J Intell Robot Syst (2016) 83:503–523
DOI 10.1007/s10846-015-0309-9

Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach

Sarat Chandra Nagavarapu · Leena Vachhani · Arpita Sinha

Received: 1 May 2015 / Accepted: 11 November 2015 / Published online: 8 December 2015

Abstract This paper proposes a decentralized multi-robot graph exploration approach in which each robot takes independent decision for efficient exploration avoiding inter-robot collision without direct communication between them. The information exchange between the robots is possible through the beacons available at visited vertices of the graph. The proposed decentralized technique guarantees completion of exploration of an unknown environment in finite number of edge traversals where graph structure of the environment is incrementally constructed. New condition for declaring completion of exploration is obtained. The paper also proposes a modification in incidence matrix so that it can be used as a data structure for information exchange. The modified incidence matrix after completion represents map of the environment. The proposed technique requires either lesser or equal number of edge traversals compared to the existing strategy for a tree exploration. A predefined constant speed change approach is proposed to address the inter-robot collision avoidance using local sensor on a robot. 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.

Keywords Graph exploration · Multi-robot · Collision avoidance · Incidence matrix · Mapping · Decentralized strategy

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.

L. Vachhani (✉) · S. C. Nagavarapu · A. Sinha
Systems and Control Engineering, Indian Institute of Technology Bombay, 400076, Mumbai, MH, India
e-mail: leena.vachhani@iitb.ac.in

S. C. Nagavarapu
e-mail: sarat@sc.iitb.ac.in

A. Sinha
e-mail: asinha@sc.iitb.ac.in

$\mathcal{E}_o^n(v_j)$	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	Proposed 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.

1 Introduction

Surveillance, search and rescue, demining, coverage, etc. tasks need to explore the entire environment. Map, a representation of the environment is incrementally built and used while exploring the environment. The exploration task is complete when it can be declared that no portion of the environment is left unexplored. A major challenge in an autonomous exploration task is to declare completion, which signifies that the entire environment is covered, particularly when the environment is explored using decentralized control of multiple robots. 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 [1].

Multiple robots exchange information in decentralized approaches by broadcasting in limited range [2] or placing book keeping devices (beacons) at important locations [3, 4]. Partially explored graph information is updated by each robot visiting a location with the beacon [3, 4]. A systematic strategy is required to complete the exploration. Breadth First Search(BFS) [5, 6], Depth First Search (DFS) [3] and Least Recently Visited (LRV) [7] strategies have been used for exploring graphs using multiple agents. The BFS strategy gradually increases the distance from root vertex and best suited for the applications where root vertex represents an important place. The LRV strategy [7] has been used with sensor network. The DFS strategy has been shown best for mapping an unknown environment [3], because the robot returns to the immediate parent for further exploration avoiding redundant edge traversals. The performance of a strategy also depends on the graph topology and method of exchanging information. This paper proposes a data structure for efficient information exchange and a modified Multi-Robot DFS (MR-DFS) strategy, which helps in achieving better performance as compared to existing strategy on trees.

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. Collision in multi-robot application has been avoided by modifying speed of the robot [8], updating steering of the robot [9–12] or deciding slaves' action based on that of master [13, 14], when possibility of a collision with other robot(s) is detected. Collision has also been avoided by developing Speed-Time Graph (STG) [15]. The decentralized strategy for multi-robot collision requires a sensor to be mounted on each robot to obtain the information on collision possibility. We propose a multi-robot collision avoidance strategy by asking following questions: Can we avoid collision with each other using limited sensor information? What should be this information, which would detect collision? Can we find a strategy in which each robot takes independent decision without knowing the locations and speeds of the robots colliding with each other? Is it possible to guarantee no collision by changing the speed of a robot without knowing the speed of other robots?

Our aim is to develop a technique in which each robot takes an independent decision avoiding redundancy for a graph-like environment. This decentralized approach considers no direct communication between the robots. The robot drops a beacon at first visit of a vertex. The information with a vertex's beacon is merged with the information of a robot visiting

this vertex. This work proposes a decentralized solution for graph exploration by an efficient use of limited information exchange and decentralized inter-robot collision avoidance technique. Each robot is equipped with a sensor (camera with limited image processing or range sensor) that is capable of finding the bearing of another robot in its field of view for avoiding inter-robot collision.

Our contributions in this paper are summarized as follows: The paper proposes a modified MR-DFS strategy using an efficient data structure. The incidence matrix as data structure is modified to accommodate a systematic and efficient multi-robot exploration. The modified incidence matrix records the partially explored graph information and represents map of the environment when exploration is complete. 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 MR-DFS method [3]. The improvement over MR-DFS [3] is guaranteed when difference between radius and half the number of edges of the tree is at least one.

A simple strategy to ensure collision avoidance between robots is developed. Examining the vertex as a shared resource that avoids conflicts by ensuring different arrival times at common vertex for multiple robots develops this collision avoidance strategy. A necessary and sufficient condition for collision is found. It is shown that this capability is sufficient to detect the possibility of collision. It is worth noting that the *sensor processing* capability is reduced with requirement of limited sensing information. 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, the proposed strategy suggests speed change for a robot without knowing the speeds of other robots and without communicating the decision of speed change. Various experimental and simulation results validate the proposed technique.

Remainder of the paper is organized as follows: Section 2 describes the terminology used in the proposed strategy. Section 3 presents the proposed multi-robot graph exploration technique providing a new condition to declare completion. Section 4 illustrates the proposed collision avoidance scheme. Section 5 presents simulation and experimental results of the proposed strategy. The proposed multi-robot exploration strategy is compared with existing methods in Section 6. Conclusion is presented in Section 7.

2 Terminology

The terminology of graph theory and exploration used in this paper is in the same spirit as in [3, 16]. Let the graph to be explored contains E edges and V vertices. The graph is unknown apriori (with unknown E and V) as the environment is unknown. 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. 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. Following terminology is used for describing the proposed technique. Each term is maintained by each robot and each vertex. Each term associated with the robot R_k is postfixed by (R_k) and that associated with the vertex v_j is postfixed by (v_j). The n^{th} update is referred by the superscript in the term.

- *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*. For example, 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).
- *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.

