R_2)$ is given by,

$$I = \begin{array}{c} \\ v_2 \\ v_1 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} & 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \end{bmatrix} \end{array},$$

In 2^{nd} row with vertex_tag v_1, the columns 1 & 9 corresponding to edge e_{12} and columns 3 & 7 corresponding to edge e_{14} are the duplicate entries of the same edge. Therefore, the entries of these columns are copied into each other and columns 9 & 3 are recorded for deletion. The incidence matrix resulted from these operations is,

$$I = \begin{array}{c} \\ v_2 \\ v_1 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & -\theta_{21} & 0 & 0 \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & \theta_{34} & \theta_{36} & 0 & -\theta_{34} & 0 & \theta_{32} & \theta_{36} \\ 0 & 0 & -\theta_{41} & 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \end{bmatrix} \end{array},$$

In row 3 with vertex_tag v_3, the columns 2 & 10 corresponding to edge e_{23}, columns 5 & 8 corresponding to edge e_{34} and columns 6 & 11 corresponding to edge e_{36} are the duplicate entries of the same edge. Therefore, the entries of these columns are copied into each other and the columns 10, 5 & 6 are recorded for deletion. The obtained incidence matrix is,

$$I = \begin{array}{c} \\ v_2 \\ v_1 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} e_{12} & e_{23} & e_{14} & e_{25} & e_{34} & e_{36} & e_{14} & e_{34} & e_{12} & e_{23} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & 0 & \theta_{25} & 0 & 0 & 0 & 0 & -\theta_{21} & -\theta_{23} & 0 \\ -\theta_{12} & 0 & -\theta_{14} & 0 & 0 & 0 & -\theta_{14} & 0 & -\theta_{12} & 0 & 0 \\ 0 & -\theta_{32} & 0 & 0 & -\theta_{34} & \theta_{36} & 0 & -\theta_{34} & 0 & -\theta_{32} & \theta_{36} \\ 0 & 0 & -\theta_{41} & 0 & -\theta_{43} & 0 & -\theta_{41} & -\theta_{43} & 0 & 0 & 0 \end{bmatrix} \end{array},$$

After removing the recorded columns (9, 3, 10, 5 and 6) from I, the obtained un-ordered merged

incidence matrix is given by,

$$I = \begin{array}{c} \\ v_2 \\ v_1 \\ v_4 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{23} & e_{25} & e_{14} & e_{34} & e_{36} \end{array} \\ \begin{bmatrix} -\theta_{21} & -\theta_{23} & \theta_{25} & 0 & 0 & 0 \\ -\theta_{12} & 0 & 0 & -\theta_{14} & 0 & 0 \\ 0 & 0 & 0 & -\theta_{41} & -\theta_{43} & 0 \\ 0 & -\theta_{32} & 0 & 0 & -\theta_{34} & \theta_{36} \end{bmatrix} \end{array}.$$

Remark 5.10 The incidence matrix is ordered, therefore termination condition needs to be checked with just the rightmost column vector.

Remark 5.11 There exists a connected half edge in I_A for which shortest path can be identified from I_A and the known vertex of selected edge can be reached using the shortest path. This is ensured from Lemma 5.5 and ordering columns of the incidence matrix I_A.

The next section describes the proposed technique to obtain an optimal fleet size for multi robot exploration.

5.4 Number of Robots

The time to complete the exploration depends on the number of robots (team size) used for exploration. Finding an analytical expression for number of robots describing its relation with a given strategy, information exchange method and topology is challenging as instantaneous decision is based on partially explored graph and completion time depends on the graph structure. An iterative method of finding the optimal number of robots is proposed based on the outcome of graph exploration using two robots.

The number of edge traversals N required to complete the exploration is obtained using various strategies by varying the number of robots K used in exploration. It is worth noting that each simulation run for a particular value of K may result in different number of edge traversals. However, variation in these results is small as compared to the variation obtained by changing the value of K. The relationship between N and K suggests that there does not exist a significant benefit from increasing K after a certain value. This limiting value for minimum exploration time is referred as saturation value K_{sat}. We propose computing the optimal number

of robots K_{opt} aiming to achieve K_{sat} using an iterative simulation method. The motivation for the proposed method is as follows: the number of half edges at a particular step of exploration is indicative to the deficiency in the number of robots.

Let the exploration of an unknown graph be completed in N_i edge traversals using K_i robots for a positive integer i. We propose finding the optimal number of robots for exploring the same graph using the knowledge of number of unexplored half edges (represented by $f_i = 1^{\pm 0}$ in (5.1)) during the exploration. Let the number of unexplored half edges of a source A at a particular edge traversal n (exploration step) be $H_A(n)$. Now, the optimal number of robots is computed using following iterative procedure:

$$K_i = K_{i-1} + \Delta K_i, \text{ where } \Delta K_i = \left(\max_{1 \leq n \leq N_{i-1}} \sum_{k=1}^{K_{i-1}} H_A(n) \right) \tag{5.18}$$

with $K_0 = 2$. The optimal number of robots K_{opt} is determined using least iterative value K_i that satisfies $K_i - K_{i-1} \leq 1$ The number of half edges at a particular instant of exploration indicates insufficient number of robots for fastest exploration, the condition $\Delta K_i = 0$ indicates $K_{i-1} = K_{sat}$ (corresponding number of robots N_{sat} represents fastest exploration).

Since the total energy is proportional to the number of robots, K and completion time (proportional to N), a performance measure η defined using the product $N \times K$ given by (5.19) is used for our analysis,

$$\eta = 1 - \frac{N_0 K_0}{NK}. \tag{5.19}$$

This performance measure indicates the improvement over the case with minimum number of robots ($K_0 = 2$) for a multi robot application. Section 5.5 illustrates the simulation results for the proposed generic framework.

5.5 Simulation Results

Simulations are carried out using MATLAB ver R2014b on various tree and graph topologies with variable number of robots and the results for the performance measure $\%\eta$ are presented. The proposed iterative approach to obtain the optimal fleet size has been verified for various strategies using these results. The proposed modifications in incidence matrix incorporates the multi robot graph exploration using existing decentralized strategies namely multi robot DFS, IMRDFS, LRV and BFS. The partially explored graph is communicated using beacons

(source) at each vertex in simulations. Each source (beacon and the robot) updates the edge-sets, whenever a robot visits a vertex.

Following representative graph topologies [49] are considered:

- Complete graph (unit root radius) with 45 edges
- Ladder graph (larger root radius) with 19 edges
- Square lattice (symmetric graph structure) with 60 edges
- 3-ary tree with 39 edges
- Binary tree with 62 edges
- Arbitrary tree structure (distinct height and width) with 39 edges as shown in Fig. 5.5

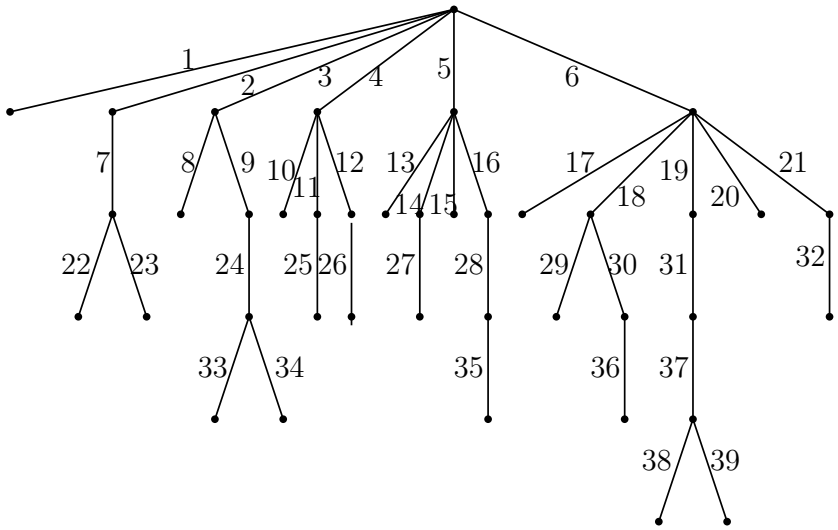


Figure 5.5: Arbitrary tree structure with 39 edges

Figs. 5.6, 5.7, 5.8 and 5.9 depict the variation in the percentage measure ($\%\eta$) computed for MRDFS, IMRDFS, LRV and BFS approaches on representative graph topologies respectively. An average value of N is computed from multiple runs for each K varied from 2 to 100 in steps of 5 and data points are connected using a smooth curve. The iterative values of $\%\eta$ are obtained till the computation of K_{opt} using (5.18) and marked in Figs. 5.6, 5.7, 5.8 and 5.9 for $K_0 = 2$. Figs. 5.6, 5.7, 5.8 and 5.9 show that the maximum change in $\%\eta$ (from K_0) is observed for the number of robots K resulted from the first few iterations and the number of iterations required to reach the optimal value depends on width and depth of the graph. Similar

results are obtained when K_0 is equal to the number of edges incident on (degree of) root vertex.

.

