Efficient Dispersion of Mobile Robots on Dynamic Graphs

Ajay D. Kshemkalyani*, Anisur Rahaman Molla†, Gokarna Sharma‡
*University of Illinois at Chicago, Chicago, IL 60607, USA, *ajay@uic.edu*
†Indian Statistical Institute, Kolkata, India, *molla@isical.ac.in*
‡Kent State University, Kent, OH 44242, USA, *sharma@cs.kent.edu*

***Abstract*—The dispersion problem on graphs asks $k \leq n$ robots placed initially arbitrarily on the nodes of an n-node anonymous graph to reposition autonomously to reach a configuration in which each robot is on a distinct node of the graph. This problem is of significant interest due to its relationship to other fundamental robot coordination problems, such as exploration, scattering, load balancing, and relocation of self-driving electric cars (robots) to recharge stations (nodes). The objective is to simultaneously minimize (or provide trade-off between) two fundamental performance metrics: (i) time to achieve dispersion and (ii) memory requirement at each robot. This problem has been relatively well-studied on static graphs. In this paper, we investigate it for the very first time on dynamic graphs. Particularly, we show that, even with unlimited memory at each robot and 1-neighborhood knowledge, dispersion is impossible to solve on dynamic graphs in the local communication model, where a robot can only communicate with other robots that are present at the same node. We then show that, even with unlimited memory at each robot but without 1-neighborhood knowledge, dispersion is impossible to solve in the global communication model, where a robot can communicate with any other robot in the graph possibly at different nodes. We then consider the global communication model with 1-neighborhood knowledge and establish a tight bound of $\Theta(k)$ on the time complexity of solving dispersion in any n-node arbitrary anonymous dynamic graph with $\Theta(\log k)$ bits memory at each robot. Finally, we extend the fault-free algorithm to solve dispersion for (crash) faulty robots under the global model with 1-neighborhood knowledge.**

***Index Terms*—Mobile robots, Dispersion, Dynamic graphs, Distributed algorithms, Runtime, Memory, Crash faults**

I. Introduction

The dispersion of autonomous mobile robots to spread them out evenly in a region is a problem of significant interest in distributed robotics [19, 20]. Recently, this problem has been formulated in the context of graphs as follows: *Given any arbitrary initial configuration of $k \leq n$ robots positioned on the nodes of an n-node graph, the robots reposition autonomously to reach a configuration where each robot is positioned on a distinct node of the graph* (which we call the DISPERSION problem) [2]. This problem has many practical applications, for example, in relocating self-driving electric cars (robots) to recharge stations (nodes), assuming that the cars have smart devices to communicate with each other to find a free/empty charging station [2, 22]. This problem is also important due to its relationship to many other well-studied autonomous robot coordination problems, such as exploration, scattering, and load balancing [2, 22].

The main objective in DISPERSION is to simultaneously minimize (or provide trade-off between) two fundamental performance metrics: (i) *time* to achieve dispersion and (ii) *memory requirement* at each robot. Several papers studied this problem recently on *static* anonymous trees, grids, and arbitrary graphs, [2, 22–25] which do not change over time. The following question naturally arises: *Is it possible to solve* DISPERSION *on anonymous graphs which change over time (i.e., dynamic graphs)?* In this paper, we establish under which conditions it is possible on any arbitrary dynamic graph and provide algorithms with provable time and memory bounds.

There are two communication models: local and global. In the *local communication* model [2, 22, 23], a robot at a graph node can only communicate with other robots present at the same node. In the *global communication* model [8, 16, 24, 25, 30], a robot at a graph node can communicate with any other robot in the graph, possibly at different nodes. The global model seems stronger than the local model at first sight, however many challenges that occur in the local model carryover to the global model. For example, two robots in two neighbor nodes of G do not know which edge connects them. Therefore, the robots operating following the global model still need to explore the graph as in the local model. Among the previous works [2, 22–25] on DISPERSION on static graphs, [2, 22, 23] considered the local model and [24, 25] considered the global model.

Model Summary and Contributions. We consider $k \leq n$ robots operating on an n-node dynamic graph G. The robot activation setting is *synchronous* – all robots are activated in a round and they perform their operations simultaneously in synchronized rounds. Dynamism of G is modeled in such a way that G may change in each round following the *1-interval connected dynamic graph model* of Kuhn *et al.* [26] in which the number of nodes in G do not change but the edges between the nodes may change, with the only constraint that G remains connected in each round. Therefore, for any round $r \geq 0$, we denote the graph by G_r. G has *dynamic diameter* $\widehat{D}$ which is the maximum diameter D_r among $G_r, r \geq 0$. In addition, G is *anonymous*, i.e., nodes have no (unique) IDs and hence are indistinguishable from each other but the ports (leading to incident edges) at each node are distinguishable, i.e., the ports of any node $v \in G_r$ have unique labels in the range $[1, \widehat{\delta(v)}]$, where $\widehat{\delta(v)}$ is the maximum degree (or dynamic degree) of node v among all $G_r, r \geq 0$. The robots are *distinguishable*, i.e., they have unique IDs in the range $[1, k]$. Runtime is measured in rounds and memory is measured as the number of bits stored at each robot.

We have the following four contributions (also see Table I):

- It is impossible to solve DISPERSION in an 1-interval

2575-8411/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDCS47774.2020.00100

Comm. model	Memory/robot (in bits)	1-neighborhood knowledge	DISPERSION
local	unlimited	yes	impossible (Thm. 1)
global	unlimited	no	impossible (Thm. 2)
global	$\Theta(\log k)$	yes	$\Theta(k)$-round algorithm (Thms. 3 & 4)
global, f crashes	$\Theta(\log k)$	yes	$O(k-f)$-round algorithm (Thm. 5)

TABLE I: Our results on DISPERSION for $k \leq n$ robots on an n-node 1-interval connected anonymous dynamic graph.

connected anonymous dynamic graph G in the local communication model, even with *1-neighborhood knowledge* available to robots (which neighbor nodes in G are occupied by robots and which are not) and with unlimited memory at each robot.

- It is impossible to solve DISPERSION on an 1-interval connected anonymous dynamic graph G in the global communication model, even with unlimited memory at each robot but without 1-neighborhood knowledge.
- There is a time lower bound of $\Omega(k)$ rounds for solving DISPERSION in an 1-interval connected anonymous dynamic graph G in the global communication model with 1-neighborhood knowledge.
- DISPERSION can be solved in asymptotically optimal $O(k)$ rounds in an 1-interval connected anonymous dynamic graph G in the global communication model with 1-neighborhood knowledge using $\Theta(\log k)$ bits at each robot (the memory bound of $\Omega(\log k)$ is from [2, 23]). For $f \leq k$ crash faulty robots, the runtime becomes $O(k-f)$ rounds and memory remains $\Theta(\log k)$ bits.

Challenges and Techniques. Depth first search (DFS) and breadth first search (BFS) traversals are the mostly commonly used approaches for solving DISPERSION on static anonymous graphs. These traversals generally start from a node and progressively visit nodes of the graphs (one node sequentially in DFS and nodes in a level sequentially in BFS) with building and growing DFS and BFS traversal trees over time until there is no more multiplicity node. When graph is dynamic, the DFS and BFS traversal trees cannot be grown consistently. This has impact on how to find free nodes for the robots to settle and argue the correctness of the solution.

