# Gathering Of Mobile Robots In Anonymous Trees

Asha Mary Joseph

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Department of Computer Science And Engineering

June 2015

## Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others 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 a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

[illegible]

(Signature)

(Asha Mary Joseph)

(CS13M1002)

## Approval Sheet

This Thesis entitled Gathering Of Mobile Robots In Anonymous Trees by Asha Mary Joseph is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

(U. RAMAKRISHNA) Examiner
Dept. of Computer Science And Engineering
IITH

[illegible]

(M. V. Panduranga Rao) Examiner
Dept. of Computer Science And Engineering
IITH

[illegible]

(Dr. N. R. Aravind) Adviser
Dept. of Computer Science And Engineering
IITH

[illegible]

(T. Bheemarjuna Reddy) Chairman
Dept. of Computer Science And Engineering
IITH

# Acknowledgements

First of all, I would like to express my sincere gratitude to my adviser Dr. N. R. Aravind for his valuable guidance and support throughout the project.

I would also like to thank my family and friends for supporting me throughout my studies at the institute, IIT Hyderabad.

Above all, I thank GOD ALMIGHTY for enabling me in completing this work successfully.

# Abstract

Gathering problem of mobile robots is a class of graph problem that has a lot of relevance in everyday life. The problem requires a set of mobile robots, initially located at different nodes of a graph, to gather at the same location in the graph, which is not decided before. This report considers the gathering problem of mobile robots in anonymous trees. The robots considered here are identical, do not communicate directly with other robots and also, all the robots execute the same algorithm to achieve gathering. Robots are assumed to have minimal capabilities with respect to the memory associated with them as well as their visibility capability. In this report, three models have been proposed for solving gathering problem under three different scenarios. Possible solutions in each of these models have been described. The current work that has already happened and the future work that can be done in each model have also been mentioned.

Exploration of graphs by mobile robots is another interesting class of graph problem. While trying to achieve gathering, robots keep exploring the graph till they gather at some location. Thus exploration problems are closely related to gathering problem and efficient exploration problem solutions could be used to improve efficiency of gathering. This report provides pseudocode for an algorithm (modified Rolling Dispersion Algorithm [1]) that allows robots to explore a tree, while remaining connected to each other. That is, all the robots stay together as a group forming a connected subgraph (where the distance between two robots cannot be more than a certain value, say d), while they explore the tree.

# Contents

# Chapter 1

# Introduction

An anonymous graph is a graph in which the vertices are not labelled. This report considers a particular class of graph problems called the gathering problem of mobile robots in anonymous trees. In the gathering problem, all the robots that were initially placed at different nodes of a graph need to gather at a single node, which is not decided before. If the vertices are labelled, then the robots can decide to meet at a predetermined vertex, which would solve the gathering problem. But in most of the cases, it would not be possible for vertices to have such unique labels or robots may not be able to identify such a unique labelled vertex due to limited sensing capability[2]. This makes the gathering problem in anonymous graphs more interesting. This report considers the case of gathering in anonymous trees. Gathering problem has many applications in day to day life. For example, as described in [3], the mobile robots could represent people who would like to meet in a city, wherein the different places in the city form the different nodes of the network. In computer science, mobile robots could represent software agents when the network is a labyrinth. There could be several reasons why the robots would like to gather at a certain node such as exchanging information already collected by the robots or to coordinate some future task such as network maintenance or finding the map of a network[3].

The mobile robots considered in this study are identical and are assumed to have minimal capabilities especially in terms of memory associated with the robots and visibility of each robot. This is particularly important as resources, such as memory, are very limited and precious in robot based computing systems. Also robots may be very small, cheap and mass-produced devices. Adding distinct labels, memory, or communication capabilities makes production of such devices more difficult and increases their size and price, which is not desirable[4]. Hence it would be desirable to consider identical robots that cannot be distinguished from one another. The scenario considered in this study is made even more interesting and the capabilities of robots are even more minimised and weakened by the assumptions that robots do not communicate directly with other robots and also execute the same deterministic algorithm to achieve gathering.

Several different capabilities have been assumed for robots in different studies, to enable them in solving different classes of robot based computing problems in graph theory. Some of these capabilities include assuming a common coordinate system for all the robots, multiplicity detection capability for robots wherein a robot can sense the number of robots at a particular node during Look operation (during which a robot perceives current global configuration of the network), limited visibility wherein a robot can see other robots only upto a certain level k, unlimited visibility wherein the robot can see the entire network, presence of a shared compass for all the robots, oblivious wherein the robot has no persistent memory associated between different cycles of its operation and non oblivious wherein robot has memory associated with it. The multiplicity detection capability of a robot can be of different types [5]. In case of global strong multiplicity detection, a robot can sense the exact number of robots at each node while in case of global weak multiplicity detection, a robot can only say whether there is one or multiple robots at every node. In case of local strong multiplicity detection, a robot can sense whether a node is occupied or not and also can determine the exact number of robots in the node where the robot resides while in local weak multiplicity detection, a robot can only determine whether the node where it resides has multiple robots or not. Many of the previous studies that solved gathering problem assumed strong capabilities for robots such as unlimited visibility, unlimited memory, distinct labels etc. But in this study, the robots are assumed to possess only minimal capabilities, especially in terms of memory and visibility, to achieve gathering.

Several challenging models of robot based computing on anonymous graphs have been studied extensively. All the models that are proposed in this study are based on a model called the asynchronous Look-Compute-Move model, which is described in [5] as follows. Identical robots initially placed in different nodes of an anonymous graph perceive the current global configuration by a Look operation, and based on the perceived configuration decide to either perform a move to a nearby node or to stay idle (Compute operation) and in the former case of Compute operation, perform a move( Move operation). Since the model is asynchronous, the amount of time required for each robot to perform the different operations Look, Compute and Move could differ. The time required for each operation is unbounded, but finite. It is also assumed that during any Look operation, a robot sees other robots on nodes of the graph and not on edges.

# Chapter 2

# Literature Survey

Many problems have already been studied in the area of robot based computing on anonymous graphs. Some of them are gathering problem, graph exploration problem, pattern formation problem, searching in network with liars, constructing maps of unknown environment, leader election problem, edge election problem, spanning tree construction problem and topology recognition problem. A lot of research has already been done in these areas and these studies show that many of these problems are closely related to each other. While solving gathering problem for multiple robots, the robots explore the graph until they gather at some location. Hence exploration problems and gathering problems are very much related and the solution to exploration problems could be used while solving gathering problem. The formation of geometric pattern by mobile robots is yet another interesting problem that is closely related to gathering problem. When the pattern that the robots form is a point, the pattern formation problem becomes the same as gathering problem. The remaining part of this section describes some of the studies that has already happened in the area of gathering problem and exploration problem in graphs. The major results that they have proposed have also been quoted.