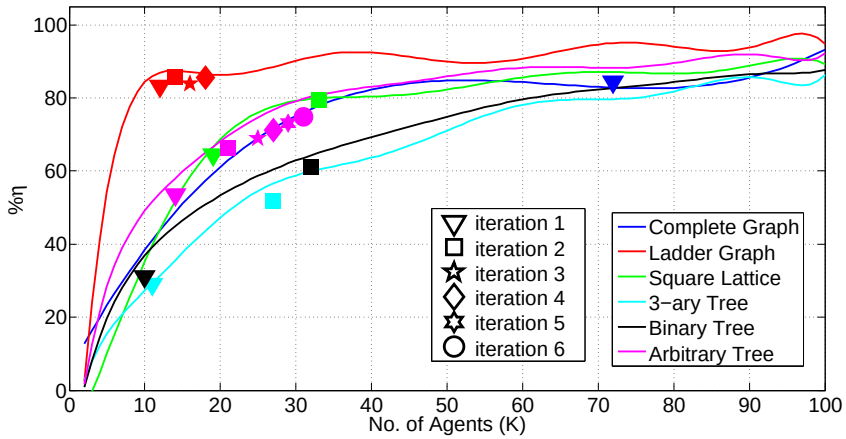


Figure 5.6: $\%\eta$ computed for MRDFS strategy on the representative graph topologies; iterative values are marked using various markers

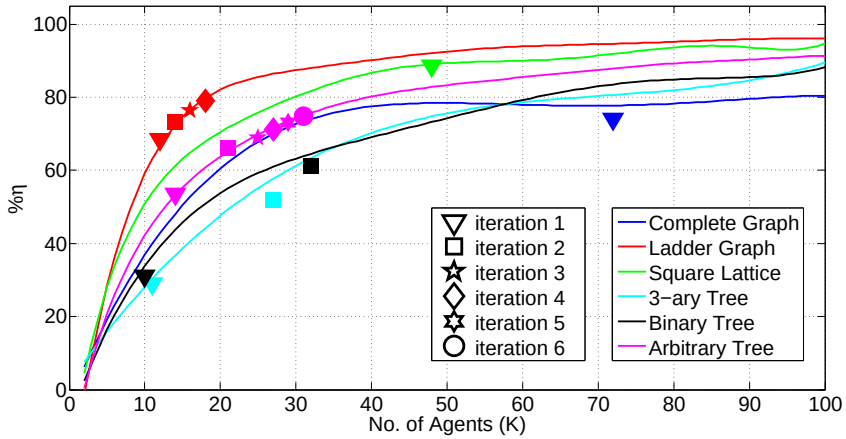


Figure 5.7: $\%\eta$ computed for IMRDFS strategy on the representative graph topologies; iterative values are marked using various markers

Section 5.6 presented next illustrates the performance of the proposed approach with that of the existing ones.

5.6 Comparison with Existing Methods

The proposed iterative method to obtain the optimal fleet size is compared with several existing techniques in the literature. An analytical method [7] for obtaining optimal number of robots

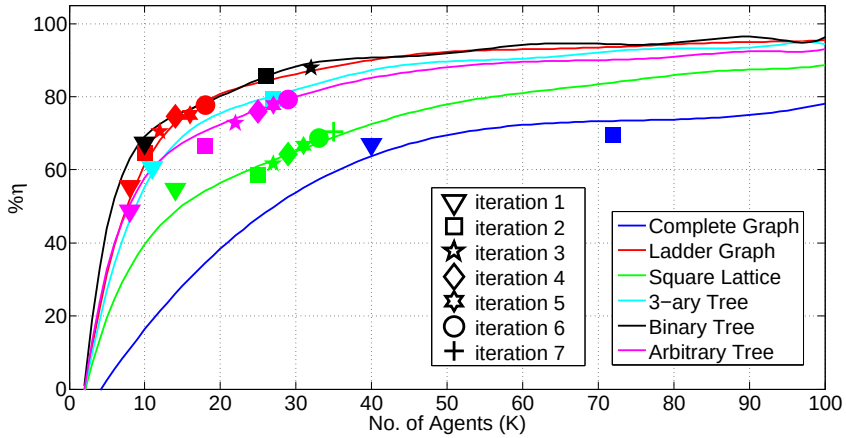


Figure 5.8: $\%\eta$ computed for MRLRV strategy on the representative graph topologies; iterative values are marked using various markers

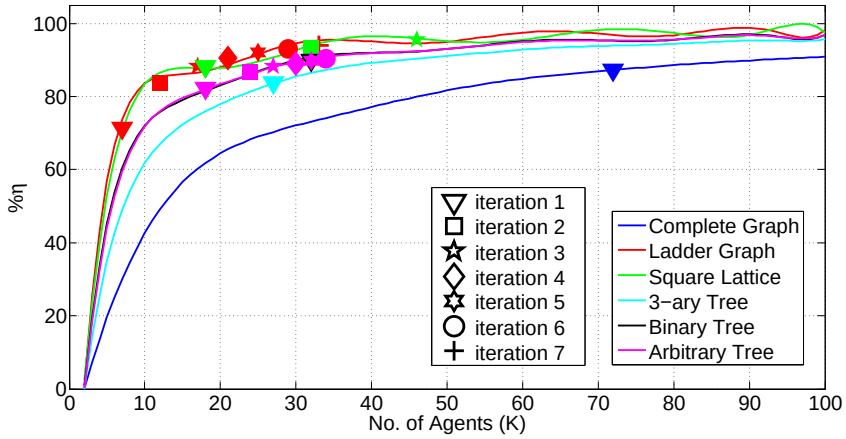


Figure 5.9: $\%\eta$ computed for MRBFS strategy on the representative graph topologies; iterative values are marked using various markers

using decentralized strategy is used to analyze the results. Applicable tree topologies and information exchange method are used for computing the result analytically. In particular MRDFS strategy and information exchange using book keeping devices at explored vertices are used for comparison. Table 5.2 presents the results obtained using the proposed method with $K_0 = 2$ and analytical method [7]. Corresponding percentage measures computed using (5.19) are also presented. Table 5.2 shows that the proposed method provides better results when compared to the analytical method [7]. Moreover, the wider applicability (any graph with or without cycle, any search strategy and any method of information exchange) of proposed method is a main advantage.

The asymptotic behaviour of the number of edge traversals to complete the exploration

Topology	Analytical method [7]			Proposed method		
	K_{opt}	N_{opt}	$\%\eta$	K_{opt}	N_{opt}	$\%\eta$
3-ary Tree	7	16	25	27	6	48.15
Binary Tree	7	28	36.73	32	10	61.25
Arbitrary Tree	7	17	32.77	31	12	79.03

Table 5.2: Comparison with analytical method

with respect to the robot team size for global and local communication with MRDFS [6] has been analyzed. The result [6] is summarized as follows: Exploration of a graph with D depth and V vertices using a robot team of polynomial size $K = DV^{1+\epsilon} < V^{2+\epsilon}$, for $\epsilon > 0$ completes the exploration in N edge traversal, where

$$D(1 + \frac{1}{(1+\epsilon)} - o(1)) < N < D(1 + \frac{1}{\epsilon} + o(1)) \tag{5.20}$$

in global communication mode. The number of robots K_{sat} obtained using proposed technique is used to know if the number of robots are large enough to achieve minimum possible completion time N_{sat}. Using K_{sat} obtained from the proposed method for global communication and MRDFS strategy on various topologies, corresponding N_{sat} are presented in last column of Table 5.3. Various N_{sat} values in Table 5.3 show that the edge traversals are near to the lower bound in (5.20).

Topology	D	V	K_{sat}	N_{sat}
3-ary Tree	3	40	27	6
Binary Tree	5	63	32	10
Arbitrary Tree	5	40	60	10
Ladder Graph	7	14	64	14
Square Lattice	10	36	32	20
Complete Graph	1	10	72	4

Table 5.3: Reasoning based on K_{sat} and N_{sat} obtained from the proposed method and results from the fast collaborative exploration approach

5.7 Conclusion

Graph exploration technique using multiple robots largely depends on decentralized/centralized information exchange method between robots, and search strategy used for visiting unexplored portions of the graph. A generalized framework for multi robot graph exploration is presented in this work to obtain necessary and sufficient conditions for declaring completion and finite time completion using vector space approach. A generic implementation using incidence matrix for systematic multi robot graph exploration is also described. The work analyses the implementation results of various existing search strategies using the generalized framework. The performance of these strategies (realized using the proposed framework) is analyzed for various tree and graph topologies and the results are presented. An application of the generalized framework is investigated in finding the optimal number of robots and minimum completion time for any given combination of graph topology, search strategy, information exchange method and initial placement of robots. Wider applicability of the proposed method is demonstrated by examining existing analytical results for various search strategies. The simulation results show that the proposed optimal fleet size technique is efficient when compared to the existing methods.

Several multi robot tasks such as exploration, surveillance, coverage, etc. need the robots to safely navigate through the environment and complete the assigned task. In order to achieve an efficient task completion, inter robot collisions should be avoided. Each robot in the system must detect the incoming collisions and the necessary avoidance measures should be taken autonomously. The next chapter presents a speed reduction based collision avoidance approach for multi robot exploration.

Chapter 6

Multi Robot Collision Avoidance while Mapping

A primary objective of a multi robot system is to ensure efficient task execution by the robots in any populated or dynamic environments. Multiple robots moving autonomously in an unknown environment while collaborating on a common task must avoid collisions to guarantee safer task execution. So, the robots should be given the ability to perceive their surroundings and react to unanticipated circumstances, to successfully complete their missions. Avoiding collisions with other robots in a decentralized multi robot exploration and mapping task is an important yet challenging problem. This chapter investigates the inter-robot collision avoidance problem in a decentralized scenario. Our approach aims to achieve an elegant solution for decentralized inter-robot collision avoidance by varying the linear speeds of the robots considering minimal sensing and communication requirements. Multiple robots exploring a graph may collide with each other when following a common edge or reaching to a common vertex. Collision avoidance with minimal sensing in a system with decentralized control has the advantage of reduced sensor data processing and computation overhead on the robots.

The following assumptions are made in designing the proposed collision avoidance approach:

1. Robot always follows the left side obstacle. This assumption is made to adopt a convention for robot movements in either direction along the edges, which in turn helps in avoiding additional decision making required at each vertex for lane selection in case of robot forward or backward edge traversals.

2. There is no communication between robots. Moreover, there is no position or speed sensing of robots.

3. Obstacles are considered to be polygonal (to ease the lane structure). We also assume edges as straight lines. The gap between the two adjacent obstacles is sufficient to construct a two lane structure.