We start from some initial configurations and show through combinatorial arguments that it is impossible to solve DISPERSION on dynamic graphs in the local communication model, even with 1-neighborhood knowledge. Additionally, we are able to show through combinatorial arguments that the global communication model also does not help. However, with 1-neighborhood knowledge, we develop a novel technique of *sliding*, through which in every round, (at least) a robot reaches an empty node (that is not previously occupied), which is enough to argue about the $O(k)$ round time complexity of the algorithm in the global model. We develop connected component, spanning tree, and disjoint path construction techniques to facilitate this sliding of robots guaranteeing the correctness and performance bounds, even when robots experience (crash) faults. The lower bounds are also established, which match (asymptotically) the bounds established for our algorithm.

Related Work. There are three previous studies [2, 22, 23] for DISPERSION in static graphs in the local model. For $k = n$, Augustine and Moses Jr. [2] proved a memory lower bound of $\Omega(\log n)$ bits at each robot and a time lower bound of $\Omega(D)$ ($\Omega(n)$ on arbitrary graphs) for any deterministic algorithm on some graph, where D is the diameter of the graph. They then provided two algorithms for arbitrary graphs for $k = n$: (i) The first algorithm with $O(mn)$ time using $O(\log n)$ bits at each robot and (ii) The second algorithm with $O(m)$ time using $O(n \log n)$ bits at each robot, where m is the number of edges in the graph. Kshemkalyani and Ali [22] provided an $\Omega(k)$ time lower bound for DISPERSION on arbitrary graphs for any $k \leq n$. They then provided three deterministic algorithms for arbitrary graphs: (i) The first algorithm with $O(m)$ time using $O(k \log \Delta)$ bits at each robot, (ii) The second algorithm with $O(\Delta^D)$ time using $O(D \log \Delta)$ bits at each robot, and (iii) The third algorithm with $O(mk)$ time using $O(\log(\max(k, \Delta)))$ bits at each robot, where Δ is the maximum degree of the graph. Recently, Kshemkalyani *et al.* [23] provided an $O(\min(m, k\Delta) \cdot \log k)$-time, $O(\log n)$-bits algorithm for arbitrary graphs, provided that the parameters m, k, Δ are known to robots beforehand.

There are two previous studies [24, 25] for DISPERSION on static graphs in the global communication model. [24] provides a deterministic algorithm for arbitrary graphs that runs in $O(\min(m, k\Delta))$ time using $O(\log(\max(k, \Delta)))$ bits at each robot, improving the time in [23] in the local model by an $O(\log k)$ factor. [25] provides a $\Theta(\sqrt{k})$ time algorithm for grids using $\Theta(\log k)$ bits at each robot (the first ever algorithm for DISPERSION that is optimal w.r.t. both memory and time). Randomized algorithms for DISPERSION are presented in [29] where the memory is reduced through randomization.

DISPERSION is closely related to the graph exploration by mobile robots. The exploration problem has been quite heavily studied for specific as well as arbitrary graphs, e.g., [4, 6, 10, 17, 28]. It was shown that a robot can explore an anonymous graph using $\Theta(D \log \Delta)$-bits memory; the runtime of the algorithm is $O(\Delta^{D+1})$ [17]. In the model where graph nodes also have memory, Cohen *et al.* [6] gave two algorithms: The first algorithm uses $O(1)$-bits at the robot and 2 bits at each node, and the second algorithm uses $O(\log \Delta)$ bits at the robot and 1 bit at each node. The runtime of both algorithms is $O(m)$ with preprocessing time of $O(mD)$. The trade-off between exploration time and number of robots is studied in [28]. The collective exploration by a team of robots is studied in [16] for trees. Another problem related to DISPERSION is the scattering of k robots in an n-node graph. This problem has been studied for rings [11, 32] and grids [5]. Recently, Poudel and Sharma [31] provided a $\Theta(\sqrt{n})$-time algorithm for uniform scattering on a grid [9]. Furthermore, DISPERSION is related to the load balancing problem, where a given load at the nodes has to be (re-)distributed among several processors (nodes). This problem has been studied quite heavily in graphs,

e.g., [7, 33]. We refer readers to [14, 15] for other recent developments in these topics.

The only existing work on DISPERSION in dynamic graphs is due to Agarwalla *et al.* [1] for dynamic rings. There is no study on DISPERSION in arbitrary dynamic graphs. The related problem exploration has been studied relatively well recently on dynamic graphs. In the work [27] the authors considered dynamic rings under the 1-interval-connected dynamic graph model (as in ours), and investigated the feasibility of their exploration. The exploration of rings in the T-interval connected dynamic graph model is studied in [21]. Exploration in the weaker model of temporal graphs (compared to the 1-interval connected dynamic graph model) is considered in [13]. Some time bounds for exploration of temporal graphs are provided in [12]. Most recently, tight runtime bounds for the exploration of 1-interval connected dynamic rings are provided in [18]. Notice that a solution to exploration is enough to solve DISPERSION but the reverse may not be true.

Roadmap. We discuss model in Section II. We present impossibility results in Section III. We then present the $\Omega(k)$-round time lower bound in Section IV and $O(k)$-round algorithm in Section VI. We then present an algorithm for crash faults in Section VII. Finally, we conclude in Section VIII.

II. MODEL DETAILS AND PRELIMINARIES

Dynamic Graph Model. We consider the synchronous dynamic network model [26] to model the dynamic graph G, in which the vertex set V is fixed, but the edges can change arbitrarily as long as the graph is connected. Time is divided into synchronous rounds. The dynamic network G is given by a sequence of undirected graphs $\langle G_0, G_1, \dots \rangle$. For any integer $r \geq 0$, we denote $G_r = (V, E_r)$ be the graph of round r, where $|V| = n$ is the number of nodes and $|E_r| = m_r$ be the number of edges in round r. We assume that the topology of the dynamic graph is controlled by a worst-case adversary which determines the dynamic graph G_r of round r with the knowledge of the algorithm and the states until round $r-1$. The adversary has complete control on the graph topology G_r of every round r as long as G_r is connected. This is a well studied dynamic graph model, known as 1-interval connected dynamic network [3, 26].

We denote the *degree* of a node $v \in G_r$ by $\delta_r(v)$. We denote the *dynamic degree* of a node $v \in G$ by $\widehat{\delta(v)}$, which is the maximum over $\delta_r(v)$ over all rounds, i.e., $\widehat{\delta(v)} = \max_{1 \leq r \leq \infty} \delta_r(v)$. Similarly, the *maximum degree* of the graph G_r is $\Delta_r = \max_{v \in V} \delta_r(v)$, which is the maximum among the degrees $\delta_r(v)$ of the nodes in G_r, and the *dynamic maximum degree* of G is $\widehat{\Delta}$ which is the maximum Δ_r over all rounds, i.e., $\widehat{\Delta} = \max_{1 \leq r \leq \infty} \Delta_r$. The *diameter* D_r of G_r is longest shortest path between any two nodes in G_r. The *dynamic diameter* $\widehat{D}$ is the maximum D_r over all rounds, i.e., $\widehat{D} = \max_{1 \leq r \leq \infty} D_r$.