This report focuses on solving gathering problem or rendezvous problem of mobile robots in anonymous trees. This is one of the most interesting and classic problems in robot based computing on anonymous graphs and has been of interest for researchers for quite a long time. Gathering problem has been studied for robots moving in two-dimensional plane as well as in discrete models like graphs. A lot of reasearch has happened assuming different capabilities for robots so as to enable them in achieving gathering. Gathering of asynchronous robots with limited visibility was studied in [6]. They proposed a deterministic algorithm that solves the problem in finite time when the robots have a sense of orientation. Gathering of two asynchronous mobile robots equipped with inaccurate compasses was studied in [7]. They provided a self-stabilizing algorithm to gather, in a finite time, two oblivious robots equipped with compasses that can differ by as much as $\frac{\pi}{4}$. Gathering an even number of robots in an odd ring without global multiplicity detection was studied in [8]. They proposed a gathering protocol for an even number of robots in a ring-shaped network that allows symmetric but not periodic configurations as initial configurations, and uses only local weak multiplicity detection. It required the number of robots k to be greater than 8 and the number of nodes n on a network to be odd and greater than $k + 3$. The running time was found to be O($n^2$) asynchronous rounds. The problem of gathering identical, memoryless, mobile robots in one node of an anonymous unoriented ring was studied in [4]. They provided gathering algorithms for initial configurations proven to be gatherable. For an odd number of robots, it was shown that gathering is feasible if and only if the initial configuration is not periodic. For an even number of robots feasibility of gathering was decided except for one type of symmetric configurations. Gathering fat robots i.e, forming a configuration for which the union of all discs representing robots is connected, was studied in [9]. This was the first algorithmic result on gathering robots represented by two-dimensional figures (fat robots) rather than points in the plane and they solved gathering problem for atmost four robots. The robots considered were autonomous, asynchronous identical robots, represented by unit discs, that move deterministically in a plane without a common coordinate system, without doing any communication and without any memory of the past. Gathering of many asynchronous oblivious robots on a n undirected ring was studied in [10]. They have provided a new technique for dealing with symmetric configurations based on preserving symmetry rather than breaking it. They have provided procedures for gathering all configurations on the ring with more than 18 robots for which gathering is feasible, and gives a full characterization of all such configurations. The problem in which two robots with distinct labels have to meet in an arbitrary, possibly infinite, unknown connected graph or in an unknown connected terrain in the plane was studied in [11]. They showed that (deterministic algorithm) rendezvous is possible in any connected countable (finite or infinite) graph, starting from any nodes, without any information of the graph. The only thing an agent needs to know is its own label. Gathering of distributed system of autonomous mobile robots that are able to freely move in the two-dimensional plane without any central control was studied in [12]. They have proposed an algorithm that solves the gathering problem for any initial conguration of the robots in a plane. [5] is a survey on recent results obtained for the gathering task over basic graph topologies i.e, rings, grids and trees. Gathering problem in regular bipartite graph of degree

$\delta$ has been studied in [13]. It was shown that the class of gatherable initial configurations are those in which the robots form a star configuration. It was also assumed that robots have only vision of immediate neighbourhood and has global or local weak multiplicity detection capability.

Another set of problems that has been studied extensively is graph exploration problems. Constrained perpetual graph exploration (CPGE) without collision assuming the presence of omniscient daemon that can coordinate robot movements has been studied in [14]. An upper bound on the number of robots that can explore a graph keeping CPGE solvability has been established. Finding the shortest time required for graph exploration such that each robot visits all the graph nodes and returns to its starting location has been studied in [15]. Tight lower and upper bounds on exploration time has been determined for general graphs and the exact value has been determined for trees, provided the mobile robots are aware of the network as well as their initial positions. Deterministic terminating grid exploration (Robots explore the grid and stops after all the nodes are explored by atleast one robot) has been studied in [16]. They have established that 3 robots are necessary and sufficient to deterministically explore a grid of at least three nodes. Also the optimal number of robots for the two remaining cases is: 4 for the (2,2)-Grid and 5 for the (3,3)-Grid, respectively. The problem of exploring an anonymous unoriented ring of size n by k robots that can sense their environment and take decisions based on their local view has been studied in [17]. They could prove that no deterministic exploration is feasible with less than five robots, and that five robots are sufficient for any n that is coprime with five. Also exploration is completed in O$(n)$ robot moves which is optimal. Constrained exploration of an unknown graph $G = (V, E)$ from a given start node s with either a tethered robot or a robot with a fuel tank of limited capacity has been studied in [18]. They have shown that exploration of the graph can be done with $\theta(|E|)$ edge traversals. The problem wherein a robot has to traverse all nodes and edges of a network (represented as an undirected connected graph), and return to the starting node has been studied in [19]. They have presented an algorithm to accomplish tree exploration (with return) using O$(logn)$-bit memory for all n-node trees. Memory efficient exploration of anonymous graphs has been studied in [20].

# Chapter 3

# Problem Formulation

This section provides a description of the three models that are being considered for solving gathering problem in three different scenarios. Proposed solution for solving gathering problem in each of these three cases has been discussed. The current work that has already happened and the future work to be done in each case has also been mentioned. Also a modified version of Rolling Dispersion Algorithm [1] for tree exploration has been discussed.

In all the models described below, the network is modelled as an anonymous tree wherein the vertices are not labelled. But it is assumed that the edges are labelled at a vertex from 1...d, where d is the degree of that vertex. Thus by means of these port numbers (labels of edges at vertices), it is assumed that robots can locally distinguish edges at a vertex. This assumption is important as otherwise robots might not be able to visit all its neighbours and hence it could happen that two robots that are adjacent to a vertex would never be able to meet [21]. Each edge (u,v) between vertices u and v will have two port numbers, one at u and one at v. These numbers could be same or different at u and v. It is assumed that when a robot enters a vertex, it is aware of the port number through which it is entering and the degree of the vertex and similarly when a robot leaves a vertex, it is aware of the port number through which it leaves [21]. A sample port numbering has been shown in Fig. 3.1.