- *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).
- *Partially completed graph* $\mathcal{G}_c^n(.)$*:* The sets $G_c^n(R_k)$, $G_c^n(v_j)$ are ordered pairs $\left(\mathcal{V}_c^n(R_k), \mathcal{E}_c^n(R_k)\right)$, $\left(\mathcal{V}_c^n(v_j), \mathcal{E}_c^n(v_j)\right)$ 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.
- *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. It is worth noting that the zero angle is represented by 2π.
- *Incidence matrix* $\mathcal{I}^n(.)$*:* The definition of incidence matrix [16] 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.

3 Proposed Decentralized Multi-Robot Graph Exploration

The information of partial graph (represented by $G_c(.)$, $\mathcal{E}_o(.)$, $\mathcal{E}_u(.)$ and $\mathcal{E}_m(.)$) 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 initializations of 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:

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

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

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. 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 Eq. 1.

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

Remark 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 since their initial values, 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 Eq. 2.

$$\begin{aligned} G_c^n(v_j) &= G_c^n(R_k) \\ &= G_c^{n-1}(R_k) \cup \left(\{v_i, v_j\}, \{e_{ij}\}\right) \cup G_c^{n-1}(v_j). \end{aligned} \tag{2}$$

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, sets are merged using Eq. 3.

$$\begin{aligned} \mathcal{E}_o^n(v_j) &= \mathcal{E}_o^{n-1}(v_j) \cup \mathcal{E}_o^{n-1}(R_k) \setminus \mathcal{E}_c^n(v_j), \\ \mathcal{E}_u^n(v_j) &= \mathcal{E}_u^{n-1}(v_j) \cup \mathcal{E}_u^{n-1}(R_k) \setminus \left(\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)\right), \\ \mathcal{E}_m^n(v_j) &= \mathcal{E}_m^{n-1}(v_j) \setminus \left(\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)\right), \text{ and} \\ \mathcal{E}_m^n(R_k) &= \mathcal{E}_m^{n-1}(v_j) + \mathcal{E}_m^{n-1}(R_k) \setminus \left(\mathcal{E}_c^n(v_j) \cup \mathcal{E}_o^n(v_j)\right), \end{aligned} \tag{3}$$

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 Eq. 3. The proposed DFS strategy dictates the robot to choose an edge based on the following order of priority:

$$\{\mathcal{E}_m^n(R_k), \mathcal{E}_u^n(R_k), \mathcal{E}_o^n(R_k)\}, \tag{4}$$

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 Eq. 4. The information about selected edge is updated in second step. Let an edge e_{lm} be selected based on the priority described in Eq. 4 for next edge traversal, the second step update is described by Eq. 5.

$$\begin{aligned} \mathcal{E}_o^n(R_k) &= \mathcal{E}_o^n(v_j) = \mathcal{E}_o^n(v_j) \cup \{e_{lm}\}, \\ \mathcal{E}_u^n(R_k) &= \mathcal{E}_u^n(v_j) = \mathcal{E}_u^n(v_j) \setminus \{e_{lm}\}, \\ \mathcal{E}_m^n(v_j) &= \mathcal{E}_m^n(v_j) \setminus \{e_{lm}\} \text{ and} \\ \mathcal{E}_m^n(R_k) &= \mathcal{E}_m^n(R_k) \setminus \{e_{lm}\}. \end{aligned} \tag{5}$$

If the beacon at a vertex v_j initiates the update, simultaneous update of multiple robots visiting vertex v_j can be addressed. From Eq. 3, the set $\mathcal{E}_m^n(.)$ is updated with a possible removal of edges, therefore

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

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,

$$\mathcal{E}_m^n(v_j) \subseteq \mathcal{E}_u^n(v_j). \tag{6}$$

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 Eq. 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 Eq. 5, we get

$$\mathcal{E}_m^n(R_k) \subseteq \mathcal{E}_u^n(R_k). \tag{7}$$

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

Lemma 1 *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 Eq. 8.

$$\mathcal{G}^n = \left\{G_c^n(v_j)\right\} \ \forall\ j = 1, 2, \ldots, V^1. \tag{8}$$

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 Eq. 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

$$G_c^{n-1}(R_k) = G_c^p(v_i). \tag{9}$$

[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.

From Eqs. 9 and 2, each element of $\mathcal{G}^n$ is updated using Eq. 10.

$$G_c^n(v_j) = \begin{cases} G_c^p(v_i) \cup \left(\{v_i, v_j\}, \{e_{ij}\}\right) \cup G_c^{n-1}(v_j) & \text{when an edge } e_{ij} \text{is completed by any robot,} \\ G_c^{n-1}(v_j) & \text{otherwise.} \end{cases} \tag{10}$$

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

$$\mathcal{G}^0 = \{(\{v_1\}, \emptyset), (\{v_2\}, \emptyset), \ldots, (\{v_V\}, \emptyset)\}. \tag{11}$$

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

Let $n = 0$, Eq. 11 gives $G_c^0(v_j) = \left(\{v_j\}, \emptyset\right)$ 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 Eq. 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 $G_c^1(v_j) = G_c^0(v_i) \cup \left(\{v_i, v_j\}, e_{ij}\right) \cup G_c^0(v_j)$, (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. □

Theorem 1 *Exploration is complete if and only if*

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

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

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

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

$$\mathcal{E} = \{\mathcal{E}_c^n(R_k), \mathcal{E}_o^n(R_k), \mathcal{E}_u^n(R_k), \mathcal{E}_x^n(R_k)\}. \tag{12}$$

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

$$\mathcal{E} = \left\{\mathcal{E}_c^n(R_k), \mathcal{E}_x^n(R_k)\right\}. \tag{13}$$

When $\mathcal{E} = \emptyset$, $\mathcal{E}_c^0(R_k) = \emptyset$ and $\mathcal{E}_x^0(R_k) = \emptyset$ from Eq. 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 Eq. 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 Eq. 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 the start of 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 Eq. 12, we have $\mathcal{E}_u^n(R_k) = \mathcal{E}_o^n(R_k) = \emptyset$. □

Corollary 1 *$\mathcal{E}_u^n(v_j) \cup \mathcal{E}_o^n(v_j) = \emptyset$ 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 Eq. 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,

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

□

Theorem 2 *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 Eq. 4 at every update instance n.*

Proof From Eq. 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 Eq. 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). Figure 1 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 subgraph $G_c^n(R_k)$ is obtained using Eq. 10 and $e_{ij} \in \{\mathcal{E}_c^{n+n_1}(R_k) \mid n_1 \leq \left|\mathcal{V}_c^n(R_k)\right| - 1\}$. 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 $n_1 \leq \left|\mathcal{V}_c^n(R_k)\right|$. From Eq. 12, we have

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

We also have $\left|\mathcal{E}_u^n(R_k)\right| + \left|\mathcal{E}_o^n(R_k)\right| = 0$ when $\left|\mathcal{E}_c^n(R_k)\right| = E$ from Theorem 1. Since $\left|\mathcal{E}_c^n(R_k)\right|$ increments by one in finite number of edge traversals, it reaches to the value E in finite number of edge traversals. □

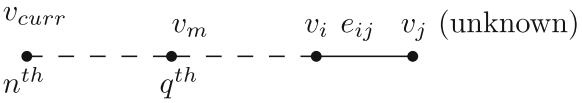


Fig. 1 Illustration of edge selection and approaching the connected vertex

Remark 2 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.