G_r is unweighted, undirected and connected graph. Thus $1 \leq \widehat{\Delta} \leq n-1$ and the dynamic diameter $\widehat{D} < n$. The dynamic graph G is *anonymous*, i.e., nodes do not have identifiers. However, for any node v, its incident edges are uniquely identified by a *label* (aka port number) in the range $[1, \widehat{\delta(v)}]$. Any edge e connecting two nodes $u, v \in G_r$ has two port numbers associated with it, one at the end of e towards u and another at the end of e towards v, and there is no correlation between two port numbers of an edge. Any number of robots are allowed to move along an edge at any round although limiting it to one is sufficient in our algorithm. The nodes of G do not have memory, i.e., they cannot store information.

Robots. Let $\mathcal{R} = \{a_1, a_2, \dots, a_k\}$ be a set of $k \leq n$ robots (or agents) residing on the nodes of G_r. No robot resides on the edges of G_r, but one or more robots can occupy the same node of G_r. Each robot has a unique $\lceil \log k \rceil$-bit ID taken from $[1, k]$. When a robot moves from node u to node v in G_r, it is aware of the port of u it used to leave u and the port of v it used to enter v. Furthermore, $a_i \in \mathcal{R}$ is equipped with memory to store information.

Initially, any node $v \in G$ has no, single, or multiple robots positioned on it. If v has no robot, we call it an *empty* node, and if v has two or more robots, we call it a *multiplicity* node.

Configuration. A *configuration* $Conf_r = \{(pos_r(a_i)) : 1 \leq i \leq k\}$ specifies the robot positions on the graph nodes at any round $r \geq 0$. $Conf_0$ is the *initial* configuration. In $Conf_0$, at least a node of G is a multiplicity node. If there is exactly one multiplicity node in $Conf_0$, the $Conf_0$ is called a *rooted* initial configuration. $Conf_r$ with no multiplicity node is called a DISPERSION configuration.

Robot's Communication Model. We assume two types of communication or sensing power among robots: *global communication* and 1-*neighborhood knowledge*. In the global communication, a robot is capable to communicate with any other robot in the graph, irrespective of their positions, in a round. However, they don't have the position information (unless at the same node) as graph nodes are anonymous and there is no correlation between edge port numbers[1].

In 1-neighborhood knowledge, a robot at a node v knows the full information about the robots in the neighbor nodes $N_r(v)$, where $N_r(v)$ is the set of neighbors of $v \in G_r$. The full information includes: (i) which neighbor nodes in $N_r(v)$ are occupied by robot(s) and which are not, (ii) the ID(s) of the robots on those nodes, (iii) the robot count corresponding to each neighbor (i.e., multiplicity at each node). Essentially, a robot can sense if a (particular) neighbor has any robots or not; and using the global communication assumption, they can exchange information like robot IDs, count, etc.

Round. At any time a robot $a_i \in \mathcal{R}$ could be active or inactive. When a robot a_i becomes active, it performs the "Communicate-Compute-Move" (CCM) round as follows.

- *Communicate:* a_i can communicate with other robots $a_j \in \mathcal{R}$ (on the same node v_i of a_i or on a different node) depending on the communication model used. Robot a_i can also observe its own memory.

[1]In contrast, there is a notion of *local communication*, in which a robot can only communicate with other robots present at the same node [2]. Hence, the global communication subsumes the local communication.

- *Compute:* a_i may perform an arbitrary computation using the information observed during the "communicate" portion of that round. This includes determination of a (possibly) port to use to exit v_i, the information to store in its memory and other robots at the same node.
- *Move:* At the end of the round, a_i writes new information (if any) in its memory and exits the node to reach to a neighbor node.

Time and Memory Complexity. We consider the synchronous setting where every robot is active in every CCM round and they perform the CCM round in a synchrony. Therefore, time is measured in *rounds* or *steps*. Another parameter is memory – the number of bits stored at each robot between rounds. We assume that sufficient temporary memory is available at each robot to perform necessary computations within a round. The information in temporary memory is not carried over to the next round and hence not counted on the memory complexity.

Dispersion. DISPERSION can be formally defined as follows.

*Definition 1 (*DISPERSION*): Given an n-node 1-interval connected dynamic graph $G = (V, E)$ having $k \leq n$ mobile robots positioned initially arbitrarily on its nodes, the robots reposition autonomously to reach a configuration where each robot is on a distinct node of G.*

III. IMPOSSIBILITY RESULTS

In Section VI, we will present a deterministic algorithm that solves DISPERSION in any n-node 1-interval connected dynamic graph under global communication and 1-neighborhood knowledge. We show here the impossibility of solving DISPERSION on dynamic graphs deterministically when dropping either global communication or 1-neighborhood knowledge. Interestingly, both impossibility results hold even if the robots have unlimited memory and infinite computational power.

*Theorem 1 (***impossibility dropping global communication):** *It is impossible to solve* DISPERSION *of $k \geq 5$ mobile robots on a dynamic graph deterministically with the robots having 1-neighborhood knowledge and unlimited memory, but without global communication.*

Proof. We show that there is no deterministic algorithm solving DISPERSION of $k \geq 5$ robots. If there is an algorithm $\mathcal{A}$ solving DISPERSION on dynamic graphs, then the algorithm must work on any dynamic graph and starting from any initial configuration $Conf_0$.

Consider the following initial configuration $Conf_0$ and the dynamic graph. A path of length $(k-1)$, where each node in the path has one robot, except one end point has two robots and the other end point connects to a single node of any connected sub-graph of the remaining $(n-k+1)$ nodes. See Fig. 1 for $k = 6$ robots. Since there is no global communication (but 1-neighborhood knowledge), the robots can communicate with their immediate neighbors. Let's consider the case for $k = 6$ and Fig. 1. To achieve dispersion, the extra one robot at end node v has to find a place to be settled. The only option for the robots at node v is to move to the single neighbor u. The node u, w and x has two occupied neighbors each. The node y at the other end of the path has only one occupied neighbor x and one empty neighbor. However, this information at node y is unknown to the robots at nodes v, u, w and x. The only way to achieve dispersion from this configuration in a single round is: one robot at v moves to u, the robot at u moves to w, the robot at w moves to x, the robot at x moves to y, and y moves to an empty neighbor. However, the local view or information of the robots at the nodes w and x are identical. Thus, deterministically it is not possible for the robots at the nodes w and x to move to the same direction, towards the node y (in other words deterministically it is impossible to break the symmetry). This is because they do not agree on the port numbering of the graph. Therefore it is not possible to achieve dispersion in a single round from this configuration.

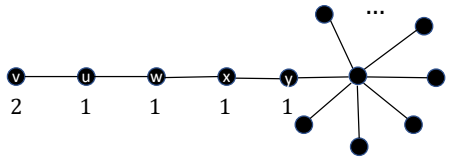


Fig. 1: The dynamic graph with the specific configuration for $k = 6$, where the node v has two robots, nodes u, w, x, y has one robot each and others nodes are empty. One can take any connected sub-graph formed by the empty nodes.