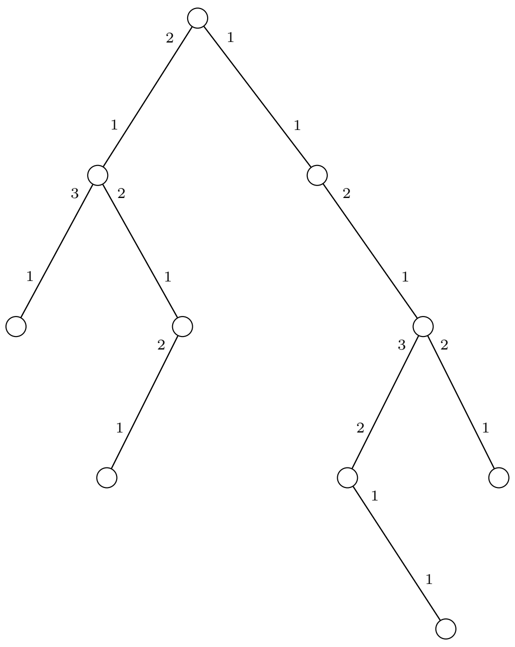

Figure 3.1: Port numbering a tree

## 3.1 Model 1

### 3.1.1 Proposed Model

In this model, we have k identical mobile robots initially placed at different nodes of the tree. It is required that all these robots meet at a single node or on an edge of the tree by following a deterministic algorithm. The robotic movements are asynchronous and the visibility of each robot is limited to 1. Hence a robot can only see its neighbours and does not have visibility of the complete network. Each robot has certain limited memory associated with it and the robots have the capability of global strong multiplicity detection. This is different from the classic gathering problem in anonymous trees as it allows robots to meet on an edge, if required.

This model could be used to represent the real life scenario wherein mobile robots represent people who are at different locations of a city, who would like to meet at a single place[3].

### 3.1.2 Existing Work

A tree has a distinguished vertex called the center of the tree. In a tree, there can be either precisely one center or there can be two centers. If there is only a single center, all the robots can gather at this node. This idea of gathering at a node was proposed in [5]. They had also mentioned an impossibility result in the case of tree with two centers which states that if the two subtrees rooted at the centers have an isomorphic disposal of robots, then gathering is impossible. If the subtrees rooted at the centers are not isomorphic with respect to the disposal of robots, then gathering is achieved by moving all the robots from the subtree with smaller number of robots to the root(which is the other center) of the subtree with larger number of robots. But if both the subtrees have same number of robots, then the choice of movement depends on the natural ordering in labelled trees. To define the smaller subtree as the one with the robots closer to the root, label 1 is associated to empty nodes and label 0 to nodes occupied by robots. Using this ordering, algorithm detects the robots to move from one subtree towards the root of the other one. If a robot moves over a node already occupied, the number of occupied nodes in the original subtree decreases and once the subtrees have different number of robots, gathering can be achieved by moving robots to the root of the larger subtree.

### 3.1.3 Solving gathering problem in proposed model

The solution mentioned in [5] does not explicitly comment on the method used to determine the center of the tree, neither is a limit mentioned on the memory requirement and visibility of robots. In the model that I have described, I am trying to propose a memory efficient gathering algorithm for mobile robots with limited visibility (robots can only see their neighbours) in anonymous trees. Also by allowing robots to meet on the edges of the tree, gathering is shown to be possible in all the cases unlike the impossibility result in [5]. Such meeting of robots (inside the edge) would be possible, for example, when we consider robotic movements in a labyrinth.

The solution uses a modified version of the standard Depth First Search (DFS) algorithm to determine the center of the tree. Each robot performs the following steps and determines the center of the tree.

1. Do a modified DFS on node of robot (v) to find the farthest node u from v.
2. Do a modified DFS from u to find the farthest node w from u.
3. Store the path u-w.
4. If the length of the path is even (only one center), center of the tree is at distance $\frac{length}{2}$.
   Else (there will be two centers) center1 will be at distance floor( $\frac{length}{2}$) and center2 will be at

distance ceil( $\frac{length}{2}$).

If there is only a single center for the tree, all the robots will gather at the center and hence the gathering problem would be solved. For example, consider Fig. 3.2. Let there be two robots in the tree. Let R1 be at node 8 and R2 at node 10. Both the robots would initially perform the four steps described above to determine the center of the tree. Both the robots will find the path length (length of diameter of the tree) to be six and hence finds out that node 1 which is at distance three is the center. Now both R1 and R2 will move towards node 1 and gathering occurs at node 1. (The nodes are numbered only for ease of explanation).

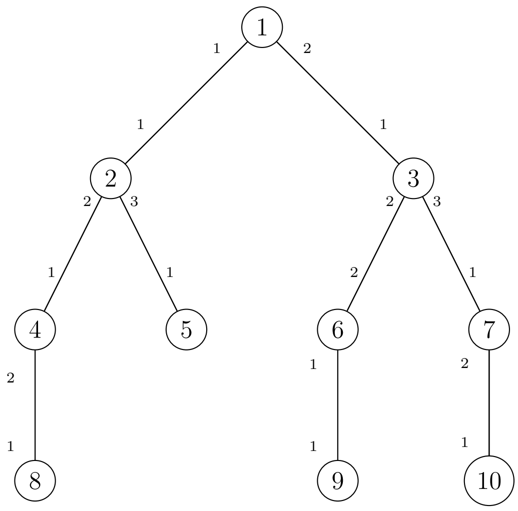
Figure 3.2: Model 1- Tree with single center

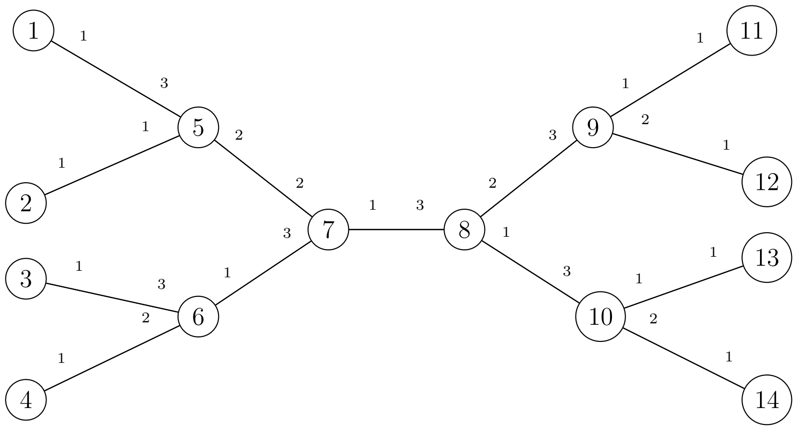
Figure 3.3: Model 1- Tree with two centers

If the tree has two centers, each robot moves to the center which is closer to it, if there is a free path from its current node to the center. If such a path does not exist, it waits until such a path is available. Once the sum of multiplicities at both centers becomes equal to the number of robots k in the network, each robot determines the multiplicity at both the centers. If the multiplicities are not equal, the robots in the center with smaller multiplicity moves to the center with greater multiplicity. If both the centers have same multiplicity, robots would be allowed to meet inside the edge connecting the two centers. For example, consider Fig. 3.3 (The nodes are numbered only for ease of explanation). Let there be 5 robots placed at node 1, 3, 4, 11, 14. Now all the five robots would execute the steps mentioned above to determine center and would find that there are two centers for the tree, that is node 7 and node 8. Now the robots would start moving to the center closest to them and thus we will have 3 robots at node 7 and 2 robots at node 8. Once all the robots have reached the centers, robots would determine the multiplicity at node 7 and node 8. Since node 8 has lesser number of robots, those robots would now move to node 7 and gathering occurs at node 7. Now let us see the case when we have 6 robots placed at nodes 1, 3, 4, 11, 13, 14. Once robots move to centers, we will have 3 robots at node 7 and 3 robots at node 8. Now in this case, all the robots would move towards the other center and meet inside the edge connecting node 7 and node 8.