3.1 Proposed Implementation Using Modified Incidence Matrix

The modified incidence matrix described in Section 2 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 2). The proposed implementation suggests maintaining 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 Eq. 14.

$$\begin{aligned} I^0(R_k) &= [0] \ \forall\, k = 1, 2, \ldots, K, \\ I^0(v_j) &= \left[\theta_{j1} \ \ldots \ \theta_{jE_j}\right]. \end{aligned} \tag{14}$$

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

Algorithm 1 *Merge_Matrices*

Require: I_1 of dimension $V_1 \times E_1$ and I_2 of dimension $V_2 \times E_2$
Ensure: 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**
        $I[i_1, j_1] = I[i_2, j_2] = -|I[i_1, j_1]|$.
      end if
      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).
        $\hat{E}_1 = \hat{E}_1 - 1$.
      end if
      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**
        $I[V_1 + 1 : V_1 + V_2, j_1] = I[V_1 + 1 : V_1 + V_2, j_1] + I[V_1 + 1 : V_1 + V_2, j_2])$.
      end if
      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**
        $I[1 : V_1, j_2] = I[1 : V_1, j_2] + I[1 : V_1, j_1].)$
      end if
      Record i_1 in list del_i (for deleting i_1 row later).
    end if
  end for
end for
Delete rows recorded in list del_i and columns recorded in list del_j.

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 rows and j_1 to j_2 columns 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 2 *Order_Matrix*

Require: Merged incidence matrix I of dimension $\hat{V} \times (\hat{E_1} + \hat{E_2})$

Ensure: 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 Eq. 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*

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

Ensure: 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.

Remark 3 The sets 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 to 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.

Remark 4 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 1).

Remark 5 The incidence matrix is ordered, therefore termination condition needs to be checked with just the rightmost column vector.

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

[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.

Algorithm 4 *Second_Step_on_Vertex_Visit*

Require: $I^n(R_k)$ of dimension $\hat{V} \times (\hat{E}_1 + \hat{E}_2)$
Ensure: 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.
end if
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**
$I^n(R_k)[i, \hat{E}_1 + \hat{E}_2] = -I^n(R_k)[i, \hat{E}_1 + \hat{E}_2] \; \forall \; i = 1, \ldots, \hat{V}$.
end if
Circularly shift-right $I^n(R_k)[:, E_c + 1 : \hat{E}_1 + \hat{E}_2]$ by one column.
$I^n(v_j) = I^n(R_k)$.

Proposition 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 1, we get $\mathcal{E}_c^n(.) = \mathcal{E}$, if $\mathcal{E}_u^n(.) \cup \mathcal{E}_o^n(.) = \emptyset$, or

$$|\mathcal{E}_c^n(.)| = E. \tag{15}$$

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

$$\hat{E} = |\mathcal{E}_c^n(.)| + |\mathcal{E}_o^n(.)| + |\mathcal{E}_u^n(.)|. \tag{16}$$

From Eq. 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 Eq. 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} happen 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 Proposed Collision Avoidance Technique

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. Our objective is to detect collision (well before actual collision) using just the bearing information of a robot viewed from another robot. Lemma 2 shows that the possibility of collision at a vertex is identified using the rate of change of bearing.

Lemma 2 *Let the bearing angle of a robot R_p viewed from another robot R_k be β_{kp}. A necessary condition for collision is*

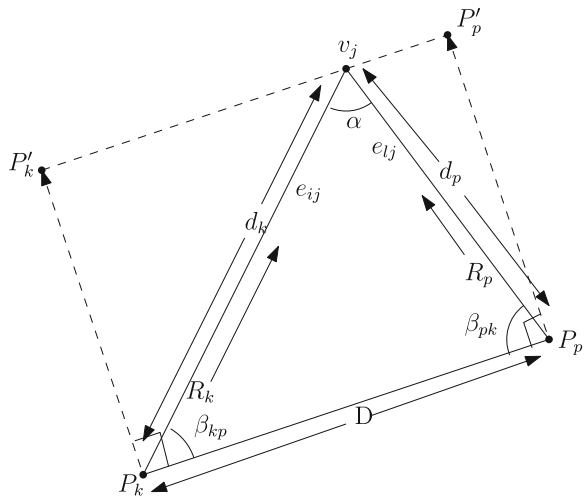


Fig. 2 Robot collision detection

$$\dot{\beta}_{kp} = 0. \tag{17}$$

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. 2. 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 $\alpha = \left|\theta_{ji} - \theta_{jl}\right|$. 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

$$\alpha + \beta_{kp} + \beta_{pk} = \pi, \tag{18}$$

or,

$$\dot{\beta}_{kp} = -\dot{\beta}_{pk}, \tag{19}$$

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

$$d_k \sin\beta_{kp} = d_p \sin\beta_{pk}. \tag{20}$$

Time derivative of Eq. 20 gives

$$\dot{d}_k \sin\beta_{kp} + d_k \cos\beta_{kp}\dot{\beta}_{kp} = \dot{d}_p \sin\beta_{pk} + d_p \cos\beta_{pk}\dot{\beta}_{pk}. \tag{21}$$

From Eqs. 19 and 21, we get

$$\dot{\beta}_{kp}(d_k \cos\beta_{kp} + d_p \cos\beta_{pk}) = \dot{d}_p \sin\beta_{pk} - \dot{d}_k \sin\beta_{kp}. \tag{22}$$

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 $P_k P_p$ 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

$$\dot{d}_k \sin\beta_{kp} = \dot{d}_p \sin\beta_{pk}, \tag{23}$$

and Eq. 22 renders

$$d_k \cos\beta_{kp} + d_p \cos\beta_{pk} = 0, \tag{24}$$

$$\text{or } \dot{\beta}_{kp} = 0. \tag{25}$$

But, Eqs. 20 and 24 gives

$$\frac{d_p}{\sin\beta_{kp}} \sin\beta_{pk} \cos\beta_{kp} + d_p \cos\beta_{pk} = 0. \tag{26}$$

Since $d_p \neq 0$, Eq. 26 gives $\beta_{kp} + \beta_{pk}$ as integer multiple of π. But, $\beta_{kp} + \beta_{pk} = \pi - \alpha$ from Eq. 18 which is possible only when $\alpha = 0$ or the two edges merge. Therefore, Eq. 25 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 Eq. 17. We propose a collision avoidance technique in which each edge is associated with two lanes as shown in Fig. 3. *A robot always follows the left side obstacle*. Theorem 3 proposes reduction in speeds of the robots for which the collision is detected.