Thus it may happen that this configuration changes in the next round. We argue that the adversary can determine the dynamic graph in such a way that this specific configuration can be reached from any other configuration in one or few rounds without reaching dispersion configuration. Since the algorithm is deterministic, the adversary can connect edges among the nodes in such a way that each of the $k-2$ robots move to $k-2$ nodes, one to each, and two robots move to a single node. Then it can add edges among this $k-1$ nodes to make a $k-1$ length path and join a connected sub-graph with the remaining $n-k+1$ nodes at the end of the path. □

*Theorem 2 (***impossibility dropping 1-neighborhood knowledge):** *It is impossible to solve* DISPERSION *of $k \geq 3$ mobile robots on a dynamic graph deterministically with the robots having global communication and unlimited memory, but without 1-neighborhood knowledge.*

Proof. If there is a deterministic algorithm $\mathcal{A}$ solving DISPERSION of $k \geq 3$ robots on dynamic graphs, then the algorithm must work on any dynamic graph and starting from any initial configuration. We show that there is a configuration such that if the execution of $\mathcal{A}$ reaches to this configuration at some round r, the adversary can determine a dynamic graph such that the progress is zero in the next round $r+1$. (Note that one can simply assume such a configuration as the initial configuration $Conf_0$). Suppose the robots have unlimited memory and can communicate with any robots at any node on the graph i.e., global communication. Since the global communication doesn't give any positional information of a robot on the graph, a robot cannot determine which of it's neighbor nodes are occupied by robot(s) and which are unoccupied or empty.

Consider a configuration where $(k-2)$ robots are at $(k-2)$ nodes (each node having one robot) and one node has two

robots. Therefore $(k-1)$ nodes are occupied by robots and $(n-k+1)$ nodes are empty. Thus to achieve dispersion it remains for a single robot to find an empty node to be settled down. The adversary determines a dynamic graph for the next round in such a way that the dispersion is not possible. Form a clique with the $(k-1)$ occupied nodes, say the clique is $K_{(k-1)}$. Form any connected graph by the other $(n-k+1)$ empty nodes, say it is H. Since the algorithm is deterministic, the adversary knows which robot at a node in $K_{(k-1)}$ will move through which port of a node in the next round. Since there are $O(k^2)$ edges in the clique $K_{(k-1)}$ and only k robots on $(k-1)$ nodes, there must exist at least one edge (u,v) through which no robot moves in the next round. The adversary removes the edge (u,v) from the clique and add two edges (u,x) and (v,y) connecting any two nodes x and y in H. This makes the dynamic graph connected. More importantly, the robots at node u and v cannot differentiate between the adjacent edges which connecting nodes in $K_{(k-1)}$ and nodes in H. This completes the dynamic graph construction for the next round. From the construction it is clear that no new node is visited by the robots in the next round; hence the progress is zero.

The considered configuration might be broken at some round. But the configuration can be reached from any configuration by adding appropriate edges in one of few rounds. Then the above dynamic graph construction continues in the further rounds. Thus the dispersion is never possible. □

IV. Lower Bound

We show a time lower bound of $\Omega(k)$ rounds for DISPERSION on a dynamic tree starting from a rooted initial configuration $Conf_0$. Thus the lower bound also holds for any arbitrary dynamic graphs. The lower bound is unconditional on memory of each robot, i.e., a robot can have unlimited memory. In fact, we show the following result.

Theorem 3 ($\Omega(k)$ **time lower bound):** *Any algorithm solving* DISPERSION *of $k \leq n$ robots on an 1-interval connected dynamic (rooted) tree of n nodes requires $\Omega(k)$ rounds, even when the dynamic diameter of the tree is $D = O(1)$. The lower bound holds even if the robots have unlimited memory.*

Proof. Let us consider a *rooted* tree, i.e., the robots are placed at a single node of the tree initially. We show that one can construct a dynamic tree i.e., a sequence of dynamic trees such that in each round at most one new node will be visited by the robots. This will evident that there exists a dynamic tree, on which it requires at least k rounds to visit k (new) nodes by the robots, and hence the dispersion requires $\Omega(k)$ rounds. Moreover, the dynamic diameter of this dynamic tree is constant. Let us now explain such a dynamic tree.

Recall that the robots are placed at a single node initially. Consider the set of nodes $A_r \subseteq V$ which holds at least one robot at any round r, and the set of nodes B_r which doesn't have any robots at round r. Then $A_r \cap B_r = \emptyset$ and $A_r \cup B_r = V$ at any round r. Further notice that $|A_0| = 1$, the single node holding all the robots initially. Then the dynamic tree in round $r+1$ would be: make two star graphs (trees) T_{A_r} and T_{B_r} by the two set of nodes A_r and B_r respectively and connect the two star graphs by adding an edge between their centres, see Fig. 2. The diameter of the connected tree is 3. Then in round $r+1$, only a single node (i.e., the centre of the star graph T_{B_r}) may be visited by the robot(s). After each round, construct such two star graphs and connect them by an edge connecting the centres of the star. The trees are always connected and diameter is 3. Thus in each round at most one new node (the centre of the star T_{B_r}) may be settled by a robot. Hence the dispersion requires at least $\Omega(k)$ rounds. □

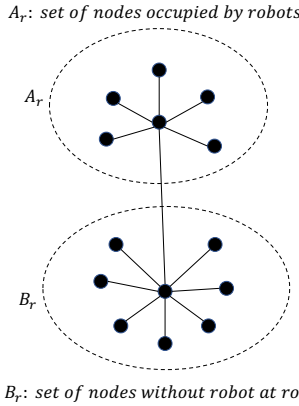


Fig. 2: The dynamic tree at any round r with diameter $D = 3$.

Remark: The above proof doesn't depend on the memory of a robot; and it also holds for randomized algorithms.

V. Connected Components, Spanning Trees, and Their Construction

We first describe how connected components are constructed and then describe the construction of spanning trees for the components. We use Fig. 3 as a running example. These concepts will be used in the algorithm in Section VI.

Connected Components. Consider a (dynamic) graph $G_r = (V, E_r)$ at round $r \geq 0$ with $k \leq n$ robots positioned arbitrarily (on one or multiple nodes of it). See Fig. 3(a) where 14 robots are placed on the nodes of a 15-node graph G_r. We give a formal definition of a connected component before discussing how it is computed by robots in round r.

*Definition 2 (***component graph):** *Consider the graph $G_r = (V, E_r)$ at round $r \geq 0$. The* component graph $CG_r = (V_r^{occupied}, E_r^{occupied})$*, where $V_r^{occupied} \subseteq V$ are the nodes of G_r which have at least a robot positioned on it at round r and $E_r^{occupied} \subseteq E_r$ are the the edges of G_r that connect any two nodes in $V_r^{occupied}$.*

CG_r may not necessarily be connected. If it is connected, then we say that CG_r is a single *connected component*, otherwise, there may be two or more *connected components* in CG_r which we write as $CG_r = \{CG_r^1, \ldots, CG_r^\beta\}$, $\beta \geq 1$. A connected component $CG_r^\phi \in CG_r$ can be defined as follows.