In order to perform the modified DFS algorithm to find the center of the tree, each robot needs to find the farthest node from its initial location and then store the nodes(using port numbers) along the path from this new node to the new node's farthest node. In the worst case, the number of nodes in this path could go upto $n - 2$, where n is the total number of nodes in the tree. In this case robot requires O($n$) bits to store these nodes. Also if the maximum degree of nodes along this path is $\Delta$ and let the depth of exploration be d, then the robots would require to store O($d * \Delta$) bits so that the algorithm can keep track of already visited nodes while backtracking to explore all the paths. By keeping track of the longest of such paths explored, the diameter of the tree can be found out and thus the center. Thus the memory associated with each robot would be O($n$) + $O(d * \Delta)$ bits in order to perform gathering.

### 3.1.4 Future work in proposed model

[2] suggests an upper bound of $O(log\Delta + logm)$ and lower bound of $\Omega(logloglog\ n)$ as memory requirement for exploration with stop (at some node) on a tree with n nodes and maximum degree $\Delta$. The upper bound assumes that the robot knows an upper bound m on the number of nodes of

a given degree. One open question in our model would be whether these results could be used to provide tighter bounds on memory requirement of each robot for solving gathering problem.

## 3.2 Model 2

### 3.2.1 Proposed model

In this model, we have k identical mobile robots initially placed at different nodes of the tree. This is different from Model 1 as it requires gathering of robots on a node of the tree and not on edges. The robotic movements are asynchronous and all the robots follow the same determinsitic algorithm for gathering. The capabilities of robots with respect to visibility is same as that of Model 1 and robots are also assumed to have multiplicity detection capability. Each robot is assumed to have certain limited memory associated with it. But the capability of robots with respect to memory differs from Model 1 due to the presence of public bits shared by all the robots. The access to the shared bit, by the robots, is subject to mutual exclusion so that simultaneous access would not result in concurrency issues. This model aims at reducing the amount of private memory required by each robot for gathering, by using shared public bits. This study considers the case when the robots have access to a single public shared bit.

This model could represent a scenario in a network with many identical robots continuously monitoring the network. If a robot senses an emergency at its location and requires additional help, it can set the public bit that is accessible by all the other robots and can remain stationary. The other robots could then gather at the location of emergency to provide additional help.

### 3.2.2 Existing Work

To the best of my knowledge, this model has not been studied so far. The idea of providing certain amount of memory to each node of the network, which can be modified by any robot visiting the node had been used in some studies. But in the proposed model, the memory shared across all the robots is not associated with any particular node of the network.

### 3.2.3 Solving gathering problem in proposed model

The shared public bit is initially not set. Every robot checks the value of the public bit regularly. If a robot sees that the bit is not set and it sets the bit, then that robot stays stationary afterwards.

All the other robots that sees the set bit, explores the network until they find the stationary robot. The robots have asynchronous Look Compute Move timesteps and hence the first robot that gets to set the public bit would be the one that remains stationary. When a mobile robot enters a node through a port number i, it calculates ($i\ mod\ d$) +1, where d is the degree of the node and chooses to explore using the calculated port number in the next time step [2], unless it has met the stationary robot. Gathering occurs at the node of the stationary robot.

According to [2], a robot can do perpetual exploration in any anonymous tree of maximum degree $\Delta$ using O($log\ \Delta$) memory bits. This perpetual exploration algorithm can be modified so that the mobile robots would stop at the node of the stationary robot. Thus for gathering, the memory requirement would be O($log\ \Delta$) private bits for each robot (for exploration) and a single public bit shared by all the robots.

Now that gathering is said to happen at the node of stationary robot, one question of interest would be how to differentiate between mobile and stationary robot. That is, how do mobile robots realise that they have met the stationary robot and not any other mobile robot, so that they should stop at the node. There are two different approaches to achieve this. The first approach would require each node of the tree to have a bit of memory associated with it. This bit would be set to zero at every node. This whiteboard memory allows stationary robot to set this bit at its node to one as soon as it sets the public bit to one. Mobile robots can then check this bit at a node to see if it has met stationary robot (if the value of the bit is one) or another mobile robot (if the value of the bit is zero). Once gathering occurs and the stationary robot receives help, it sets the bit at its node back to zero along with resetting the public bit. The second approach is to equip each robot with a marker called pebble. Whenever a robot sets public bit and becomes stationary, it drops its pebble. When a mobile robot sees this pebble during exploration, it can identify the node of stationary robot. Mobile robots need to check for pebble only when the public bit is set. Once the stationary robot has received help, it can pick up the pebble along with resetting the public bit. The presence of public bit is still required to avoid any synchronization problem that might result from multiple robots dropping pebble and hence creating multiple points of multiplicity.

Our current model considers the case of a single stationary robot. But this solution could be extended to serve multiple robots as well. This modification to the current problem would be helpful when there is more than one robot that senses emergency in a network and requires assistance from one or a few other robots. Whenever the public bit is set to one, any robot that requires assistance could drop its pebble. Once it gets assistance, it can pick up its pebble. Public bit would be reset only by the robot that set it, once it gets required help. Hence every robot that requires assistance need to check the public bit regularly to make sure that it is not reset by any other robot so that if at all it gets reset by the robot that set it, it could be set back to one until required assistance is received. If every robot requires assistance from only one another robot, atmost $k/2$ requests could be served when there are k robots. Whenever a stationary robot receives help from more robots than it requires and the public bit is set, it could choose to release the robots that are not required so that they could serve other locations of emergency. It could choose to keep the robots that came through lower port numbers and release the ones that came through higher port numbers (or the other way).

Gathering problem is solved in this model in situations where a certain robot (or robots in case of extended model) senses emergency and requests for additional assistance. The robot would require help as quickly as possible. It would be interesting to see how this could be done quickly. We know that the robots are performing exploration by calculating ($i \ mod \ d$) +1 at every step. Since every robot has a visibility of one, they could check for presence of pebble in its neighbouring nodes whenever public bit is set. If a neighbouring node has pebble, then the robot can decide to move to that node instead of following the result of its calculation for exploration. But in other cases, this method donot have much control over how quickly a stationary robot would receive help.