Theorem 3 *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 Eq. 27, when the collision is detected with the robot R_p using Eq. 17 ($\dot{\beta}_{kp} = 0$),*

$$S_k = \begin{cases} 0 & \text{if } \beta_{kp} = 0, \\ S_k - \epsilon_k S & \text{otherwise.} \end{cases} \tag{27}$$

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

$$\epsilon_1 < (K-1)\epsilon_1 < \epsilon_2 < (K-1)\epsilon_2 < \epsilon_3 < \cdots < (K-1)\epsilon_{K-1} < \epsilon_K < (K-1)\epsilon_K < 1, \tag{28}$$

ensures inter-robot collision avoidance.

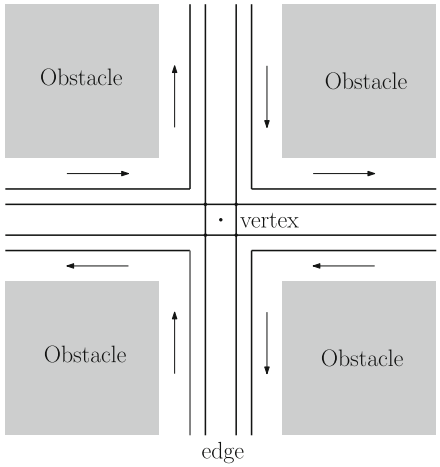


Fig. 3 Edges with lanes

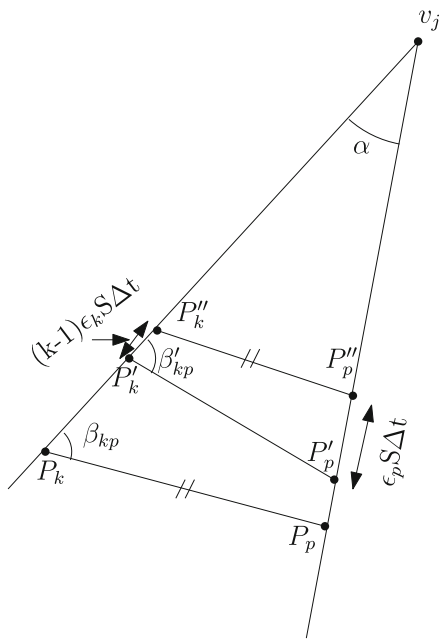


Fig. 4 Collision avoidance using independent decisions

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. 4. 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 Eq. 27. 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

$$P_k P'_k < P_k P''_k \text{ and } P_p P'_p < P_p P''_p.$$

The robots R_k and R_p reduce their speeds by $\epsilon_k S$ and $\epsilon_p S$ using Eq. 27. Without loss of generality, we consider $k < p$. From Eq. 28, we have $\epsilon_k < \epsilon_p$ and

$$S-\epsilon_k S > S-(K-1)\epsilon_k S > S-\epsilon_p S > S-(K-1)\epsilon_p S. \tag{29}$$

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

$$\epsilon_k S\Delta t \leq P'_k P''_k \leq (K-1)\epsilon_k S\Delta t, \text{ and} \tag{30}$$

$$\epsilon_p S\Delta t \leq P'_p P''_p \leq (K-1)\epsilon_p S\Delta t. \tag{31}$$

From Eqs. 29, 30 and 31, we have

$$P'_k P''_k < P'_p P''_p, \text{ or}$$
$$P_k P'_k > P_p P'_p. \tag{32}$$

In particular, Eq. 32 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 avoids collision 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 Eq. 33, when the robot R_p goes out of view provided speed reduction for avoiding collision with the robot R_p is applicable [3],

$$S_k = \begin{cases} S & \text{if } S_k = 0, \\ S_k + \epsilon_k S & \text{otherwise,} \end{cases} \tag{33}$$

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

$$S_k = S, \tag{34}$$

when it crosses the common vertex.

Remark 6 Changing speed S_k using Eqs. 27, 33 and 34 ensures 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 Eqs. 27, 33 and 34 under following possible occurrences of encountering two robots with each other:

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

1 A robot traversing in front of another with the same direction of traversal as shown in Fig. 5a.
2 A robot traversing in front of another with the opposite direction of traversal as shown in Fig. 5b
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. 5c.
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. 5d.
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. 5e.

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 Eqs. 27, 33 or 34:

– *Case 1:* Two robots traversing same edge in the same direction satisfy

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

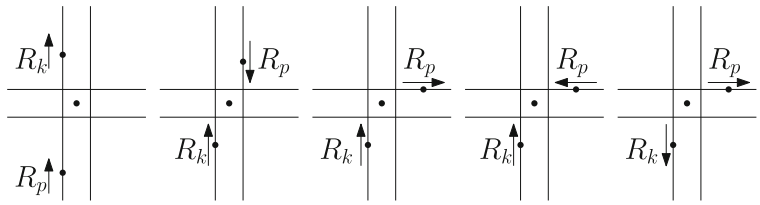
Speed reduction using Eq. 27 is applied, since $\dot{\beta}_{kp} = 0$ and speed of the observer robot R_k is changed to zero. When the robot moving in front of the observer robot is out of view, the observer robot changes its speed using Eq. 33 and it is modified back to S. Except the robot, which is 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 2). Theorem 3 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 Eq. 33, 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 (17) 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 Eq. 33. In particular, the



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

Fig. 5 All possibilities of encountering between two robots

speeds are changed to $S_k + \epsilon_k S$ and $S_p + \epsilon_p S$ respectively.

Corollary 2 *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 Eq. 27, 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 Eq. 27. 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 3.

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 $\beta_{p_i k} \neq 0$ and vice versa.