4. Robots move with constant speed, step change in speed is only allowed. Any dynamic obstacle present in the environment moves with the constant velocity and follows the lane rule.

5. Sensor on each robot measures the bearing angle of any visible robot/dynamic obstacle. The sensor senses in all the directions and has infinite range. However, in practical scenarios with sensors having limited sensing range, the collision avoidance mechanism is turned on as soon as other robots are visible within the sensor range.

The next section presents the proposed collision detection mechanism using minimal sensing.

6.1 Collision Detection

In multi robot systems, collision detection and avoidance is often achieved by gathering the robots' position and velocity information using a centralized communication system. Whereas, in a multi robot system with a decentralized communication scheme, there is no central unit available to communicate or monitor the inter robot collisions. Therefore, collision avoidance using minimal information in a system with decentralized control is an interesting yet challenging problem in multi robot systems. The main objective is to detect collision (well before actual collision) using just the bearing information of a robot viewed from another robot. Lemma 6.1 shows that the possibility of collision at a vertex is identified using the rate of change of bearing.

Lemma 6.1 *Let the bearing angle of a robot R_p viewed from another robot R_k be β_{kp}. A necessary condition for collision is*

$$\dot{\beta_{kp}} = 0. \tag{6.1}$$

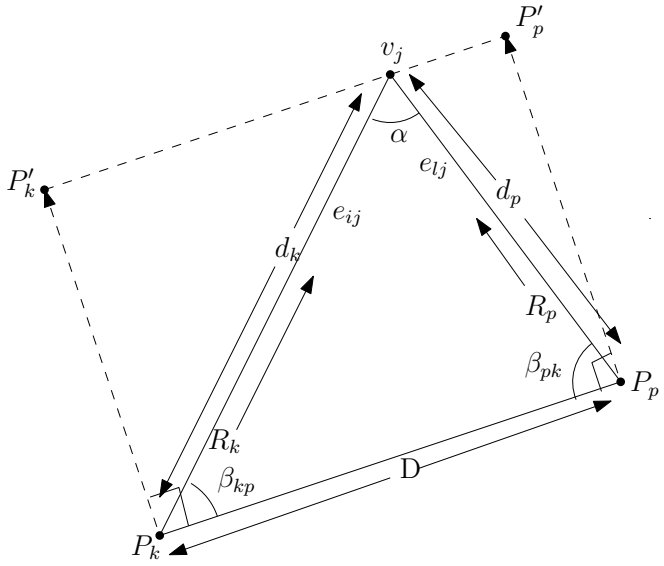


Figure 6.1: Robot collision detection

Proof Let a robot R_k traversing an edge e_{ij} with speed S_k be d_k distance away from a vertex v_j at a particular instant as shown in Fig. 6.1. Let another robot R_p be approaching vertex v_j from an edge e_{lj} with speed S_p and the angle between edges e_{ij} and e_{lj} at vertex v_j be $\alpha = |\theta_{ji} - \theta_{jl}|$. Let the instantaneous position of robot R_p be point P_p and the distance of point P_p from vertex v_j be d_p. Let the bearing of instantaneous position of robot R_p as viewed by the robot R_k be β_{kp} and that of robot R_k as viewed by the robot R_p be β_{pk}.

From $\triangle P_k v_j P_p$, we have

$$\alpha + \beta_{kp} + \beta_{pk} = \pi, \tag{6.2}$$

or,

$$\dot{\beta_{kp}} = -\dot{\beta_{pk}}, \tag{6.3}$$

as the angle between two edges, α is fixed. We also have

$$d_k \sin \beta_{kp} = d_p \sin \beta_{pk}. \tag{6.4}$$

Time derivative of (6.4) gives

$$\dot{d_k} \sin \beta_{kp} + d_k \cos \beta_{kp} \dot{\beta_{kp}} = \dot{d_p} \sin \beta_{pk} + d_p \cos \beta_{pk} \dot{\beta_{pk}}. \tag{6.5}$$

From (6.3) and (6.5), we get

$$\dot{\beta_{kp}}(d_k \cos \beta_{kp} + d_p \cos \beta_{pk}) = \dot{d_p} \sin \beta_{pk} - \dot{d_k} \sin \beta_{kp}. \tag{6.6}$$

Two robots traversing different edges collide with each other *if and only if* times taken by each robot to reach a vertex from their instantaneous positions are equal. The projections of speeds

$S_k = \dot{d_k}$ and $S_p = \dot{d_p}$ along the orthogonal direction of line joining points P_kP_p are $\dot{d_k}\sin\beta_{kp}$ and $\dot{d_p}\sin\beta_{pk}$ respectively (the edges intersect only at a vertex). In particular, the robots collide at the common vertex if and only if times taken to traverse $d_k\sin\beta_{kp}$ and $d_p\sin\beta_{pk}$ are equal. Therefore, the robots collide, if

$$\dot{d_k}\sin\beta_{kp} = \dot{d_p}\sin\beta_{pk}, \tag{6.7}$$

and (6.6) renders

$$d_k\cos\beta_{kp} + d_p\cos\beta_{pk} = 0, \tag{6.8}$$

$$\text{or } \dot{\beta_{kp}} = 0. \tag{6.9}$$

But, (6.4) and (6.8) gives

$$\frac{d_p}{\sin\beta_{kp}}\sin\beta_{pk}\cos\beta_{kp} + d_p\cos\beta_{pk} = 0. \tag{6.10}$$

Since $d_p \neq 0$, (6.10) gives $\beta_{kp} + \beta_{pk}$ as integer multiple of π. But, $\beta_{kp} + \beta_{pk} = \pi - \alpha$ from (6.2) which is possible only when $\alpha = 0$ or the two edges merge. Therefore, (6.9) is the only condition for collision of two robots traversing different edges and approaching a vertex. ■

It is worth noting that the collision while traversing a common edge ($\beta_{kp} = 0$) is also detected using (6.1).

6.2 Collision Avoidance Strategy

In any multi robot system, collision avoidance plays a crucial role to ensure efficient task completion. Once a collision is detected (using Lemma 6.1), avoidance measures should be taken to prevent unwanted interruptions in task execution due to robot failures. We propose a collision avoidance technique in which each edge in the graph is associated with two lanes as shown in Fig. 6.2.

Once a robot detects a collision with any other robot(s) observed within its field of view, the speed of the robot is reduced to zero or by a predefined value to avoid the collision. For the case when there are multiple robots moving in the same direction along the same edge, the speeds of all these robots except the one moving ahead are reduced to zero. In all the other cases, the speeds of the robots are reduced by a predefined value to avoid collisions. Robots move with the reduced speeds until they are in collision course with some other robots, and once there is

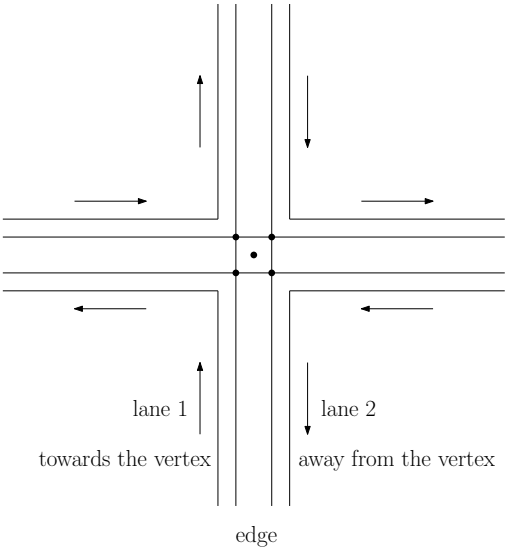


Figure 6.2: Edges with lanes

no possibility of collision detected, their speeds are increased. Here, the collision avoidance is achieved by using the bearing information of the robots only. This method can also be extended to environments with vertices having 4 or more edges. Whenever the minimum bearing angle of the observed robot(s) is zero or 2π, the speed has to be set to zero else reduced by a constant predefined value. Next section presents the analytical results on collision detection and speed reduction for collision avoidance.

6.3 Analytical Results

Theorem 6.2 proposes reduction in speeds of the robots for which the collision is detected.

Theorem 6.2 *Let S be a fixed initial speed of each robot and $\epsilon_k \in \mathbb{R}$ be a fixed positive speed change term associated with each robot R_k. Let the speed S_k of a robot R_k be reduced using* (6.11)*, when the collision is detected with the robot R_p using* (6.1) *($\dot{\beta}_{kp} = 0$),*

$$S_k = \begin{cases} 0 & \text{if } \beta_{kp} = 0, \\ S_k - \epsilon_k S & \text{otherwise}. \end{cases} \tag{6.11}$$

The change in speed S_k with distinct ϵ_k for $k = 1, \ldots, K$ satisfying

$$\epsilon_1 < (K-1)\epsilon_1 < \epsilon_2 < (K-1)\epsilon_2 < \epsilon_3 < \cdots < (K-1)\epsilon_{K-1} < \epsilon_K < (K-1)\epsilon_K < 1, \tag{6.12}$$

ensures inter-robot collision avoidance.

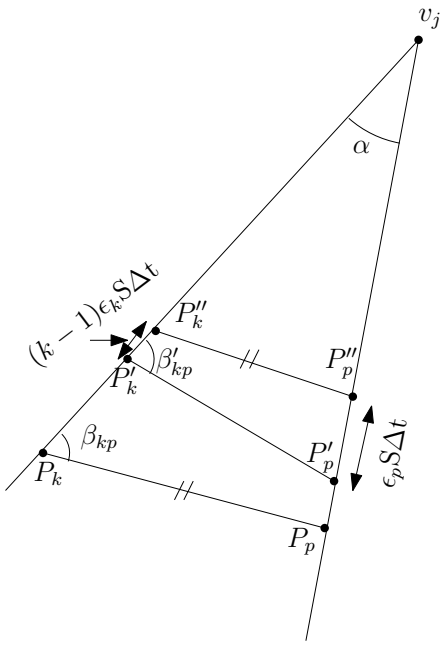


Figure 6.3: Collision avoidance using independent decisions

Proof Let the collision of the robot R_k moving with speed S_k be detected with the robot R_p moving with speed S_p. When $\beta_{kp} = 0$, the robot R_k follows the robot R_p on a common lane and the collision is avoided by stopping the robot R_k ($S_k = 0$).