Another approach to provide quick help would be to pair each robot with another robot. Both these robots could explore the tree together staying at a distance of one so that if one of them goes down, the other could help immediately. But this would increase the number of robots that would be required for exploration. This idea could further be extended by keeping robots at a distance, say d, from each other. All the robots in the tree form a connected subgraph and they would explore the tree by moving together as a group. Every robot should have another robot at distance d from it all the time during exploration. In this case, it is assured that in d timesteps, a robot would receive help. But this would increase the time that would be required to explore the tree. Hence it is required to find a tradeoff between the time required to explore the tree and the number of robots that is needed to explore the tree. Chapter 4 below gives pseudocode for a tree exploration algorithm in which the robots form a connected subgraph and stay connected at a distance d from each other throughout exploration. Using this algorithm, whenever a robot sets public bit and stays stationary, other robots could immediately gather at the node of the stationary robot in d timesteps. Thus gathering is ensured in O(d) time. But the algorithm requires robots to start exploration from the same node. Hence it is required to place the robots initially in a single node or the robots should be placed in such a way that they form a connected subgraph, where each robot is at a maximum distance, say d, from some other robot.

### 3.2.4 Future work in proposed model

1. Does increasing the number of public bits further reduce the memory requirement per robot?
2. Can the current solution be used for anonymous graphs instead of anonymous trees?

## 3.3 Model 3

### 3.3.1 Proposed Model

Dynamic trees are those trees that change over time by the removal and addition of edges. This model looks at a special basic case of dynamic tree in which a single edge gets removed and added every k timestep. Also the model considers the case of gathering of two robots in the tree.

The tree has two subtrees that are connected by a single edge. This edge could be removed after every k timesteps and a new edge gets added ensuring that the subtrees are still connected. The new edge that is added could be same as the removed edge also. The problem requires two identical mobile robots, initially placed in two different subtrees of the dynamic tree to meet at a single node by following the same algorithm. There could be two cases in this problem. In the first case, one robot stays stationary while the other robot is mobile. In the second case, both the robots are mobile.

This model could represent the scenario in a city, if the city can be represented as per the model. The mobile agents could represent two people who would like to meet each other in the city. In cities, the traffic restrictions keeps changing. For example, a certain road maybe blocked due to some reason and a certain other route might get added instead. In such cases, where edge gets removed and new edge gets added, the proposed gathering problem solution could be used so that the two people can meet.

### 3.3.2 Existing Work

To the best of my knowledge, this model has not been studied so far for solving gathering problem. But the problem of gathering two mobile robots in anonymous trees has already been studied in [22]. They have used the idea of a distance calculator at each node of the tree using which a robot can determine the distance to the other robot. A robot makes a move to a neighbouring node only if the distance to the other robot doesn't increase as a result of the move. They have shown that this idea reduces the time required for gathering from exponential to polynomial in terms of degree of the graph.

### 3.3.3 Solving gathering problem in proposed model

In the case of a stationary robot and a mobile robot, the gathering problem can be solved by using the idea of distance calculator. The mobile robot can do exploration of the tree such that its distance to the stationary robot does not increase during any move. Once the mobile robot reaches the node of stationary robot, gathering is achieved.

In the case of two mobile robots, gathering can be achieved in a similar way as in the problem of gathering two mobile robots in a tree in [22] .

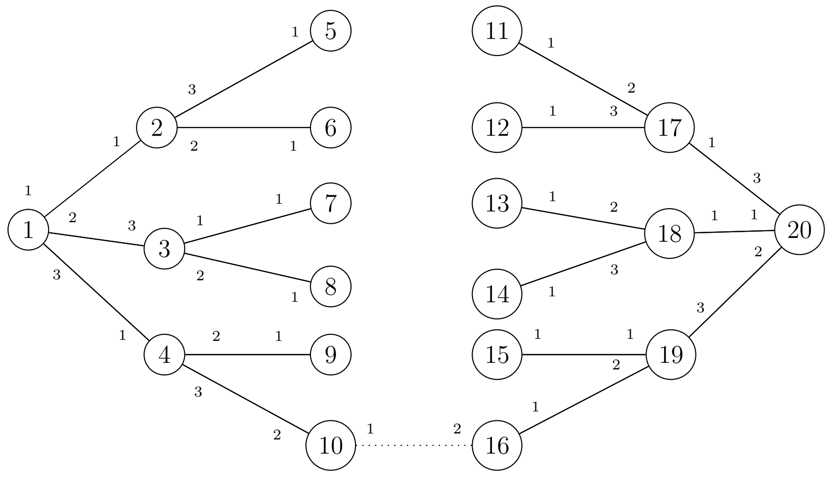

Figure 3.4: Model 3- Dynamic Tree

In the proposed model, the edge between the two subtrees are removed every k timesteps. In order to ensure gathering in all the cases, the value of k need to be one more than the length of the longest path in the subtrees. If the value of k is less than this, then even with distance calculator, the robots might not be able to gather as they might not be able to explore the other subtree. For example, consider the case of a mobile and a stationary robot in subtrees T1 and T2 respectively of the dynamic tree T. Suppose the mobile robot r is about to explore the edge connecting T1 and T2 and the edge gets removed. Now to further continue its exploration, r has to move to the node in T1 where the new edge got added so that it can enter into T2. The maximum distance that r will have to travel in T1 would be the length of the longest path in T1. Hence if the value of k is lesser, it could happen that by the time r reaches the node where the edge got added, that edge might get removed. If this continues, then gathering can never occur. The same could happen in the case of two mobile robots placed in T1 and T2. Fig. 3.4 could be used to illustrate this example (The nodes are numbered only for ease of explanation). Let there be an edge connecting node 10 and node 16. Suppose the stationary robot is at node 20 and mobile robot is at node 10. Now as the mobile robot is about to cross the edge between node 10 and node 16, that edge could get removed and suppose that the new edge gets added between node 5 and node 11. The mobile robot needs to travel the length equivalent to diameter of the subtree, which is 4 in this case, to reach the location of the newly added edge. Hence for the robot to cross that edge, the value of k in this case should be atleast 5. Otherwise it could happen that it can never reach to the other subtree and hence gathering would never occur.