*Definition 3 (***connected component):** *Consider the* component graph $CG_r = (V_r^{occupied}, E_r^{occupied})$*. A* connected component $CG_r^\phi = (V_r^\phi, E_r^\phi) \subseteq CG_r$*, $\phi \leq \beta$, such that*

Algorithm 1: $ConnectedComponent(InfoPacketSet(a_i))$

$CG_r^\phi \leftarrow v_i$ (the node of G_r where a_i is currently positioned);
$N_r(v_i) \leftarrow$ the neighbors of v_i in G_r;
$N_r^{occupied}(v_i) \leftarrow$ the nodes in $N_r(v_i)$ which have at least a robot on them;
add the nodes in $N_r^{occupied}(v_i)$ and the corresponding edges with port numbers in CG_r^ϕ;
$ToBeProcessedNodeSet(a_i) \leftarrow N_r^{occupied}(v_i)$;
$AlreadyProcessedNodeSet(a_i) \leftarrow v_i$;
while $ToBeProcessedNodeSet(a_i) \neq \emptyset$ *or some node in* CG_r^ϕ *has* $N_r^{occupied}()$ *that leads to (at least) a node of* G_r *that is not already in* CG_r^ϕ
    order the nodes in $ToBeProcessedNodeSet(a_i)$ in the increasing order of their IDs;
    $v_{min} \leftarrow$ smallest ID node in $ToBeProcessedNodeSet(a_i)$;
    add the nodes in $N_r^{occupied}(v_{min})$ and the corresponding edges with port numbers in CG_r^ϕ;
    $AlreadyProcessedNodeSet(a_i) \leftarrow AlreadyProcessedNodeSet(a_i) \cup \{v_{min}\}$;
    $ToBeProcessedNodeSet(a_i) \leftarrow ToBeProcessedNodeSet(a_i) \cup N_r^{occupied}(v_{min})$;
    $ToBeProcessedNodeSet(a_i) \leftarrow ToBeProcessedNodeSet(a_i) \backslash AlreadyProcessedNodeSet(a_i)$;
return CG_r^ϕ;

Algorithm 2: $ComponentSpanningTree(CG_r^\phi)$

$v_r^\phi(mult) \leftarrow$ the multiplicity node in CG_r^ϕ; if two or more multiplicity nodes in CG_r^ϕ, $v_r^\phi(mult)$ is the smallest ID multiplicity node;
$ST_r^\phi \leftarrow v_r^\phi(mult)$;
$Stack(a_i) \leftarrow$ the neighbor nodes of $v_r^\phi(mult)$ in CG_r^ϕ pushed in the decreasing order of the port numbers at $v_r^\phi(mult)$ leading to them;
while $Stack(a_i) \neq \emptyset$
    $v \leftarrow$ the node on the top of $Stack(a_i)$;
    connect v with the node from which it was explored and add it to ST_r^ϕ;
    push the unexplored neighbors of v to $Stack(a_i)$ in the decreasing order of their port numbers at v leading to them;
return ST_r^ϕ;

(i) *for any two nodes* $u, v \in CG_r^\phi$*, there is a path following the edges in* CG_r^ϕ*, and*
(ii) *there is no edge from any node* $v \in CG_r^\phi$ *to any node* $v' \in CG_r \backslash \{CG_r^\phi\}$.

For the configuration shown in Fig. 3(a), there are two connected components CG_r^1 and CG_r^2 as shown in Fig. 3(b).

Connected Components Construction. The goal is to construct connected components $CG_r^1, \ldots, CG_r^\beta$ by the robots in each round r. Let a_i, a_j belong to the same connected component $CG_r^\phi, 1 \leq \phi \leq \beta$, at round r. Suppose a_i, a_j are positioned on nodes $v_i, v_j \in V$ at round r. We would like both a_i, a_j to construct the same CG_r^ϕ. For example, the robots $2, 4, 6, 8-11$ compute the same connected component CG_r^2 (shown in red) in Fig. 3(b); the remaining robots compute CG_r^1 (shown in green).

We describe how a_i constructs CG_r^ϕ at round r. The pseudocode is given in Algorithm 1.

We need some notations. Suppose $N_r(v_i) = \{v_{1_{v_i}}, \ldots, v_{\delta_{v_i}}\}$ be the neighbors of node $v_i \in G_r$ where a_i is currently positioned. Let $N_r^{occupied}(v_i) = \{v \in N_r(v_i)$ such that v has (at least) a robot on it$\}$.

Furthermore, let $P_r(v_i)$ be the ports at node v_i leading to the nodes in $N_r(v_i)$. Let $P_r^{occupied}(v_i) = \{p \in P_r(v_i)$ such that p leads v_i to a node in $N_r^{occupied}(v_i)\}$. With 1-neighborhood knowledge, a_i can have information about $N_r^{occupied}(v_i)$ and $P_r^{occupied}(v_i)$, i.e., a_i knows which nodes in $N_r(v_i)$ have robot(s) and which ones have not, and which ports in $P_r(v_i)$ connect v_i to those nodes.

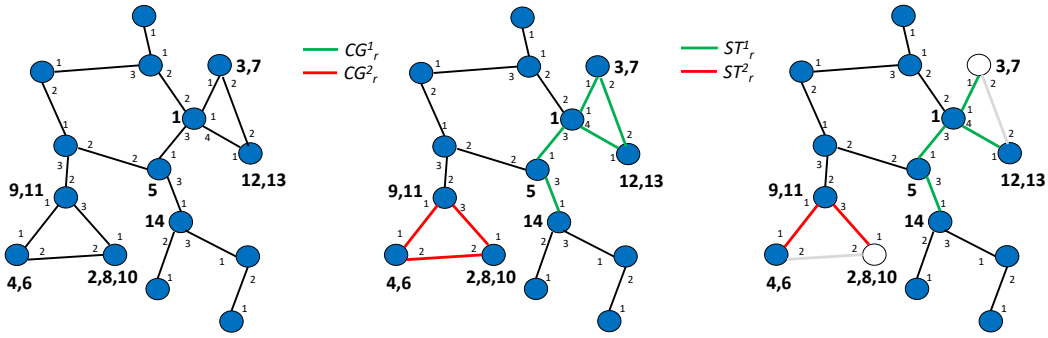
Additionally, suppose $InfoPacket_r(v_i) = \{a_i, count(a_i), N_r^{occupied}(v_i), P_r^{occupied}(v_i)\}$ is a quadruple, which we call the *information packet*, for a_i at round r, where $count(a_i) \geq 1$ is the number of robots positioned on v_i. Essentially, $InfoPacket_r(v_i)$ contains the ID of a_i, number of robots on v_i, the ids of the robots on nodes $N_r^{occupied}(v_i)$, and the port IDs of v_i from which the nodes (or robots) in $N_r^{occupied}(v_i)$ are reached from v_i. If there are multiple robots on any node in $N_r^{occupied}(v_i)$, then the smallest ID among them is used in $InfoPacket_r(v_i)$ (the robots on a node agree locally on the smallest ID robot among them to broadcast the information packet). Furthermore, since a_i uses global communication, it can broadcast $InfoPacket_r(v_i)$ to all other robots in $\mathcal{R}\backslash\{a_i\}$ and receive $InfoPacket_r(.)$s from all the robots in $\mathcal{R}\backslash\{a_i\}$.