When $\beta_{kp} \neq 0$, the robots R_k and R_p follow different lanes which meet at a common meeting point v_j. Let the positions of the robots R_k and R_p be P_k and P_p respectively at the instant collision is detected between the robots R_k and R_p ($\dot{\beta}_{kp} = 0$) as shown in Fig. 6.3. In particular, the robots R_k and R_p measures β_{kp} and β_{pk} respectively at each instant. The collision is detected by the robot R_k, if there is no change in the consecutive measurements of β_{kp}. Similarly, the collision is detected by the robot R_p, if there is no change in the consecutive measurement of β_{pk}. Let the positions of the robots be P_k'' and P_p'' after time interval Δt, if the robots moved with the same speeds S_k and S_p respectively. But, the robots reduce their speeds according to (6.11). Let the robots reach at the points P_k' and P_p' respectively with new speeds S_k' and S_p' in time interval Δt. Since $S_k' < S_k$ and $S_p' < S_p$, we have

$$P_kP_k' < P_kP_k'' \text{ and } P_pP_p' < P_pP_p''.$$

The robots R_k and R_p reduce their speeds by $\epsilon_k S$ and $\epsilon_p S$ using (6.11). Without loss of generality, we consider $k < p$. From (6.12), we have $\epsilon_k < \epsilon_p$ and

$$S - \epsilon_k S > S - (K-1)\epsilon_k S > S - \epsilon_p S > S - (K-1)\epsilon_p S. \tag{6.13}$$

Since a robot can have collision possibility with maximum $(K-1)$ robots, we have

$$\epsilon_k S\Delta t \leq P'_k P''_k \leq (K-1)\epsilon_k S\Delta t, \text{ and} \tag{6.14}$$

$$\epsilon_p S\Delta t \leq P'_p P''_p \leq (K-1)\epsilon_p S\Delta t. \tag{6.15}$$

From (6.13), (6.14) and (6.15), we have

$$P'_k P''_k < P'_p P''_p, \text{ or}$$

$$P_k P'_k > P_p P'_p. \tag{6.16}$$

In particular, (6.16) is ensured even if the robots R_k and R_p change their speeds later due to collision detection with other robots. Therefore, line joining instantaneous positions of the robots R_k and R_p rotates in the direction of increasing β_{kp}. This implies that the robot R_k always reaches the common vertex v_j earlier than the robot R_p and the collision is avoided between the pair of robots R_k and R_p till they cross the common vertex v_j. ■

Next, we present the increments in speed S_k required at different occurrences to accommodate multiple collisions at multiple common vertices and edges. The speed S_k is increased using (6.17), when the robot R_p goes out of view provided speed reduction for avoiding collision with the robot R_p is applicable [1]

$$S_k = \begin{cases} S & \text{if } S_k = 0, \\ S_k + \epsilon_k S & \text{otherwise,} \end{cases} \tag{6.17}$$

and the speed S_k is modified back to its initial speed S, or

$$S_k = S, \tag{6.18}$$

when it crosses the common vertex.

Remark 6.3 *Changing speed S_k using* (6.11)*,* (6.17) *and* (6.18) *ensure that the maximum reduction in speed S_k using ϵ_k is $(K-1)\epsilon_k S$ and maximum speed of any robot is S.*

We investigate changes of speed S_k using (6.11), (6.17) and (6.18) under following possible occurrences of encountering two robots with each other:

1 A robot traversing in front of another with the same direction of traversal as shown in Fig. 6.4(a).

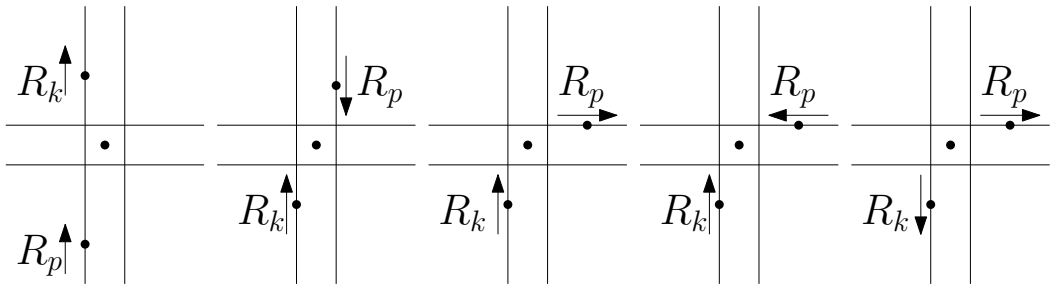
[1] If the speed S_k has a reduction component $\epsilon_k S$ due to the robot R_p, then the increment is applicable.

2 A robot traversing in front of another with the opposite direction of traversal as shown in Fig. 6.4(b)

3 both the robots traversing different edges connected to a common vertex; one robot is traversing towards the common vertex whereas other robots is traversing away from the common vertex as shown in Fig. 6.4(c).

4 both the robots traversing different edges connected to a common vertex; traversal direction of both the robots is towards the common vertex as shown in Fig. 6.4(d).

5 both the robots traversing different edges connected to a common vertex; traversal direction of both the robots is away from the common vertex as shown in Fig. 6.4(e).



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5

Figure 6.4: All possibilities of encountering between two robots

We show that the the collision is avoided under all the possibilities of encountering another robot, when speed S_k of the robot R_k is changed using either (6.11), (6.17) or (6.18):

- *Case 1:* Two robots traversing same edge in the same direction satisfies

$$\beta_{kp} = 0 \text{ and } \dot{\beta_{kp}} = 0.$$

Speed reduction using (6.11) is applied, since $\dot{\beta}_{kp} = 0$ and speed of the observer robot R_k is changed to zero. When the robot moving in front of the observer robot is out of view, the observer robot changes its speed using (6.17) and it is modified back to S. Except the robot, which is moving ahead on a lane, all the other robots along the same lane do not move until the lane is cleared for traversal.

- *Cases 2 and 3:* When two robots are traversing the opposite lane of an edge and when the robots traverse different edges with one robot coming toward the common vertex and

another going away from a common vertex, the condition $\dot{\beta_{kp}} \neq 0$ is strictly satisfied. Since the robots are following opposite lanes or moving apart, they would never collide with each other. Therefore, speeds of the robots are unchanged (S_k).

- *Case 4:* A critical collision possibility occurs when two or more robots approach the same vertex from different edges. The robots traversing different edges ($\beta_{kp} \neq 0$) and approaching a common vertex collide each other if the condition $\dot{\beta_{kp}} = 0$ (from Lemma 6.1). Theorem 6.2 shows that the proposed strategy of reducing the speeds of the robots avoids collision.

  Modifying speed of the robot to S, when it crosses the common vertex allows it to accommodate collision avoidance at a different common vertex. Since the reduction of speed ($\epsilon_k S$) is not applicable after the robot R_k crossed the common vertex, its speed ($S_k = S$) is *not* increased using (6.17), even if the robot R_p goes out of view.

- *Case 5:* Two robots R_k and R_p traversing different edges, but moving away from the common vertex can not collide with each other at the common vertex. However, $\dot{\beta_{kp}} = \dot{\beta_{pk}} = 0$ may satisfy and the robots change their speeds by $S_k - \epsilon_k S$ and $S_p - \epsilon_p S$ respectively (since a robot can satisfy (6.1) with maximum $(K-1)$ robots and there exists only one edge between any two vertices, $S_k \neq 0$ and $S_p \neq 0$). The changes in their speeds are applicable till the robots R_k and R_p are in each other's field of view. It is worth noting that this possibility does not result in collision, but the robots traverse with changed speeds. However, these changes in speeds are valid till the robots are visible to each other. Since the robots are moving apart, they would go out from each other's field of view and respective speeds are changed using (6.17). In particular, the speeds are changed to $S_k + \epsilon_k S$ and $S_p + \epsilon_p S$ respectively.

Corollary 6.4 *Let the collisions of the robot R_k be detected simultaneously with multiple robots $R_{p_i} \forall i = 1, \ldots, n \mid n < K$. Avoiding collision with one robot by reducing the speed once using* (6.11)*, the collision is avoided with each robot R_{p_i}.*

Proof When simultaneous collisions are detected with the robots R_{p_i}, we have $\dot{\beta}_{kp_i} = 0$. Let the robot R_k changes its speed considering only one robot R_{p_1} in collision using (6.11). Similarly, each robot R_{p_i} reduces its speed once by $\epsilon_{p_i} S$, because at least $\dot{\beta}_{p_i k} = 0$. With the reduced speeds of the robots R_k and R_{p_i}, the collision is avoided when each $\beta_{kp_i} \neq 0$, because each

robot reduced its speed once and speed change between any pair of robots is according to the Theorem 6.2.

The collision is avoided even if $S_k = 0$. It is worth noting that $S_k = S_{p_i} = 0$ is not a deadlock condition because if $\beta_{kp_i} = 0$, then $\beta_{p_ik} \neq 0$ and vice versa.