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, 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. □

5 Simulation and Experimental Results

Proposed multi-robot strategy is simulated using MATLAB ver R2014b for various tree and graph topologies. Figure 7 depicts tree topologies with 12 edges used for simulation.

The proposed technique is simulated for each topology shown in Fig. 7 using various number of robots

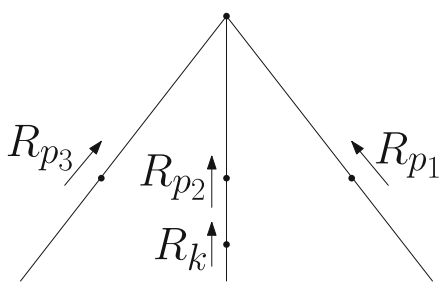


Fig. 6 Simultaneous collision detections with multiple robots

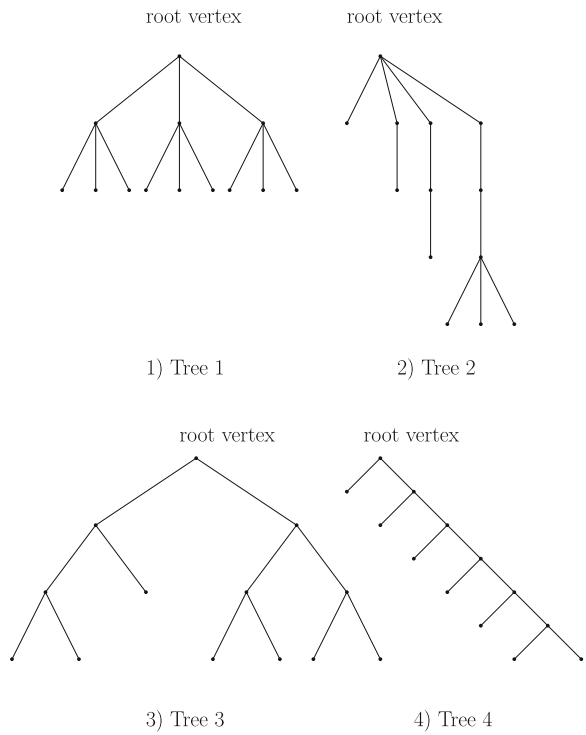


Fig. 7 Tree topologies with 12 edges

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. 8 (exploration time for a particular tree topology using given number of robots depends on the division of number of robots along edges, at a vertex therefore worst case exploration time is reported). Figure 7 also shows results using MR-DFS technique [3] for comparison. Comparison is further discussed in Section 6. Figure 8 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. 7 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. 9. This graph consists of 10 vertices and 12 edges. Each edge is considered being traversed in same time.

Figure 9 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 $v_1 - v_2 - v_4 -$

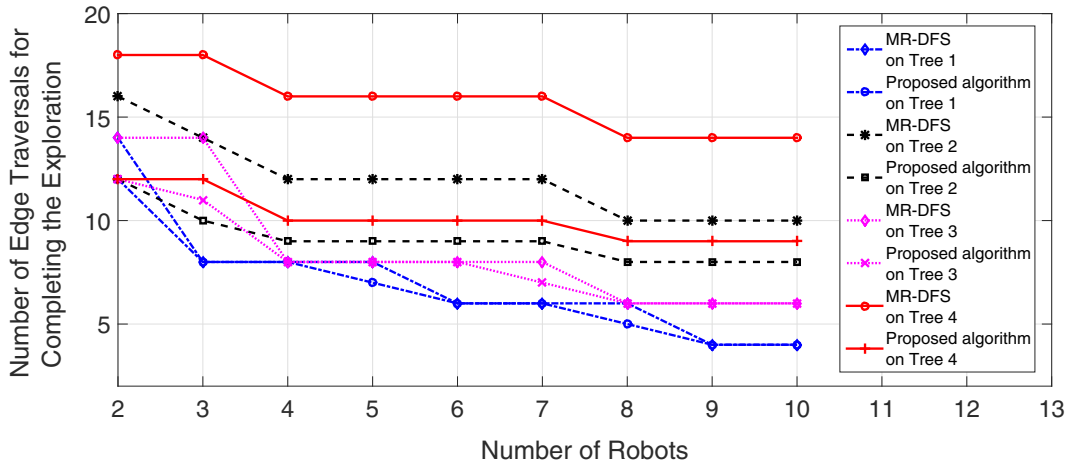


Fig. 8 Number of edge traversals required to complete exploration Vs. number of robots

$v_2 - v_5 - v_6 - v_5 - v_7 - v_8$ 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 experiments are conducted with a team of Spark V robots. Spark V as shown in Fig. 10 is a low cost differential drive mobile 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 driven using two DC motors in differential drive configuration. A free caster wheel is used to support the weight of the robot. 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 control the robot to follow a free path (marked using black strips) 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. The experiment is performed by retrieving/updating the information on vertices from a

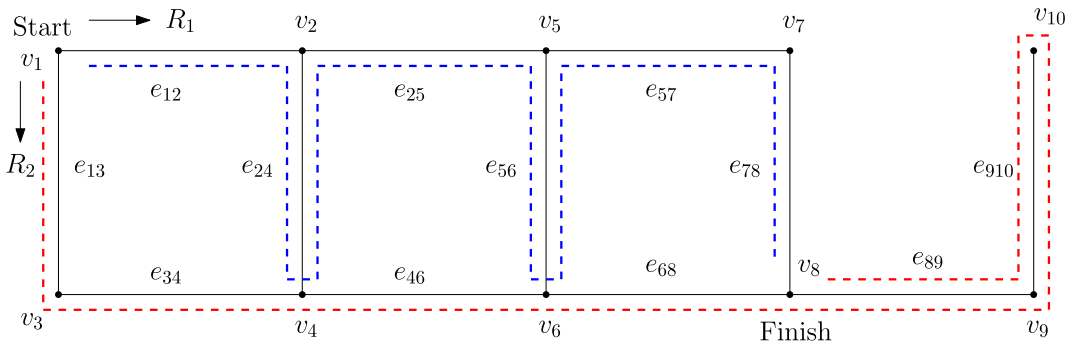


Fig. 9 Simulated result of a graph exploration using two robots

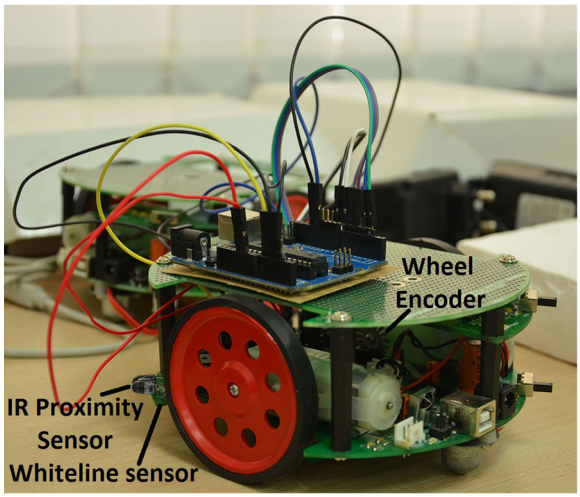


Fig. 10 Spark V Robot

central station through Xbee communication to emulate beacon dropping capability of a robot. The central station keeps separate information for each vertex.