We now describe CG_r^ϕ construction algorithm by a_i at round r (Algorithm 1). In the beginning of round r, a_i broadcasts $InfoPacket_r(v_i)$ and receives $InfoPacket_r(.)$s from other robots. In fact, if k robots are on $\alpha < k$ nodes in round r, then α nodes broadcast one information packet each and receive $\alpha - 1$ packets. Algorithm 1 starts with a_i initializing CG_r^ϕ a single node v_i (the current node position of a_i). Node v_i is also assigned the $count(a_i)$, which will be used later for identifying multiplicity at any node. a_i then updates CG_r^ϕ adding the nodes in $N_r^{occupied}(v_i)$. Node v_i is then connected to the nodes in $N_r^{occupied}(v_i)$ with the corresponding edges. The corresponding port numbers in $P_r^{occupied}(v_i)$ are also assigned to the edges. a_i now orders the nodes of $CG_r^\phi \backslash \{v_i\}$ (which are the nodes in $N_r^{occupied}(v_i)$) in the increasing order of the robot IDs. For each node $v \in N_r^{occupied}(v_i)$, a_i adds the nodes in $N_r^{occupied}(v)$ to CG_r^ϕ and the corresponding edges and ports using the information from $InfoPacket_r(v)$. Robot v also assigns $count(.)$ to node v. This process then repeats until either

i. all the $\alpha - 1$ information packets have not been processed by a_i. This condition implies that $CG_r^\phi = CG_r$, i.e., CG_r^ϕ is itself the component graph CG_r.
ii. there is some node $v' \in CG_r^\phi$ such that $N_r^{occupied}(v')$ leads to (at least) a node of G_r that is not already on CG_r^ϕ. This condition implies that there are two or more connected components in the component graph CG_r and CG_r^ϕ is one of them.

Observation 1: Each node in the connected component CG_r^ϕ *has an unique ID.*

The above observation is due to the fact that each node in CG_r^ϕ has a robot on it which supplies ID to that node. If $count(v) > 1$ for any node $v \in CG_r^\phi$, then the smallest ID robot among $count(v)$ robots provides ID to v.



(a) Graph $G_r = (V, E_r)$ at round r (b) Connected components CG_r^1 and CG_r^2 of G_r (c) Spanning trees ST_r^1, ST_r^2 of CG_r^1, CG_r^2, resp.

Fig. 3: An illustration of how connected components and spanning trees are constructed in round r for a 15-node, 17-edge dynamic graph G_r. The ◯ nodes denote the roots of the respective spanning trees. The port numbers of the edges of G_r are also shown (small text) and the 14 robots are shown near to the respective nodes (big text).

Observation 2: Consider a connected component $CG_r^\kappa \neq CG_r^\phi$ constructed by some other robot $a_l \neq a_i$. Pick any node $v \in CG_r^\phi$ and any node $w \in CG_r^\kappa$. Nodes v, w are at least 2 hops away from each other in G_r.

The above observation is due to the fact that if a node $v \in CG_r^\phi$ is not 2-hop away from any node $w \in CG_r^\kappa$, due to 1-neighborhood knowledge of v, w, CG_r^ϕ, CG_r^κ would be the same connected component when constructed (by the robots positioned on v, w).

Lemma 1: Consider the connected component CG_r^ϕ constructed by a_i in round r. Suppose $a_j \neq a_i$ is positioned at some node of CG_r^ϕ in round r. Both the robots a_i, a_j construct the same connected component CG_r^ϕ.

Proof. Due to global communication, after broadcast and receipt of information packets, each robot will have the same information. As Algorithm 1 is deterministic, two robots a_i, a_j perform the same connections using Algorithm 1 while constructing a connected component they belong to and hence they ends up forming the same connected component. □

Component Spanning Trees. Given β connected components $CG_r = \{CG_r^1, \ldots, CG_r^\beta\}$, we will construct β component spanning trees $ST_r = \{ST_r^1, \ldots, ST_r^\beta\}$, ST_r^ϕ corresponding to $CG_r^\phi, 1 \leq \phi \leq \beta$. A component spanning tree can be defined as follows.

*Definition 4 (***component spanning tree):** *Consider a connected component $CG_r^\phi = (V_r^\phi, E_r^\phi)$. A* component spanning tree *$ST_r^\phi \subseteq CG_r^\phi$ is a tree which includes all of the nodes of CG_r^ϕ.*

Since the nodes of $CG^\phi, r, 1 \leq i \leq \beta$, have unique IDs, the nodes of ST_r^ϕ also have unique IDs. Fig. 3(c) depicts the spanning trees ST_r^1, ST_r^2 constructed for CG_r^1, CG_r^2 (Fig. 3(b)).

Component Spanning Trees Construction. We discuss how a_i that is positioned on node $v_i \in CG_r^\phi$ constructs ST_r^ϕ for CG_r^ϕ. The pseudocode is given in Algorithm 2. First, a_i finds a multiplicity node in CG_r^ϕ. (This information is available due to $count(a_i)$ sent in $InfoPacket_r()$.) Let that node be $v_r^\phi(mult)$. If no node in CG_r^ϕ is a multiplicity node, then a_i does not construct ST_r^ϕ. The reason is that since no node of CG_r^ϕ has multiple robots, CG_r^ϕ is already in a DISPERSION configuration. On the other hand, if two or more nodes in CG_r^ϕ are multiplicity nodes, $v_r^\phi(mult)$ is chosen as the smallest ID node (breaking ties using IDs). Refer the roots of spanning trees (shown as ◯) in Fig. 3 which are the lowest ID multiplicity nodes in their respective components.

Robot a_i uses a simple depth-first search (DFS) approach (a breadth-first search, BFS, approach can also be used) to construct ST_r^ϕ from CG_r^ϕ. Initially, a_i sets $v_r^\phi(mult)$ the root of ST_r^ϕ. It then explores the nodes of CG_r^ϕ, starting from $v_r^\phi(mult)$, looping through the neighbors of the vertices they discover and adding each unexplored neighbor to a data structure (typically a stack) to be explored later. Robot a_i connects each vertex, other than $v_r^\phi(mult)$, to the vertex from which it was discovered. The tree construction finishes as soon as the stack becomes empty (no unexplored neighbor).

Observation 3: Each node of component spanning tree ST_r^ϕ has a unique ID and ST_r^ϕ has a distinct root node.

Lemma 2: Consider the spanning tree ST_r^ϕ for CG_r^ϕ formed by a_i in round r. Suppose $a_j \neq a_i$ is at some node of CG_r^ϕ. Both a_i, a_j construct the same spanning tree ST_r^ϕ.

Proof. If two robots $a_i, a_j \in CG_r^\phi$, then from Lemma 1, both a_i, a_j construct the same CG_r^ϕ. Furthermore, the algorithm to construct ST_r^ϕ is deterministic. Therefore, both a_i, a_j select the same node $v_r^\phi(mult) \in CG_r^\phi$ as the root of the spanning tree they construct. Now the algorithm proceeds following the same steps for both a_i, a_j and hence if an edge is added in the spanning tree formed by a_i, that would also be added in the spanning tree formed by a_j. □

VI. ALGORITHM

In this section, we present and analyze an $O(k)$-time algorithm solving DISPERSION of $k \leq n$ robots in any n-node 1-interval connected dynamic anonymous graph with global communication and 1-neighborhood knowledge. The pseudocode is given in Algorithm 4.