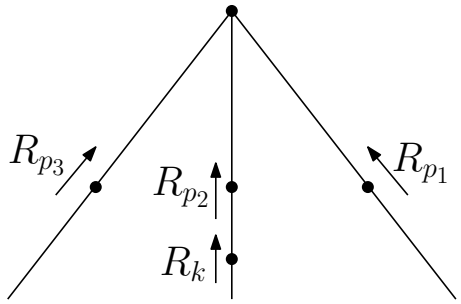


Figure 6.5: Simultaneous collision detections with multiple robots

The robot R_k avoids collision with the robot R_{p_2} if $\beta_{kp_2} \neq 0$. But, if $\beta_{kp_2} = 0$ as shown in Fig. 6.5, then we have $\dot{\beta}_{kp_2} = 0$ even after reduction in speed. The next instance, $S_k = 0$ and the collision is avoided with all the robots with which collisions are detected. ■

We assume that the dynamic obstacles (humans, etc.) move along any non-maneuverable trajectories. The algorithm guarantees collision avoidance with these dynamic obstacles. The bearing information of each of these dynamic obstacles is obtained and verified against the collision condition continuously. If the collision condition is satisfied, speed reduction will be initialized to avoid collisions.

The next section presents the algorithm for the proposed speed reduction based approach for collision avoidance.

6.4 Algorithm for Collision Avoidance

Algorithm 8 demonstrates the proposed speed reduction based multi robot collision avoidance. In algorithm 8, the bearing angle of a robot R_i viewed from another robot R_k is denoted by β_{ki}, S is the initial speed of each robot. Whenever a collision is detected by a robot R_k, the speed is reduced by a predefined speed change term ϵ_k associated with it. The reduced speed of the robot R_k to avoid a collision is represented by S_k. As soon as a robot R_k observes some other robot traversing in the same direction in front of it, it stops (speed reduces to zero). Once the

other robot is out of view, the speed is incremented. The function *robot_vertex_crossed* checks whether a robot has crossed a vertex or on its way towards it. The function *robot_exiting_fov* checks if a robot which was inside the field of view of an observing robot is still inside the FoV or not.

Algorithm 8 *Collision Avoidance*

Input: Bearing angle β_{ki}.

Output: Modified speed S_k to avoid collision.

Functions: *robot_vertex_crossed*(), *robot_exiting_fov*().

1: **if** $\dot{\beta_{ki}} = 0$ **then**
2: **if** $\beta_{ki} = 0$ **then**
3: $S_k = 0$.
4: **else**
5: $S_k = S_k - \epsilon_k S$.
6: **else**
7: **if** $S_k = 0$ || *robot_vertex_crossed*(R_k) = *yes* **then**
8: S_k = S.
9: **else if** *robot_exiting_fov*(R_k) = *yes* **then**
10: $S_k = S_k + \epsilon_k S$.

6.5 Conclusion

This chapter discussed the proposed collision avoidance strategy using minimal sensing. Robots move along the edges of the graph (paths in the environment) along the lanes. Since, the approach is decentralized, there is no exchange of messages possible to indicate the collision occurrence. Therefore, a collision condition to detect the possibility of collision for robots reaching the same vertex or traversing the same edge is obtained. Whenever a collision is detected, the speeds of the robots in collision course are changed by a fixed parameter to avoid collisions. The possible collision occurrences for the robots are also investigated and analyzed.

The results of hardware implementation of the proposed exploration strategy along with collision avoidance technique is presented in next chapter. The robustness of the proposed IMRDFS strategy to robot failures is also verified experimentally.

Chapter 7

Hardware Implementation

This chapter describes the experiments conducted to validate the proposed IMRDFS strategy with collision avoidance. The experiments have been performed using a team of differential drive robots. The arena for the experiments have been designed to model various graph/tree structures. The experiments results for the implementation of the proposed strategies and robustness validation have been explained in detail subsequently.

7.1 Hardware Setup

The experiments are conducted with a team of Spark V robots on different arenas to validate the algorithm. Spark V (as shown in Fig. 7.1) is a low cost differential drive robot. It has three analog white line sensors, three analog IR proximity sensors, battery voltage sensor and position encoders. The diameter and height of the robot are 15 cms and 7 cms respectively. The robot is equipped with two DC motors in differential drive configuration along with a caster wheel positioned on the back for support. Motors are controlled by L293D dual motor driver which can provide up to 600 mA of current to each motor. Maximum speed of the robot is 20 cm/sec.

The Xbee transceivers are interfaced to each robot to establish the communication framework among the robots and the vertices. The IR proximity sensors obtain the obstacle information around the path within proximity of 10 cms and used to identify arrival of a vertex. The data obtained from the white line sensors controls the robot to follow a free path using line following technique. The controller of a robot is implemented on the Arduino-Uno board to ease the Xbee transceiver interfacing. The robots store the updated incidence matrix in the 1 KB on board EEPROM.

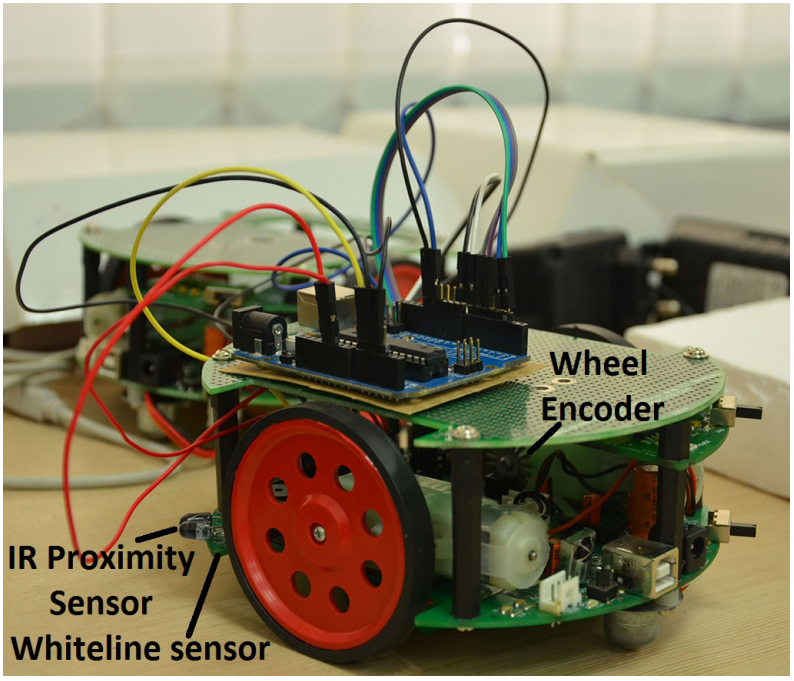


Figure 7.1: Spark V Robot

The experiment is performed by retrieving/updating the information on vertices from a central station through Xbee communication to emulate beacon dropping capability of a robot to account for the delay in Xbee transmission. A delay (less than 0.5 sec) is introduced by the Xbee communication at each vertex update. Each robot stops at each vertex for a very small time (for 1 sec) and selects the next edge to be traversed based on the updated partial map. The central station keeps separate information for each vertex. The black strips on the floor indicate the edges of the graph and facilitate the straight-line movement for the robots. Two black strips are marked each corresponding to the direction of following left side obstacle along each edge in the graph.

Each robot identifies a vertex using the IR proximity sensor by detecting the absence of obstacle to the left of the robot for a predefined distance. When a robot reaches a vertex, the robot updates the incoming edge information and it's incidence matrix to the central station via Xbee. The central station then merges the incidence matrices and updates the partial map. From the updated vertex incidence matrix, the incidence angle corresponding to the right most edge is selected for next traversal. Once an edge is selected, the robot sends the selected edge information to the central station to update the incidence matrix of corresponding vertex.

7.2 Experimental Arena

Experiments are conducted on two arenas: one represents a graph with 8 vertices and 9 edges and other represents a tree with 13 vertices and 12 edges. Each edge in the graph is associated with two lanes and the robots always follow the left obstacle to move in forward direction. In Figs. 7.2 and Fig. 7.3, robots and vertices are shown by R_i and v_i respectively.