One interesting and important question in this model would be the bounds on time that would be required for gathering. A walk is called a random walk when the nodes of a graph are visited in some random order. The random walk is called simple when the next node is chosen uniformly at random from the set of neighbors [23]. The cover time of a graph G is defined as the expected time taken by a simple random walk to visit all the nodes in G [23]. There are several studies that deals with cover time of graphs. [24] studies the expected time needed for a random walk on a finite graph to visit every vertex at least once. They provide an upper bound of O($n^2$) for the expectation of the cover time for regular (or nearly regular) graphs and a lower bound of $\Omega(nlogn)$ for the expected cover time for trees. [23] considered the case of simple random walks on dynamic undirected graphs with fixed underlying vertex set. They studied cover time of graphs assuming presence of adversary that modifies the graph. They had shown that there are adversary strategies that can make the expected cover time of a simple random walk on connected dynamic graphs exponential. Also they show that a random walk on any directed graph G can be simulated by a random walk on an undirected dynamic graph that is constructed from G. They also allowed the random walk to make more than a single step between each graph change. They show that exponential cover time can be obtained by allowing upto $n^{1-\epsilon}$ steps before making each change. But they have provided a strategy called lazy random walk (a walk that picks each adjacent edge with probability $1/d$, where d is the maximum degree of the graph, and with the remaining probability it stays at the current vertex) that guarantees polynomial cover time regardless of the changes made by the adversary.

In our model, we are considering the case of gathering of two robots initially placed in the two different subtrees of the dynamic tree. When we consider the case of stationary and mobile robot, we can see that cover time could be an upper bound on the expected time required for gathering. This is because the mobile robot is trying to explore the graph and find the node of stationary robot. Hence in the worst case, it would need to cover the whole graph, the expected time required for which is nothing but the cover time. Also [23] considers presence of an adversary that can change the graph which would handle all the worst case situations that could occur during gathering in our model. Hence the results from [23] provides an upper bound on the expected time required for gathering in our model in the case of a stationary and mobile robot. But in the case of two mobile robots, the expected time for gathering could go even higher, which would be interesting to study further.

### 3.3.4 Future work in proposed model

1. Provide tight bounds on the time required for gathering in the case of two mobile robots in the model.
2. Our model considers the case when a single edge gets added and removed every k timestep. It would be interesting to study the case when multiple edges are getting removed or added every k timestep and whether gathering would be possible in such situations.

# Chapter 4

# Tree Exploration Algorithm

The problem in which robots are required to fully explore an environment while maintaining connectivity was discussed in [1]. They proposed an algorithm to achieve this with a small number of robots. This rolling dispersion algorithm allows robots to disperse as much as possible while maintaining communication, and then advance as a group, leaving behind beacons to mark explored areas and provide a path back to the entrance[1]. But this algorithm has certain drawbacks when applied to trees. As per the algorithm, from the entry point robots are split across the different branches available at the entry node. The robots then continue exploration and requests for additional explorer robots whenever required. But in the case of trees, this initial split could be an overhead in most of the cases. Let the distance at which robots can communicate be 'd', 'k' be the total number of robots and 'h' be the height of the tree. When $k * d < h$, it would always be required to call additional explorers in a branch. This would require the robots at the end of the connected subgraph of robots (all robots are at distance d from some other robot and all of them together forms a connected subgraph) to travel back the distance it has explored to reach the requesting robot. To avoid these overheads, the algorithm has been modified to be suitable for exploration in trees.

The algorithm proposed here ensures that every part of the tree is explored atleast by one robot. Robots move together as a group and all the robots execute different sections of the algorithm based on their current state. Robots can have two states, that is explorer and sentry. Explorers explore the tree while remaining connected to some sentry. Sentries ensure connectivity of explorers to the subgraph connecting all the robots. The distance over which robots can communicate is 'd' and they use wireless signal intensity to ensure that they are within communication range [1].

The path that a robot follows from entry is stored by storing port numbers along the path. At the start of execution of algorithm, only one robot out of k robots at entry is chosen as explorer while others remain as sentry. Robots complete exploration of one branch at entry node before moving on to other branches. Once robots finish exploring a branch, they move back and once all of them reach the entry node, explorer robot will start exploring the next unexplored edge at the entry node while sentries would remain at the entry node until they receive request from explorer for more robots. At every step, it is made sure that all robots are connected to some other robot or else steps are taken to establish connection as soon as possible before performing any other action. Whenever robots reach a dead end, they mark the intersection to those branches with an 'explored' beacon so that the same area would not be explored again and again [1].

Each robot, based on its current state chooses certain behaviour. When a robot goes beyond its communication range from sentry, it chooses SEEK CONNECTION behaviour [1]. In this, the robot goes in reverse to see if a connection can be reestablished. If that does not work, it turns around and moves forward, changing direction occasionally until a connection with a robot is made. When an explorer chooses DISPERSE behaviour [1], it changes direction and continues moving to unexplored edge, moving away from beacons marking explored areas. This behaviour is chosen by explorers when they are at unexplored edges and also when they find 'explored' beacons. When a dead end is reached, explorer chooses RETRACT behaviour [1]. In this, it will drop a beacon 'explored' to avoid further exploration of that area. Also it informs its neighbours that it is about to move back in its path. When additional explorers are required, explorer passes EXPLORER request to its sentry and moves forward. Sentry would then pass it over to a neighbouring sentry, if one exists and moves forward towards the explorer, if the explorer is at a distance d from it. In this way it is ensured that whenever explorers move forward, the entire group of robots move forward, thus remaining connected throughout. In case during any step a multiplicity gets created, all except one among the robots at the node is chosen as explorer while others become sentries. The explorer will then move towards the unexplored edge.

Consider Fig. 4.1. There are three robots in the tree, that is R1, R2 and R3. Before start of exploration (at t=0), all the robots are at entry node, node 1. R1 is the explorer, while R2 and R3 are sentries. Let d=2 be the maximum distance between every pair of robots in the connectivity graph. R1 chooses DISPERSE behaviour at t=0 and moves forward.

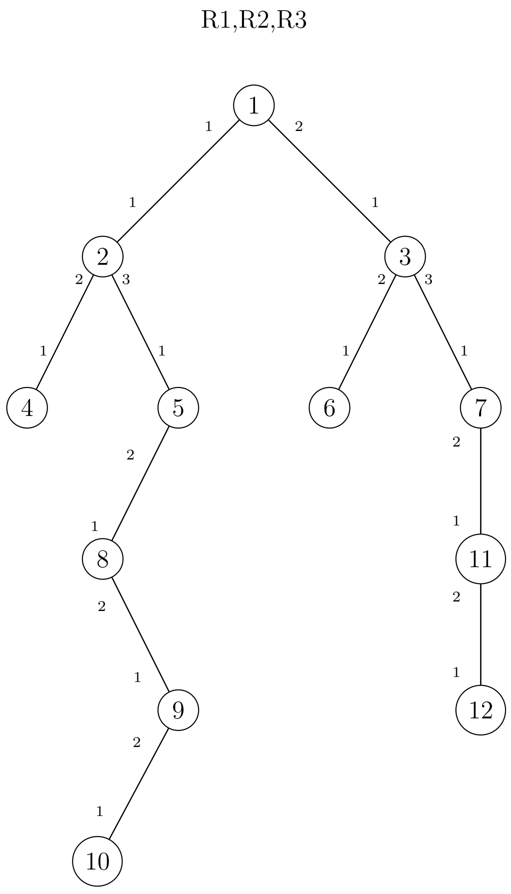

Figure 4.1: Before start of exploration(t=0), - Explorer (R1) and Sentries (R2,R3)

At t=2, R1 is at node 5, at a distance of 2 from R2 and R3 (Fig. 4.2).