Experiments are conducted on two arenas. Arena 1 as shown in Fig. 11 represents a graph with 8 vertices and 9 edges (edge represents free path). Arena 2 as shown in Fig. 12 represents a tree with 13 vertices and 12 edges. 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 along each edge, each corresponding to the direction of following left side obstacle.

The proximity sensor detects a vertex by detecting the absence of an obstacle. When a robot reaches

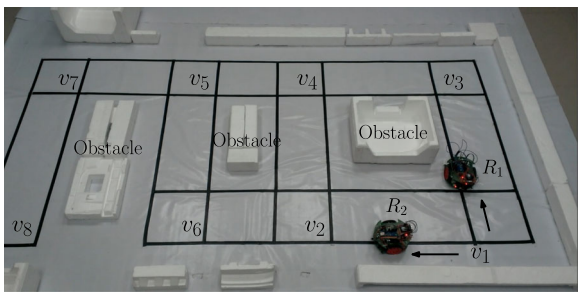


Fig. 11 Experiment on Arena 1 using two robots

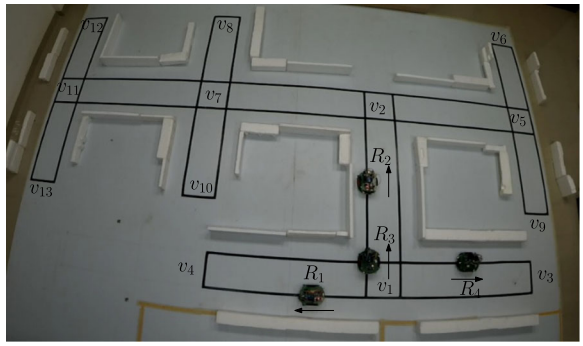


Fig. 12 Experiment on Arena 2 using four robots

a vertex, the robot updates the incidence matrix via Xbee and selects an edge for next traversal. Once an edge is selected for the traversal, the robot updates the incidence matrix with the selected edge information to the vertex (second step update).

Arena 1 is explored using two robots and Arena 2 is explored using four robots. Videos of these experiments are available at http://www.sc.iitb.ac.in/~leena/multi_robot_exploration.html. 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. The experiments also demonstrate the collision avoidance using the proposed speed changing approach. Initial speed of each robot is 5 cm/sec.

While exploring Arena 1, 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), when the robots are approaching a vertex. The exploration is completed in 7 edge traversals. 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 3. At various other instances during the exploration, the speed of the observer robot 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

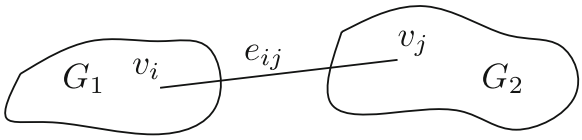


Fig. 13 Illustration of partitioning of a tree in two subtrees G_1 and G_2

maximum distance between two robots while stopping the observer robot would result in faster exploration.

6 Comparison with Existing Methods

We compare proposed technique with MR-DFS [3] and Flooding [4] 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 1 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 avoids 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 MR-DFS [3] and Flooding [4] using two robots is $E+r$, where r is radius of the tree. We show benefit of proposed method using following proposition.

Proposition 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

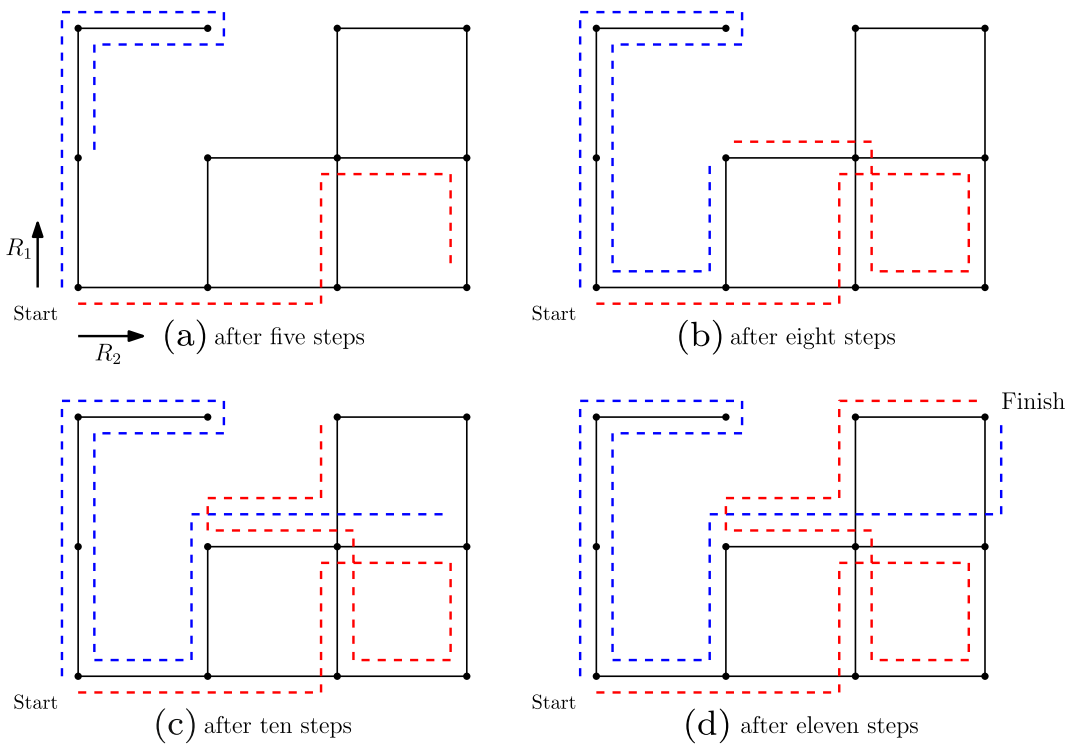


Fig. 14 Exploration using proposed technique on an example graph from [3]