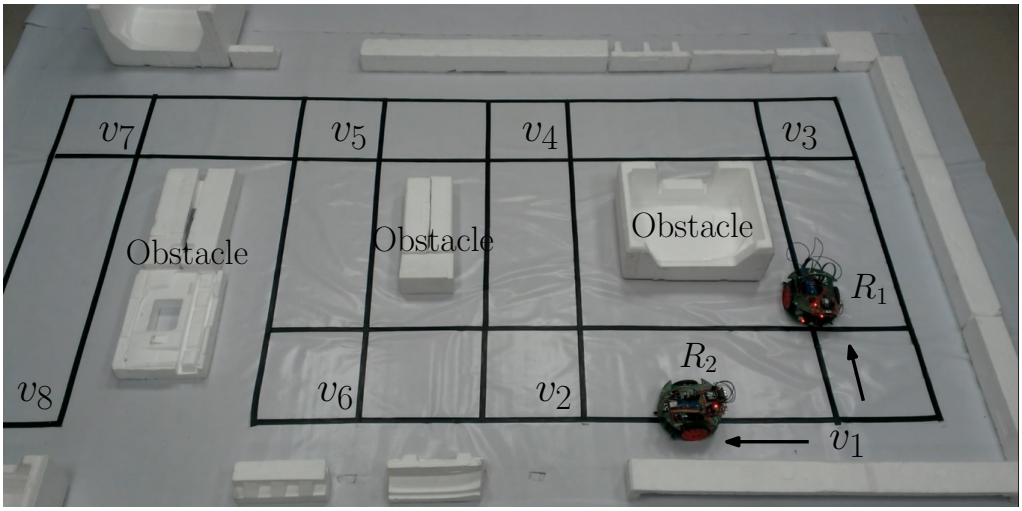


Figure 7.2: Experiment on Arena 1 using two robots

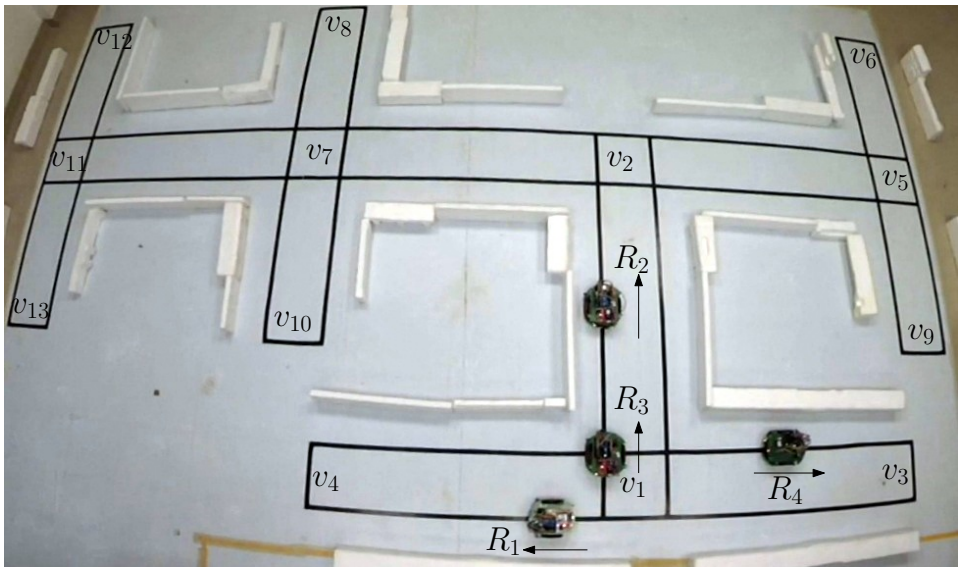


Figure 7.3: Experiment on Arena 2 using four robots

The dimensions of arena 1 (shown in Fig. 7.2) are 2.7 m x 1.35 m with minimum and maximum edge lengths to be 50 cms and 20 cms respectively. Arena 2 (shown in Fig. 7.3) measures 4.2 m x 2.4 m with minimum and maximum edge lengths to be 100 cms and 65 cms

respectively. The details of these the experiments are explained in the subsequent paragraphs. The mount boards placed on either sides of the lanes (paths) represent the obstacles in the environment. In both the experiments, robots identify a vertex if there is no obstacle detected on the left side of the robot for a distance of 11 cms. Initial speed of each robot is $5\ \mathrm{cm/sec}$.

7.3 Experimental Results

Arena 1 is explored using two robots and Arena 2 is explored using four robots. Since, the white line sensors available on these robots are not very accurate, some times robots go off the lane while traversing the edges. So, the robot is picked and placed on its lane whenever it went off a lane. This manual placement did not disturb the implementation because decisions are taken when a robot reaches a vertex. Videos of these experiments are available at `http://www.sc.iitb.ac.in/~leena/multi_robot_exploration.html`.

The experiments also demonstrate the collision avoidance using the proposed speed changing approach. Collision avoidance between the robots is achieved using the edge traversal information of the robots at each instance and the tree/graph topology considered. The source vertex and the selected incidence angle at that vertex determines the edge and the direction of traversal for a robot. Thus, at a given instance the speed of the robots is modified using the edge and traversal direction. If any of the robots found to be traveling edges along the same direction and within the LoS of other robots (obtained from the graph topology), all the robots except the robot moving ahead are made to stop. In all the other possible cases of collisions, speed of the robots is reduced based on the predefined ϵ_k.

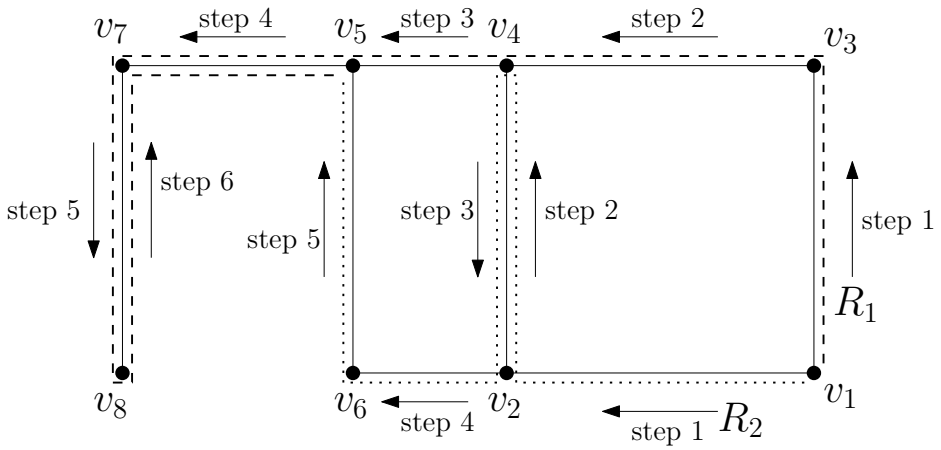


Figure 7.4: Robot path traversals for the experiment on Arena 1 using two robots

Fig. 7.4 illustrates the paths of traversal for each of the two robots for the experiment conducted on arena 1. The edge traversals of robots R_1 and R_2 are shown using the dashed and dotted lines respectively. Robots start the exploration from vertex v_1 and finishes at vertex v_5. When the robots are approaching vertex v_4, the speeds of the robots R_1 and R_2 are changed by considering $\epsilon_1 = 0.05$ and $\epsilon_2 = 0.25$ respectively to avoid the collision (observed at $00 : 18$ sec in the video). The exploration is completed in 7 edge traversals.

Figs. 7.5, 7.6, 7.7 and 7.8 illustrate the paths of traversal for each of the robots for the experiment carried out on arena 2. Robots start the exploration from vertex v_1 and completes at vertex v_7. While exploring Arena 2, the speeds of the robots R_3 and R_4 are changed by considering $\epsilon_3 = 0.1$ and $\epsilon_4 = 0.5$ respectively to avoid the collision (observed at $01 : 45$ sec in the video), when the robots are approaching vertex v_3. At various other instances during the exploration, the speed of the observer robots traversing the same edge is changed to zero (observer robots stopped at 00.08, 00.46, 01.27, 01.40, 01.53, 02.04, 02.27 and 02.45 sec in the video) to avoid collisions. The exploration is completed in 10 edge traversals. It is worth noting that selecting a maximum distance between two robots while stopping the observer robot would result in faster exploration.