R1 will send EXPLORER request to its sentry since it has not reached dead end and requires additional explorers for further exploration. Thus at t=3, when R1 moves to node 8, its sentry R2 moves to node 2 to ensure connectivity (Fig. 4.3).

At t=4, R1 again requests R2 for EXPLORER and hence when R1 moves to node 9, its sentry R2 moves to node 5. R2 passes the EXPLORER request to R3. But as R2 is still within the range of R3, R3 stays at its node (Fig. 4.4).

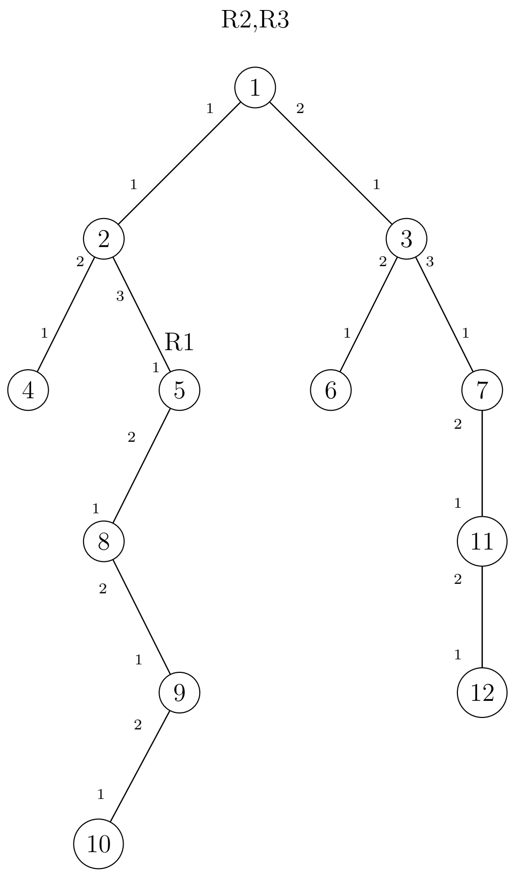

Figure 4.2: At t=2, Explorer (R1) and Sentries (R2,R3)

But at t=5, as R1 moves to node 10, R2 moves to node 8 and R3 moves to node 2 after leaving entry beacon at node 1 (Fig. 4.5).

Since R1 has reached dead end at node 10, it will choose to RETRACT. Once the robots start retracting, they will explore other unexplored paths on the way back and finally returns to entry node. Then they begin exploring other unexplored edges at the entry node.

Pseudocode of the above described modified version of Rolling Dispersion algorithm is given below.

```
1. loop
2.     Share and Update connectivity graph with neighbours
```

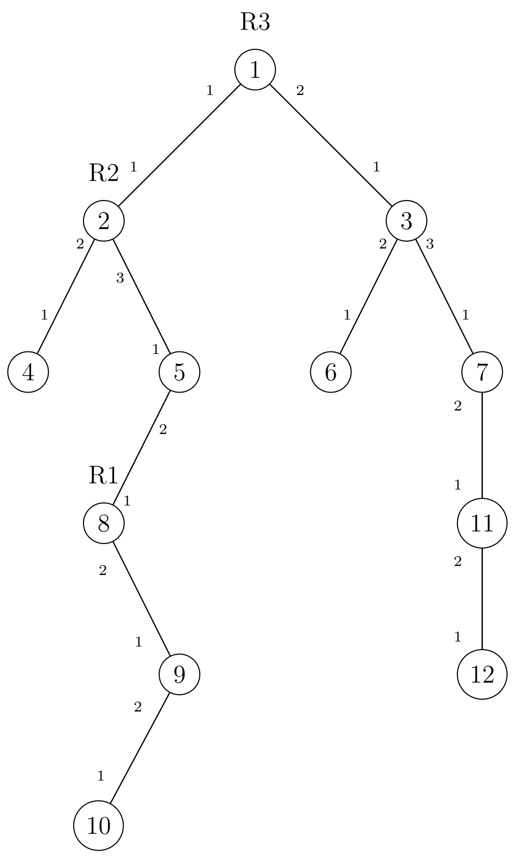

Figure 4.3: At t=3, Explorer (R1) and Sentries (R2,R3)

```
3.   If explorer and disconnected
4.       set behaviour to 'SEEK CONNECTION'
5.   else if explorer and at dead end
6.       set behaviour to 'RETRACT'
7.   else if explorer and is at distance d from sentry
8.       If not RETRACTING
9.           If only neighbour is sentry
10.              Pass EXPLORER request to sentry
11.              set behaviour to 'DISPERSE'
12.  else if explorer approaching 'explored' beacon
```

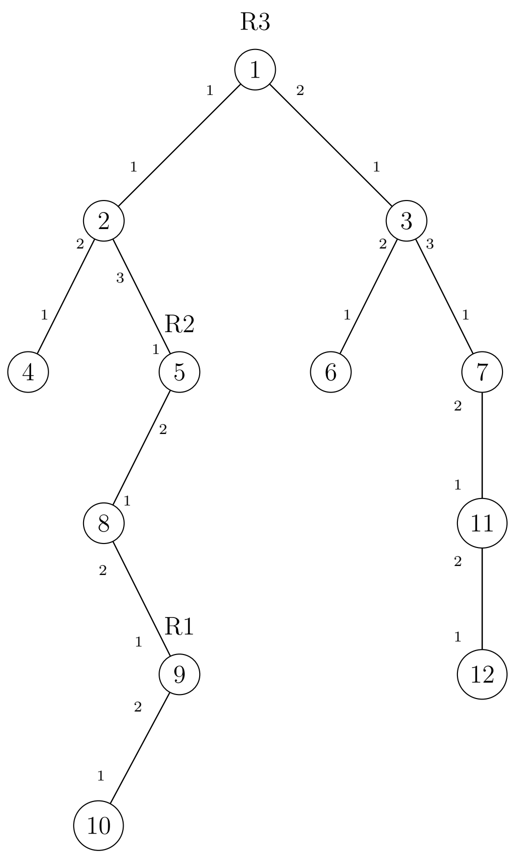

Figure 4.4: At t=4, Explorer (R1) and Sentries (R2,R3)

```
13.             turn before continuing on
14.             set behaviour to 'DISPERSE'
15.         else if explorer and unexplored edge
16.             If at entry and RETRACTING
17.                 If no other explorer at entry node
18.                     Stay at the node as explorer
19.                 else
20.                     Change status to sentry
21.                     Stay at the node
22.             else if neighbour set to RETRACT
```