High Level Overview of the Algorithm. The idea is to slide the robots, in every round r, starting from the multiplicity nodes to reach the empty nodes (that were not previously

Algorithm 3: $DisjointPaths(ST_r^\phi)$

// *Each robot $a_i \in \mathcal{R}$ executes this algorithm in each round $r \geq 0$*
Input: An n-node 1-interval connected dynamic anonymous graph $G_r, r = 0$, with $k \leq n$ robots positioned on the nodes of G_r arbitrarily. Robots have 1-neighborhood knowledge and global communication.
$v_i \leftarrow$ the node of G_r where a_i is currently positioned;
Broadcast information packet $InfoPacket(v_i)$ and receive $InfoPacket()$s broadcast by the robots $\mathcal{R}\backslash\{a_i\}$;
$InfoPacketSet(a_i) \leftarrow$ all the $InfoPacket()$s received;
$CG_r^\phi \leftarrow ConnectedComponent(InfoPacketSet(a_i))$;
$ST_r^\phi \leftarrow ComponentSpanningTree(CG_r^\phi)$;
$v_r^\phi(root) \leftarrow$ the root node of ST_r^ϕ;
$LeafNodeSet(ST_r^\phi) \leftarrow$ the nodes in ST_r^ϕ which have at least an empty neighbor node, i.e., for each $v \in LeafNodeSet(ST_r^\phi)$, $N_r(v)\backslash N_r^{occupied}(v) \neq \emptyset$;
$DisjointPathSet_r^\phi(a_i) \leftarrow \emptyset$;
while $LeafNodeSet(ST_r^\phi) \neq \emptyset$
  $v_{min} \leftarrow$ the smallest ID node in $LeafNodeSet(ST_r^\phi)$;
  $RootPath_r^\phi(v_{min}) \leftarrow$ the path that connects v_{min} to $v_r^\phi(root)$;
  if $RootPath_r^\phi(v_{min})$ *shares no node and edge with any of the paths already in* $DisjointPathSet_r^\phi(a_i)$ **then**
    $DisjointPathSet_r^\phi(a_i) \leftarrow RootPath_r^\phi(v_{min})$;
  $LeafNodeSet(ST_r^\phi) \leftarrow LeafNodeSet(ST_r^\phi)\backslash\{v_{min}\}$;
return $DisjointPathSet_r^\phi(a_i)$;

Algorithm 4: $Dispersion\_Dynamic(k)$

input: An n-node 1-interval connected dynamic anonymous graph G_0 with $k \leq n$ robots initially positioned arbitrarily on the nodes of G_0.
while *k robots are on $\alpha < k$ nodes of $G_r, r \geq 0$*
  compute $DisjointPathSet_r^\phi(a_i)$ using Algorithm 3;
  $count(v_{root}) \leftarrow$ the number of robots on the root node v_{root} of ST_r^ϕ;
  if $|DisjointPathSet_r^\phi(a_i)| \geq count(v_{root})$ **then**
    order the paths in $DisjointPathSet_r^\phi(a_i)$ in the increasing order of the IDs (of the last nodes in each path) and only keep $count(v_{root}) - 1$ paths in the order in $DisjointPathSet_r^\phi(a_i)$;
  if *a_i (at node v_i) belongs to a path* $RootPath_r^\phi(v) \in DisjointPathSet_r^\phi(a_i)$ **then**
    if *a_i is not on v (the last node)* **then**
      $succ \leftarrow$ the next node in $RootPath_r^\phi(v)$ towards v;
      a_i moves to $succ$;
    else
      $succ \leftarrow$ a node in $N_r(v)\backslash N_r^{occupied}(v)$ which has the smallest port connection from v;
      a_i moves to $succ$;

occupied). Consider a path $path(v_q) = v_1, v_2, \ldots, v_{q-1}, v_q$ such that v_1 is a multiplicity node, $v_2, \ldots, v_{q-1}$ has a robot each, and v_q is an empty node. Given $path(v_q)$, sliding means that a robot from node v_i moves to node v_{i+1}, for $1 \leq i < q$. After this sliding, node v_q in $path(v_q)$ which previously was empty, now has a robot positioned on it. The algorithm finishes solving DISPERSION when there is no multiplicity node. The main challenge is how to compute such paths to perform sliding and guaranteeing that eventually a DISPERSION configuration is achieved. Particularly, robots have to agree on the paths to slide robots. This is difficult since the computation of such paths is done by robots individually based on the information they collect through global communication and 1-neighborhood knowledge. The key idea is the *disjoint path computation* framework that allows the robots to agree on the paths to slide the robots to reach at least a new empty node in every round. This altogether guarantees $O(k)$-round runtime of the algorithm, despite the graph being dynamic.

We continue to discuss disjoint paths computation and then the details of the algorithm and its analysis. We use connected components and spanning trees constructed in Section V.

Disjoint Paths. Notice that each component spanning tree $ST_r^\phi \in ST_r$ has a distinct root node. Furthermore, all the nodes in ST_r^ϕ have at least a robot positioned on it with the root $v_r^\phi(mult)$ as a multiplicity node. There is a unique path following the edges in ST_r^ϕ to reach the root $v_r^\phi(mult)$ from any node $v_{ST} \neq v_r^\phi(mult)$ of ST_r^ϕ. Let the unique path for $v_{ST} \neq v_r^\phi(mult)$ from v_{ST} to the root $v_r^\phi(mult)$ be denoted as $RootPath_r^\phi(v_{ST})$.

*Definition 5 (***disjoint path):** *Given a spanning tree ST_r^ϕ and two paths $RootPath_r^\phi(v'), RootPath_r^\phi(v'')$ from any two nodes $v', v'' \in ST_r^\phi$ (except the root $v_r^\phi(mult)$ itself), $RootPath_r^\phi(v'), RootPath_r^\phi(v'')$ are called* disjoint *if $RootPath_r^\phi(v'), RootPath_r^\phi(v'')$ share no node and edge, i.e., for any edge $e = (a, b) \in ST_r^\phi$, (i) if $e \in RootPath(v')$ then $e \notin RootPath(v'')$ (and vice versa), and (ii) if a (or b) $\in RootPath(v')$ then a (or b) $\notin RootPath(v'')$.*

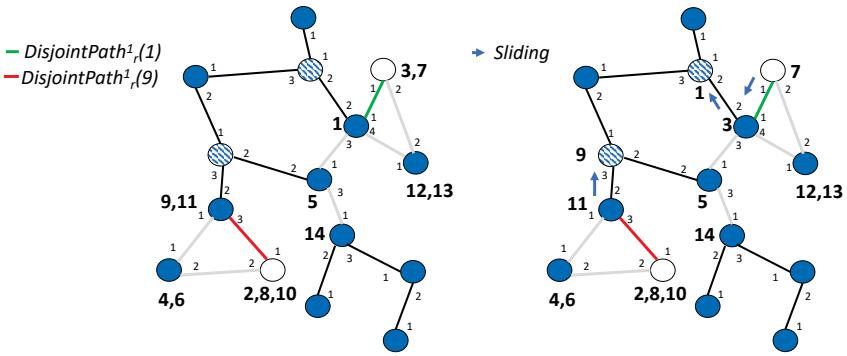
The computation of disjoint paths uses two concepts we outlined in Section V: (i) connected components CG_r^ϕ and (ii) component spanning trees ST_r^ϕ.