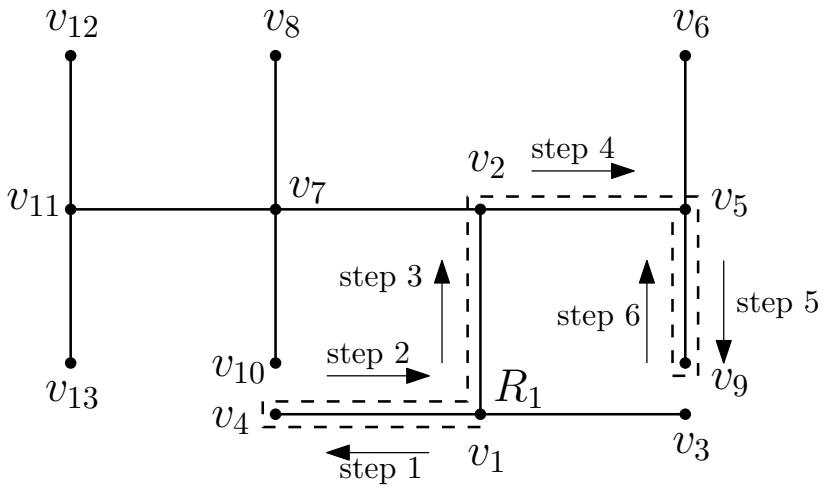


Figure 7.5: Path traversals for Robot 1 on Arena 2

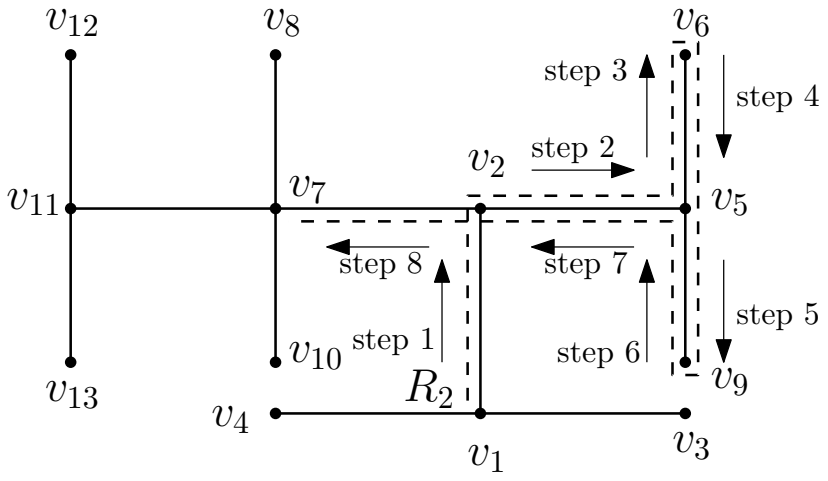


Figure 7.6: Path traversals for Robot 2 on Arena 2

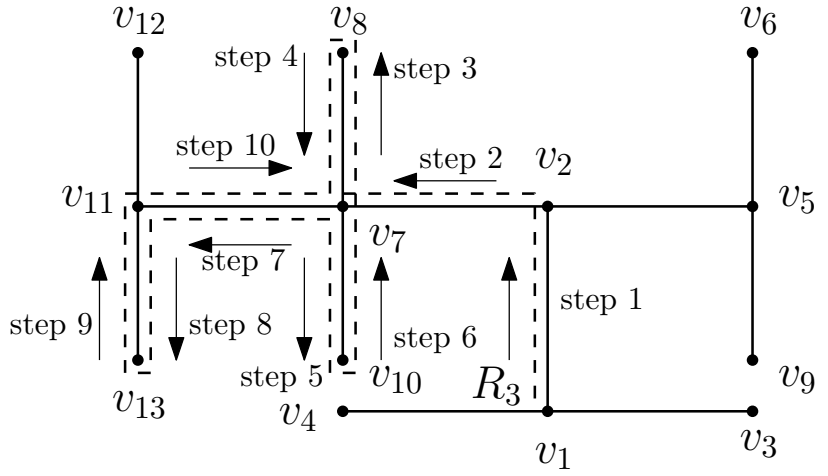


Figure 7.7: Path traversals for Robot 3 on Arena 2

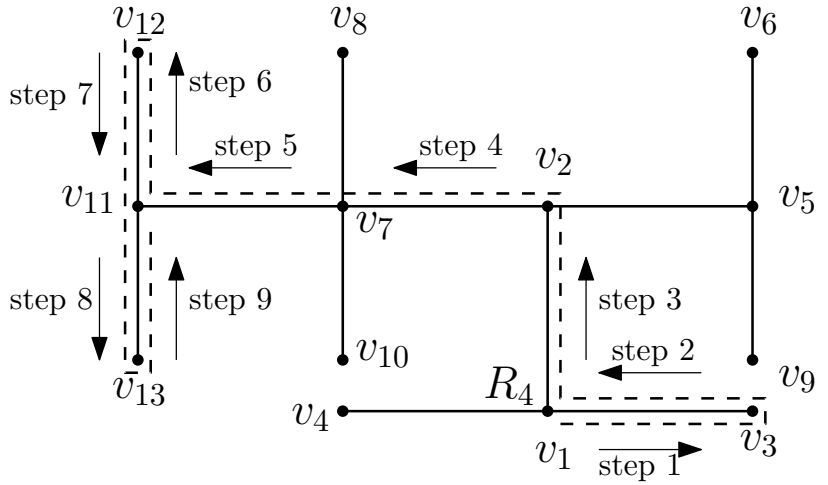


Figure 7.8: Path traversals for Robot 4 on Arena 2

7.4 Robustness to Robot Failures

Robustness of a multi robot strategy is the key to achieve efficient completion of a given task. For a multi robot exploration strategy to guarantee finite time completion, the strategy should be able to handle unanticipated robot failures. In this case, the strategy should be able to detect the faulty robot(s) and take further measures to complete the task with the available fleet. Detection of such a fault becomes challenging if the communication framework is decentralized. For this work, fault detection is assumed to validate the robustness of the proposed IMRDFS strategy.

Experiments are performed to show the robustness of the proposed strategies for robot failures. The experimental demonstration of the robustness of the proposed strategies is illustrated using a team of spark V robots exploring an environment with tree structure. The robot to beacon communication is achieved using a pair of Xbees, one interfaced with the robot and the other with the beacon, as described in the earlier experiment. To check the robustness of the proposed strategy, one of the robots is stopped to emulate the faulty robot and removed from the arena.

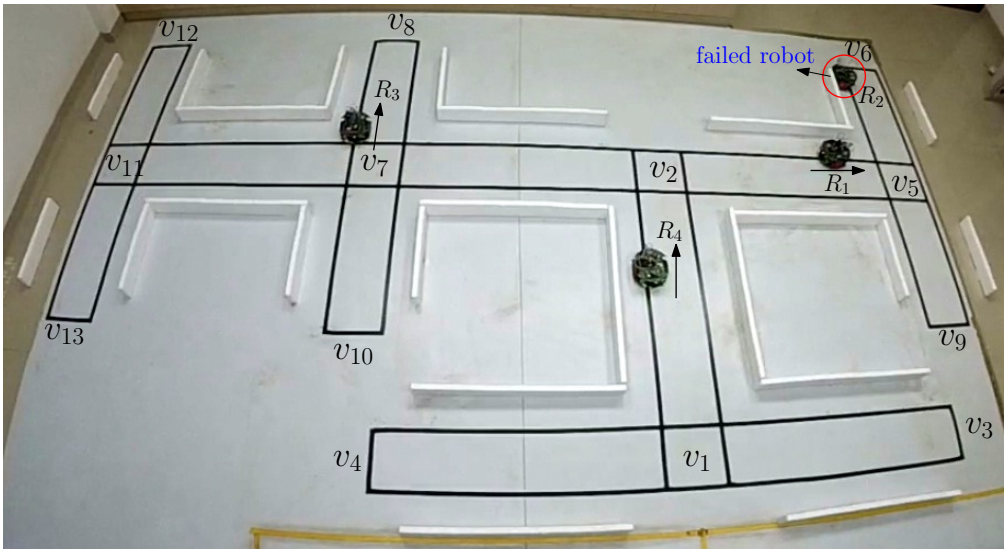


Figure 7.9: Robot failure while exploring the environment

The initial arrangement of the robots is as shown in Fig. 7.3. After two edge traversals, the robot R_2 is stopped along the edge e_{56} (faulty robot) as shown in Fig. 7.9 and removed from the arena so that it does not hinder the path of other robots during the exploration. At this instance, the edge e_{56} is recorded as an out edge at vertex v_5. As there is no update from the robot R_2 about the edge e_{56}, the status of this edge does not get updated at vertex v_5. So, if a robot R_1 reaches vertex v_5 with no unexplored edges, the edge e_{56} is an out edge and is selected for the

next traversal. The exploration is advanced with the remaining robots and is completed in 10 steps. So, the exploration is complete in finite time (as the edge traversal time is finite for any edge) despite the failure of the robot.

7.5 Conclusion

The experimental validation of the proposed IMRDFS strategy with collision avoidance is carried out using a team of Spark V robots exploring the environments. The tree and graph explorations have been validated using two different arenas. In both the scenarios, the exploration is complete in finite steps and the performance is as desired. Robot collision avoidance using speed reduction is also verified using the hardware implementation. The robustness of the proposed strategy is also validated experimentally.

Next chapter presents a summary of the thesis with directions to future research.

Chapter 8

Conclusion and Future Work

This work analyzes multi robot exploration and mapping strategy to completely explore an unknown environment. A modification in existing multi robot DFS [1] for decentralized decision technique (IMRDFS strategy) is proposed and new condition for declaring completion of exploration is obtained. The new condition is capable of declaring the completion without reaching the root vertex (location from where the exploration started). The introduction of incidence matrix as data structure helps in efficient information exchange. This information exchange using incidence matrix is the partially completed map and unexplored edges with various status, thus reducing redundancy in exploration. The IMRDFS strategy also guarantees completion of exploration in finite number of edge traversals. While exploring an environment, robots have the possibility of colliding with each other. To address this problem, an approach to multi robot collision avoidance is proposed. The avoidance strategy is based on changing the speed of a robot without knowing the speeds of any other robot. The condition for detecting the collision possibility using minimal sensing is also developed. The IMRDFS strategy is analyzed on both trees and graphs. It is shown that this strategy performs better than the single and multi robot DFS [1] strategies. The performance of IMRDFS strategy along with collision avoidance is demonstrated experimentally using a team of mobile robots. The robustness of the IMRDFS strategy to robot failures is also validated for an environment with tree structure.

Another major contribution of this work is the development of a generalized framework for multi robot exploration and mapping using the modified incidence matrix for any initial placement of the robots, search strategy and information exchange method. Any new/existing exploration strategy can be developed/evaluated using this proposed unified framework. Further, a method to determine the optimal robot count (fleet size) is proposed and compared with

the existing techniques. Various search strategies are simulated and a comparison of their performance on illustrative graph topologies are presented to show the effectiveness of generalized framework.

8.1 Future Work

The thesis presents a decentralized multi robot exploration and mapping strategy; *improved multi robot depth first search (IMRDFS)*. The strategy guarantees finite time completion, however, the work can be extended to determine the bounds on completion time and compute the minimum number of beacons required to complete the exploration. A generalized framework for multi robot exploration and mapping has also been presented which can be used for any initial position of the robots, communication scheme and search strategy. The possible directions for future work could be the identification of the optimal initial positions for the robots and obtaining an analytical expression for the negative exponential relationship between number of robots and completion time relating the optimal fleet size, graph topology and exploration strategy.

The proposed work on multi robot collision avoidance can be extended to address curvilinear paths. A piece-wise linear approximation to a curved edge between two vertices results in multiple virtual vertices with linear edges. The collision condition in proposed work is obtained for the linear edges. Since the collision condition may be true when any robot is traversing an edge connecting two virtual vertices, revisiting the proposed conditions for speed increment would address the collision when the robots cross virtual vertices. It is worth noting that the proposed collision condition is valid when the robots are traversing the edge connecting vertex and its adjacent virtual vertices. Similarly, if the robot changes its linear path to avoid a dynamic obstacle for some time, the collision condition may be satisfied for some time interval. Checking the collision condition at every time instant and revisiting the speed increment condition would address the inter-robot collision avoidance in the presence of dynamic obstacles.

References

[1] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, 2011.

[2] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[3] R. Fleischer and G. Trippen, "Exploring an unknown graph efficiently," in *Algorithms–ESA*. Springer, 2005, pp. 11–22.

[4] H. Wang, M. Jenkin, and P. Dymond, "Enhancing exploration in graph-like worlds," in *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*, 2008, pp. 53–60.

[5] M. A. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661–675, 2007.

[6] D. Dereniowski, Y. Disser, A. Kosowski, D. Pająk, and P. Uznański, "Fast collaborative graph exploration," *Information and Computation*, vol. 243, pp. 37 – 49, 2015.

[7] F. Cabrera-Mora and J. Xiao, "Fleet size of multi-robot systems for exploration of structured environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 370–375.