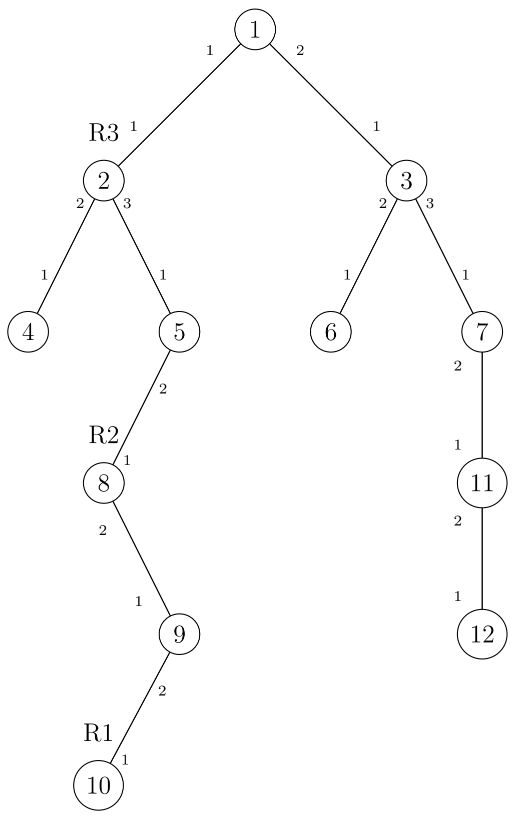

Figure 4.5: At t=5, Explorer (R1) and Sentries (R2,R3)

```
23.             If any other unexplored edge
24.                 explore to that edge
25.             else
26.                 move back along its path
27.         else
28.             set behaviour to 'DISPERSE'
29.     else if explorer and at multiplicity
30.         If you entered through the least port number
31.             set behaviour to 'DISPERSE'
32.         else
33.             change status to sentry and stay at the node
```

```
34.   else if sentry and receives EXPLORER request
35.       If it has a sentry
36.           Pass EXPLORER request to its sentry
37.           If the robot that passed the request is at distance d
38.               Move forward towards the robot that passed the request
39.       else
40.           If it is the only robot at entry
41.               If the robot that passed the request is at distance d
42.                   Drop beacon marking entry
43.                   Move forward towards the robot that passed the request
44.           else
45.               If the robot that passed the request is at distance d
46.                   Move forward towards the robot that passed the request
47.   else if sentry and neighbour set to RETRACT at distance 1
48.       If unexplored edge
49.           If at distance d from its sentry
50.               Pass EXPLORER request to sentry
51.           change status to explorer
52.           explore to that edge
53.       else if at entry
54.           Stay at the node
55.       else
56.           set to RETRACT
57.   else if sentry and RETRACTING
```

```
58.         If at entry node
59.             Stay at the node
60.     Apply chosen behaviour
```

# Chapter 5

# Conclusion

In this report, three different models that solve gathering problem of mobile robots in anonymous tree under three different scenarios have been discussed. A modified version of Rolling Dispersion Algorithm [1] for exploration of trees has also been discussed. Ideas for solving gathering problem have been proposed for each model. Analysis of these ideas opens up some interesting questions for further research so that the proposed solutions could further be improved. Model 1 tries to limit the memory that each robot requires for gathering. It would be interesting if exact bounds on these memory requirements could be determined. Model 2 reduces the private memory associated with a robot by providing a single shared bit for all the robots. The results provided for this model leaves an open problem i.e, to determine the effect of increasing the number of shared public bits on the private memory associated with the robots. Model 3 considers gathering in dynamic tree in which a single edge gets added or removed every k timestep. This model can be studied further for the case of trees in which multiple edges get removed or added every k timestep. Also all the models proposed here are for networks modelled as trees and hence one natural question of interest would be how could these ideas proposed for trees be used for general graphs.

# References

[1] E. A. Jensen and M. Gini. Rolling dispersion for robot teams 2473–2479.

[2] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory 588–597.

[3] Y. Dieudonné and A. Pelc. Deterministic gathering of anonymous agents in arbitrary networks. *arXiv preprint arXiv:1111.0321* .

[4] R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science* 390, (2008) 27–39.

[5] G. DAngelo, G. Di Stefano, and A. Navarra. Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model 197–222.

[6] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science* 337, (2005) 147–168.

[7] S. Souissi, X. Défago, and M. Yamashita. Gathering asynchronous mobile robots with inaccurate compasses 333–349.

[8] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection 542–553.

[9] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science* 410, (2009) 481–499.

[10] R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science* 411, (2010) 3235–3246.

[11] J. Czyzowicz, A. Pelc, and A. Labourel. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms (TALG)* 8, (2012) 37.

[12] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem 1181–1196.

[13] S. Guilbault and A. Pelc. Gathering asynchronous oblivious agents with local vision in regular bipartite graphs 162–173.

[14] R. Baldoni, F. Bonnet, A. Milani, and M. Raynal. Anonymous graph exploration without collision by mobile robots. *Information Processing Letters* 109, (2008) 98–103.

[15] J. Czyzowicz, D. Dereniowski, L. Gasieniec, R. Klasing, A. Kosowski, and D. Pajak. Collision-free network exploration 342–354.

[16] S. Devismes, A. Lamani, F. Petit, P. Raymond, and S. Tixeuil. Optimal grid exploration by asynchronous oblivious robots 64–76.

[17] A. Lamani, M. G. Potop-Butucaru, and S. Tixeuil. Optimal deterministic ring exploration with oblivious asynchronous robots 183–196.

[18] C. A. Duncan, S. G. Kobourov, and V. Kumar. Optimal constrained graph exploration. *ACM Transactions on Algorithms (TALG)* 2, (2006) 380–402.

[19] L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory 585–594.

[20] L. Gasieniec and T. Radzik. Memory efficient anonymous graph exploration 14–29.

[21] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science* 355, (2006) 315–326.

[22] S. Das, D. Dereniowski, A. Kosowski, and P. Uznański. Rendezvous of Distance-aware Mobile Agents in Unknown Graphs 295–310.

[23] C. Avin, M. Kouckỳ, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs) 121–132.

[24] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks. On the cover time of random walks on graphs. *Journal of Theoretical Probability* 2, (1989) 121–128.