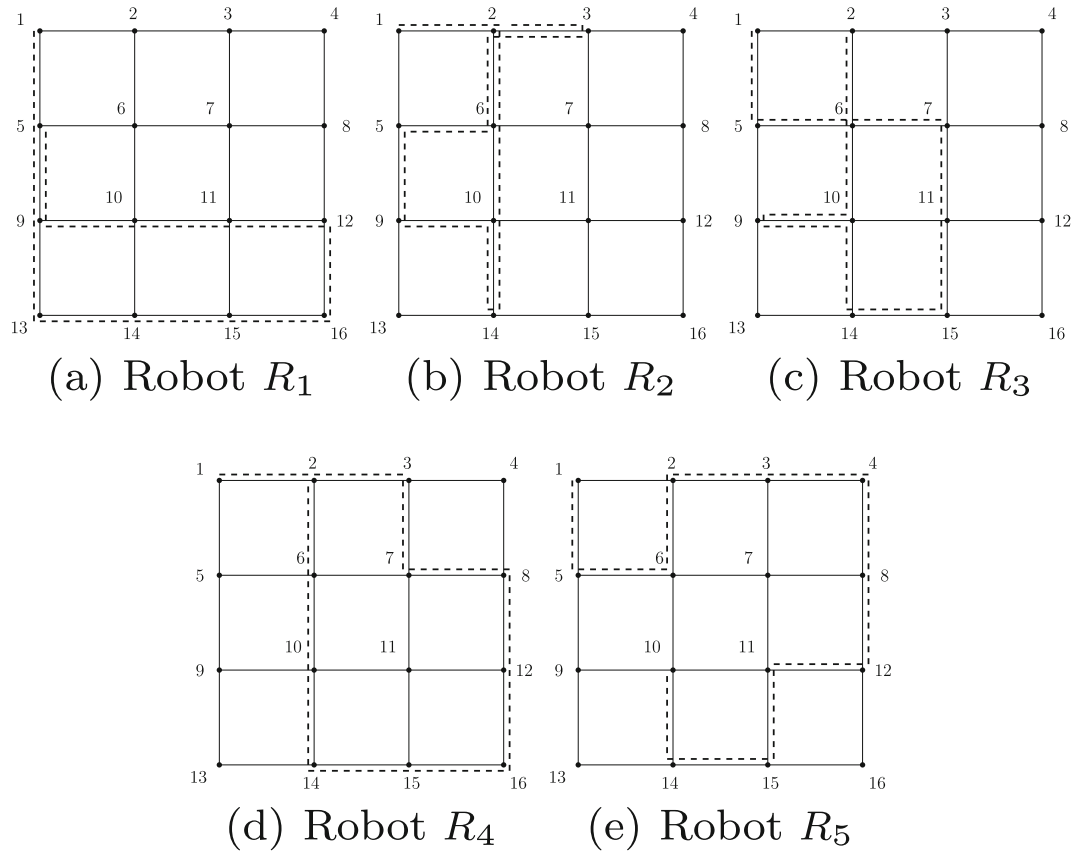


Fig. 15 Exploration using proposed method with 5 robots

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. 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 modified MR-DFS is

$$E_l = \max(2E/K, r + 1) \tag{35}$$

as compared to $\max(2E/K, 2r)$ for MR-DFS [3].

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 MR-DFS method. Therefore, the worst case analysis for finding the exploration time in [3] using excess multiplicity of edges is applicable to the proposed MR-DFS method as well. Now, upper bound for number of edge travers als required to complete exploration using modified MR-DFS (from Proposition 2 and analysis on excess multiplicity of edges [3]) is given by

$$E_u = \min\left(E, \frac{2E}{K} + \frac{2}{K}\left(1+\frac{K}{r}\right)^{K-1}\frac{2}{K!}r^{K-1}\right). \tag{36}$$

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 Eqs. 35 and 36 with corresponding lower and upper bounds for MR-DFS [3] also show that modified MR-DFS gives larger benefit in total edge traversals required to complete the exploration as compared to existing MR-DFS [3] when $r > E/2+1$.

The benefit is verified by exploring tree topologies (shown in Fig. 8) using various number of robots. Fig. 8 conveys that the number of edge traversals required for completing the exploration using proposed technique is always less or equal to that using existing MR-DFS [3]. 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 [3] with 12 vertices and 14 edges. Exploration steps using proposed technique for two robots R_1 and R_2