[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[9] E. J. Rodriguez-Seda, "Decentralized trajectory tracking with collision avoidance control for teams of unmanned vehicles with constant speed," in *Proceedings of the American Control Conference (ACC)*, 2014, pp. 1216–1223.

[10] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC)*, 2003, pp. 539–543.

[11] A. Souliman, A. Joukhadar, H. Alturbeh, and J. F. Whidborne, "Real time control of multi-agent mobile robots with intelligent collision avoidance system," in *Proceedings of the Science and Information Conference (SAI)*, 2013, pp. 93–98.

[12] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859–865, 1991.

[13] F. Cabrera-Mora and J. Xiao, "A flooding algorithm for multirobot exploration," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 3, pp. 850–863, 2012.

[14] C. Gong, S. Tully, G. Kantor, and H. Choset, "Multi-agent deterministic graph mapping via robot rendezvous," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1278–1283.

[15] M. Meghjani and G. Dudek, "Multi-robot exploration and rendezvous on graphs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5270–5276.

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

[17] A. Franchi, L. Freda, L. Marchionni, G. Oriolo, and M. Vendittelli, "Decentralized cooperative exploration: Implementation and experiments," in *Proceedings of the Intelligent Autonomous Systems (IAS)*, vol. 10, 2008, pp. 348–355.

[18] F. F. Carvalho, R. C. Cavalcante, M. A. Vieira, L. Chaimowicz, and M. F. Campos, "A multi-robot exploration approach based on distributed graph coloring," in *Proceedings of the Latin American Robotics Symposium and Competition (LARS/LARC)*, 2013, pp. 142–147.

[19] J. Freeman, "Parallel algorithms for depth-first search," Technical Report, MS-CIS-91–71, University of Pennsylvania, 1991.

[20] S. Hong, T. Oguntebi, and K. Olukotun, "Efficient parallel graph exploration on multi-core CPU and GPU," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2011, pp. 78–88.

[21] S. El Ghoul, A. S. Hussein, M. S. Abdel-Wahab, U. Witkowski, and U. Rückert, "A modified multiple depth first search algorithm for grid mapping using mini-robots Khepera," *Journal of Computing Science and Engineering (JCSE)*, vol. 2, no. 4, pp. 321–338, 2008.

[22] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

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

[24] K. Senthilkumar and K. Bharadwaj, "Multi-robot exploration and terrain coverage in an unknown environment," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 123–132, 2012.

[25] X. Li, G. Fletcher, A. Nayak, and I. Stojmenovic, "Placing sensors for area coverage in a complex environment by a team of robots," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, p. 3, 2014.

[26] X. Deng, T. Kameda, and C. Papadimitriou, "How to learn an unknown environment," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, 1991, pp. 298–303.

[27] E. Ferranti, N. Trigoni, and M. Levene, "Brick& Mortar: an on-line multi-agent exploration algorithm," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 761–767.

[28] M. Andries and F. Charpillet, "Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendezvous using ant algorithms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5571–5578.

[29] R. Pandey, A. K. Singh, and K. M. Krishna, "Multi-robot exploration with communication requirement to a moving base station," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, 2012, pp. 823–828.

[30] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.

[31] Y. Pei, M. W. Mutka, and N. Xi, "Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 5460–5465.

[32] L. Wu, M. Á. García García, D. Puig Valls, and A. Solé Ribalta, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no. 1, pp. 37–44, 2007.

[33] J. Kim, "Cooperative exploration and protection of a workspace assisted by information networks," *Annals of Mathematics and Artificial Intelligence*, vol. 70, no. 3, pp. 203–220, 2014.

[34] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart, "Distributed multirobot exploration, mapping, and task allocation," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 229–255, 2008.

[35] R. Vaughan, K. Støy, G. S. Sukhatme, and M. Mataric, "Go ahead, make my day: robot conflict resolution by aggressive competition," in *Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB)*, 2000.

[36] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009, pp. 177–187.

[37] D. V. Dimarogonas and K. J. Kyriakopoulos, "Formation control and collision avoidance for multi-agent systems and a connection between formation infeasibility and flocking behavior," in *Proceedings of the 44th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC)*, 2005, pp. 84–89.

[38] S. Kumar, T. P. Parekh, and K. M. Krishna, "A hierarchical multi robotic collision avoidance scheme through robot formations," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2010, pp. 306–311.

[39] P. Vrba, V. Marik, L. Preucil, M. Kulich, and D. Sislak, "Collision avoidance algorithms: Multi-agent approach," in *Holonic and Multi-Agent Systems for Manufacturing*. Springer, 2007, pp. 348–360.

[40] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, 2013, pp. 3944–3950.

[41] Z. Shijie and D. Guangren, "Collision avoidance in multi-agent formation keeping cooperative control systems," in *Proceedings of the 30th Chinese Control Conference (CCC)*, 2011, pp. 4758–4762.

[42] K. S. Hwang and M. Y. Ju, "Speed planning for a maneuvering motion," *Journal of Intelligent and Robotic Systems*, vol. 33, no. 1, pp. 25–44, 2002.

[43] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *Journal of Intelligent and Robotic Systems*, vol. 72, no. 2, pp. 147–165, 2013.

[44] Y. Mei, Y. H. Lu, Y. C. Hu, and C. S. G. Lee, "Determining the fleet size of mobile robots with energy constraints," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, 2004, pp. 1420–1425.

[45] D. Caballero-Parga, S. Esteva, and A. Figueras, "Fleet size of robots for rescue missions," in *Proceedings of the IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, vol. 1, 2010, pp. 1–4.

[46] Y. Higashikawa, N. Katoh, S. Langerman, and S. Tanigawa, "Online graph exploration algorithms for cycles and trees by multiple searchers," *Journal of Combinatorial Optimization*, vol. 28, no. 2, pp. 480–495, 2014.

[47] N. Bouraqadi, L. Fabresse, and A. Doniec, "On fleet size optimization for multi-robot frontier-based exploration," in *Proceedings of the 7th National Conference on Control Architectures of Robots*, 2012.

[48] Y. Mei, Y. H. Lu, Y. C. Hu, and C. S. G. Lee, "Reducing the number of mobile sensors for coverage tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 1426–1431.

[49] C. Godsil and G. F. Royle, *Algebraic Graph Theory*. Springer-Verlag New York, Inc., 2001, vol. 207.

[50] F. W. Lawvere, "Qualitative distinctions between some toposes of generalized graphs," *Contemporary mathematics*, vol. 92, pp. 261–299, 1989.

[51] S. C. Nagavarapu, L. Vachhani, and A. Sinha, "Multi-robot graph exploration and map building with collision avoidance: A decentralized approach," *Journal of Intelligent and Robotic Systems*, DOI:10.1007/s10846-015-0309-9.

[52] P. R. Halmos, *Finite dimensional vector spaces*. Springer Science & Business Media, 2012.

[53] R. Diestel, *Graph theory*, 3rd ed., ser. Graduate Texts in Mathematics. Springer, 2006.

[54] K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms*. John Wiley & Sons, 2011.

Publications

Papers in referred Journals:

- **Sarat Chandra Nagavarapu**, Leena Vachhani and Arpita Sinha, “Multi-robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach”, *Journal of Intelligent and Robotic Systems*, vol. 83, no. 3, pp. 503-523, 2016.

- **Sarat Chandra Nagavarapu**, Leena Vachhani, Arpita Sinha and Pranjal Vyas, “Generalized Framework for Multi Agent Graph Exploration” (Submitted to *IEEE Transactions on Cybernetics*).

Papers in Conference proceedings:

- **Sarat Chandra N.**, Leena Vachhani and Arpita Sinha, “A Decentralized Approach for Autonomous Multi-Robot Exploration and Map Building for Tree Structures," *In Proceedings of the 1st Indian Control Conference (ICC)*, pp. 274-279, IIT Madras, Chennai, January, 2015.

Acknowledgments

While completing this work I realized that, it is not only mine but of many! I want to acknowledge all those many!! God has given me the strength, patience and courage to sail across in spite of many constraints. I acknowledge that great source of power.

I would like to express my sincere gratitude to my supervisors, Profs. Leena Vachhani and Arpita Sinha. It is due to their guidance, patience, enthusiasm, and direction, which made the completion of this dissertation possible. It is my good fortune that I have been able to work with them. The members of my research progress committee, Profs. Hemendra Arya and Sukumar Srikant for their advice, support, time and effort expended to improve this work; I cannot thank them enough.

I am eternally indebted to my family for their immense encouragement and immeasurable support to pursue my career. I would also like to express my sincere thanks to my gurus, Janardhana Sir, Srinivas Yadav Sir, Manjula Ma'am, Prof. Durga Prasad and Prof. Srinivasa Prasanna for inspiring me to march towards my goal.

In addition I wish to thank my colleagues at the Embedded Control Systems Laboratory, in particular Pranjal, Sanket, Gaurav, Kumar S. Khot, Anupa Ma'am and Arun kumar GK for their support and encouragement. I want to thank my friends Jack, Asif, Jeya and Twinkle for their valuable companionship and moral support during the ups and downs of my PhD journey. During this time, I enjoyed the good company of friends at IITB such as: Shyam, Abhisek, Ramesh Sir, Manas Sir, Lal Priya Ma'am, Sangeeta Ma'am, Sunil Sir, Swaroop, Karthikeyan, Venkatesh GS, Aseem, Rayyan, Arun, Jerome, Sankar, Somesh, Samba, Prasad, Venky, Ratna, Tarak, Srinu, DP, Ajay, Ravi and Ganapathi. I am also grateful to the office staff of SysCon, especially Sangeeta Ma'am, Kanchan and Nagesh for being helpful to me in office related work.

Date: _____ Sarat Chandra Nagavarapu