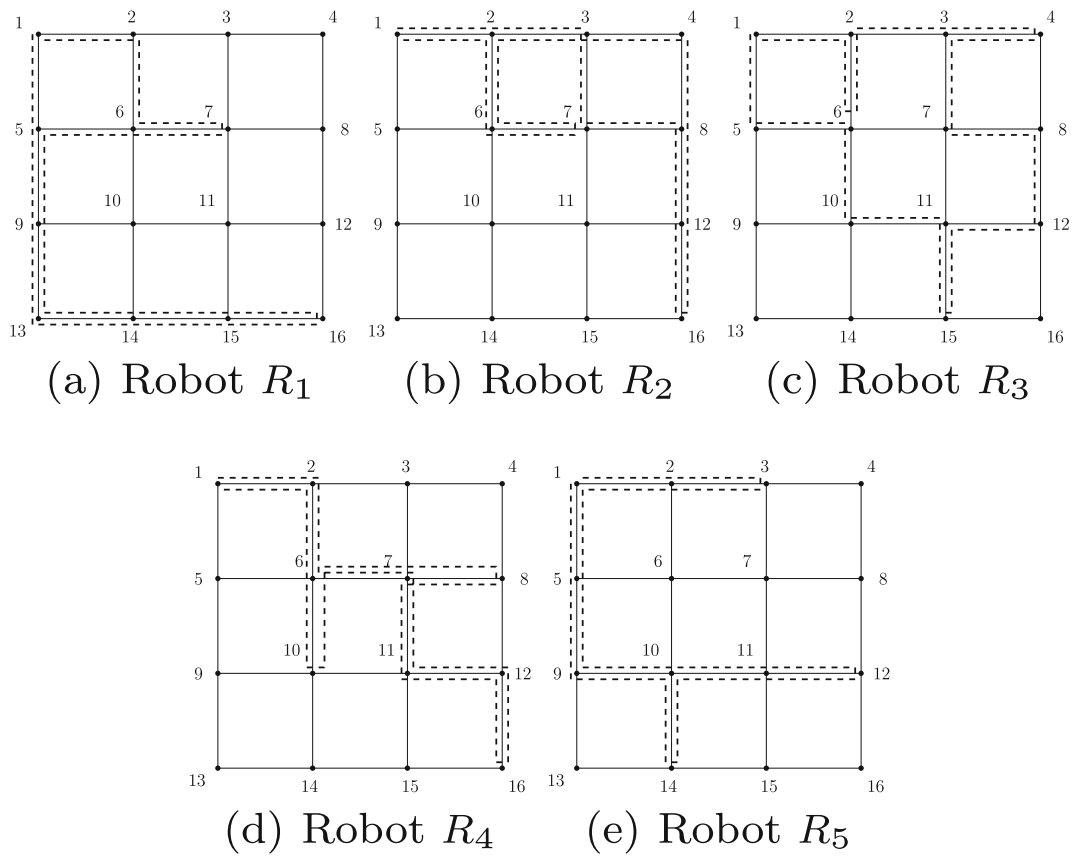


Fig. 16 Exploration using the MR-DFS [3] with 5 robots

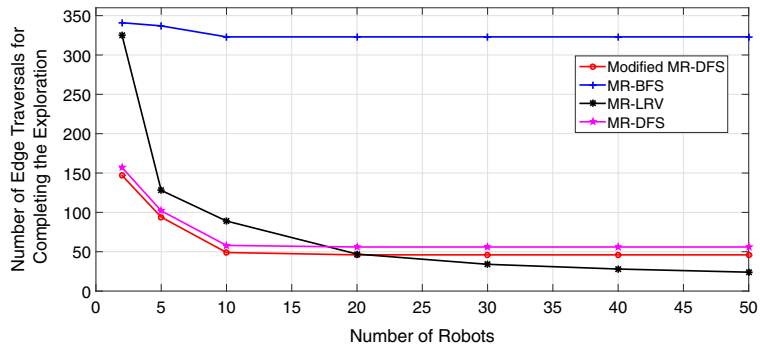


Fig. 17 Number of edge traversals required to complete exploration Vs. number of robots for a square lattice with 100 vertices and 180 edges

are shown in Fig. 14. For both the strategies, the worst case exploration time are compared. The MR-DFS method [3] 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 MR-DFS methods. These exploration steps are shown in Figs. 15 and 16. The MR-DFS algorithm [3] with 5 robots takes 16 edge traversals to complete the exploration, whereas the proposed strategy takes 11 steps to complete the exploration. Figures 15 and 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 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 [5, 6], LRV [7] and MR-DFS [3] methods on a square lattice graph with 100 vertices and 180 edges. The simulation results shown in Fig. 17 present the total edge traversals required to complete the exploration for various values of K. Figure 17 shows the benefit obtained in total edge traversals using modified MR-DFS.

7 Conclusion and future work

The paper presents a decentralized technique for exploring a graph-like unknown environment using multiple robots. A modification in existing multi-robot DFS technique is proposed and new condition for declaring completion of exploration is obtained. The new condition is capable of declaring the completion without reaching to 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 information of partially completed map and unexplored edges with various status, thus reducing redundancy in exploration. The robots while exploring the graph-like environment have the possibility of colliding with each other. A simple strategy for avoiding inter-robot collision is proposed. This strategy is based on changing 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 proposed technique is analyzed on trees and graphs. It is shown that the proposed modification in existing MR-DFS strategy [3] performs better than the single robot DFS strategy and the existing multi-robot DFS strategy. The proposed technique guarantees completion of exploration in finite number of edge traversals. Experiment and simulation results validate the theory presented in this paper.

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. Liu, Y., Nejat, G.: Robotic urban search and rescue: A survey from the control perspective. J. Intell. Robot. Syst. **72**(2), 147–165 (2013)
2. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005)
3. Brass, P., Cabrera-Mora, F., Gasparri, A., Xiao, J.: Multirobot tree and graph exploration. IEEE Trans. Robot. **27**(4), 707–717 (2011)
4. Cabrera-Mora, F., Xiao, J.: A flooding algorithm for multirobot exploration. IEEE Trans. Syst. Man Cybern. B: Cybern. **42**(3), 850–863 (2012)
5. Fleischer, R., Trippen, G.: Exploring an unknown graph efficiently. In: Algorithms–ESA, pp. 11–22. Springer (2005)
6. Wang, H., Jenkin, M., Dymond, P.: Enhancing exploration in graph-like worlds. In: Proceedings of the Canadian Conference on Computer and Robot Vision (CRV), pp. 53–60 (2008)
7. Batalin, M.A., Sukhatme, G.: The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. IEEE Trans. Robot. **23**(4), 661–675 (2007)
8. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**(1), 23–33 (1997)
9. Rodriguez-Seda, E.J.: Decentralized trajectory tracking with collision avoidance control for teams of unmanned vehicles with constant speed. In: Proceedings of the American Control Conference (ACC), pp. 1216–1223. IEEE (2014)
10. Dimarogonas, D.V., Kyriakopoulos, K.J.: 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, 2005 and 2005 European Control Conference, pp. 84–89. IEEE (2005)
11. Chang, D.E., Shadden, S.C., Marsden, J.E., Olfati-Saber, R.: Collision avoidance for multiple agent systems. In: Proceedings of the 42nd IEEE Conference on Decision and Control, pp. 539–543. IEEE (2003)
12. Kumar, S., Parekh, T.P., Madhava Krishna, K.: A hierarchical multi robotic collision avoidance scheme through robot formations. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 306–311. IEEE (2010)
13. Souliman, A., Joukhadar, A., Alturbeh, H., Whidborne, J.F.: Real time control of multi-agent mobile robots with intelligent collision avoidance system. In: Proceedings of the Science and Information Conference (SAI), pp. 93–98. IEEE (2013)
14. Vrba, P., Marik, V., Preucil, L., Kulich, M., algorithms, D.avid.Sislak.C.ollision.a.voidance.: Multi-agent approach. In: Holonic and Multi-Agent Systems for Manufacturing, pp. 348–360. Springer (2007)
15. Hwang, K.-S., Ming-Yi, J.: Speed planning for a maneuvering motion. J. Intell. Robot. Syst. **33**(1), 25–44 (2002)
16. Godsil, C., Royle, G.F.: Algebraic Graph Theory. Springer, New York (2001)