Disjoint Paths Computation. We are interested to compute the root paths from the all nodes in ST_r^ϕ which have at least an empty neighbor node in $N_r()$. Let the nodes in ST_r^ϕ that satisfy this empty neighbor condition be denoted as $LeafNodeSet(ST_r^\phi)$. For each node $v \in LeafNodeSet(ST_r^\phi)$, it is true that $N_r(v)\backslash N_r^{occupied}(v) \neq \emptyset$, i.e., at least a neighbor with no robot on it.

Robot $a_i \in ST_r^\phi$ constructs disjoint paths in ST_r^ϕ as follows. The pseudocode is given in Algorithm 3. a_i orders the nodes in $LeafNodeSet(ST_r^\phi)$ in the increasing order of their IDs. Let v_{min} be the smallest ID node in $LeafNodeSet(ST_r^\phi)$. Let $DisjointPathSet_r^\phi(a_i)$ denotes the set of disjoint paths computed by a_i. Initially, $DisjointPathSet_r^\phi(a_i) \leftarrow \emptyset$. a_i first computes $RootPath_r^\phi(v_{min})$ and adds in the set $DisjointPathSet_r^\phi(a_i)$. It then computes $RootPath_r^\phi()$ for the nodes in $LeafNodeSet(ST_r^\phi)$ going in the increasing order of the IDs and adds to the set $DisjointPathSet_r^\phi(a_i)$ if and only if the new root path does not share any node and edge with the path(s) already included in $DisjointPathSet_r^\phi(a_i)$. Fig. 4(a) shows the disjoint paths computed for the spanning trees of Fig. 3(c) in round r.

Lemma 3: Using Algorithm 3, there is at least a path in $DisjointPathSet_r^\phi(a_i)$, i.e, $|DisjointPathSet_r^\phi(a_i)| \geq 1$.

Proof. Assume for a sake of contradiction that $|DisjointPathSet_r^\phi(a_i)| \geq 1$ is not true. This happens only when $N_r(v)\backslash N_r^{occupied}(v) = \emptyset$, i.e., there is no neighbor of any node in ST_r^ϕ that is not occupied by a robot. If a node



(a) Disjoint paths computed on the spanning trees ST_r^1, ST_2^r of Fig. 3 in round r

(b) The robots slide on the disjoint paths to reach an empty node

Fig. 4: An illustration of how connected components and spanning trees are constructed in round r for a 15-node, 17-edge dynamic graph G_r. The ◯ nodes denote the roots of the respective spanning trees. The port numbers of the edges of G_r are also shown (small text) and the 14 robots are shown near to the respective nodes (big text).

has all neighbors occupied, it does not compute a root path. In this case, either (i) all $k = n$ robots must have already dispersed to n different nodes of G_r or (ii) for $k < n$, there is no multiplicity node in ST_r^ϕ. Since, ST_r^ϕ has a multiplicity node (the root) and $k \leq n$, there must be a node in ST_r^ϕ with $N_r(v) \backslash N_r^{occupied}(v) \neq \emptyset$, a contradiction. □

Lemma 4: Consider the paths in $DisjointPathSet_r^\phi(a_i)$ computed by a_i in the tree ST_r^ϕ. Suppose $a_j \neq a_i$ is at some node of ST_r^ϕ. $DisjointPathSet_r^\phi(a_j)$ computed by a_j are the $DisjointPathSet_r^\phi(a_i)$ computed by a_i.

Proof. From Lemma 2, the spanning trees computed by two robots a_i, a_j that are in the same component is the same. Therefore, a_i, a_j order the nodes of $LeafNodeSet()$ in the same order. Therefore, in the algorithm, if a_i has a overlap for the new path computed with the path already in $DisjointPathSet_r^\phi(a_i)$ then same situation happens for a_j between the new path computed and the path(s) already in $DisjointPathSet_r^\phi(a_j)$. □

Lemma 5: Consider any root path $RootPath_r^\phi(v) \in DisjointPathSet_r^\phi(a_i)$ at any round r. The node v is such that $N_r(v) \backslash N_r^{occupied}(v) \neq \emptyset$, i.e., there is at least a neighbor of v which is not occupied at that round.

Proof. $RootPath(.)$s are computed only for the nodes of ST_r^ϕ that belong to $LeafNodeSet(ST_r^\phi)$. For $RootPath_r^\phi(v)$ to be in $DisjointPathSet_r^\phi(a_i)$, v must be in $LeafNodeSet(ST_r^\phi)$. According to the definition of $LeafNodeSet(ST_r^\phi)$, $LeafNodeSet(ST_r^\phi)$ consists only the nodes in ST_r^ϕ which have at least a neighbor node in $N_r()$ with no robot on it, i.e., for each $w \in LeafNodeSet(ST_r^\phi)$, $N_r(w) \backslash N_r^{occupied}(w) \neq \emptyset$. □

Observation 4: Any node $v \neq v_{root} \in ST_r^\phi$ can be in at most one root path.

Detailed Description of the Algorithm. We are now ready to describe the algorithm. The pseudocode is in Algorithm 4 for any robot $a_i \in \mathcal{R}$ in any round $r \geq 0$. Algorithm 4 starts at round $r = 0$ with $k \leq n$ robots in $\mathcal{R}$ positioned arbitrarily on one or more nodes of an n-node 1-interval connected dynamic graph G_r. Each robot operates with 1-neighborhood knowledge and global communication. Algorithm 4 runs until there is no multiplicity node on G_r or equivalently a dispersion configuration is achieved. Each robot can detect the dispersion configuration due to global communication.

Algorithm 4 does the following in each round $r \geq 0$. Suppose robot a_i belongs to a connected component CG_r^ϕ. First of all, a_i runs Algorithm 3 to compute a set of disjoint paths, $DisjointPathSet_r^\phi(a_i)$, in CG_r^ϕ. Each $RootPath(v) = \{v_{root}, v_2, \ldots, v_l, v\} \in DisjointPathSet_r^\phi(a_i)$ satisfies that v_{root} is a multiplicity node and $v_2, \ldots, v_l, v$ have one or more robots, and v has (at least) a neighbor node that is empty. Moreover, all the paths in $DisjointPathSet_r^\phi(a_i)$ have one end v_{root} (the root node of ST_r^ϕ).

Let $count(v_{root})$ be the number of robots on v_{root}. Due to global communication, a_i has this information. If $|DisjointPathSet_r^\phi(a_i)| \geq count(v_{root})$, then a_i only keeps the $count(v_{root}) - 1$ paths in $DisjointPathSet_r^\phi(a_i)$ by ordering the paths in $DisjointPathSet_r^\phi(a_i)$ in the increasing order of the IDs of the leaf nodes and removing all the paths that are ordered $count(v_{root})$-th or higher. This is to make sure that one robot can be slided in each root path.

Robot a_i (at node v_i) may or may not be on a $RootPath(v)$. If it is in any $RootPath(v)$, then it cannot be in any other $RootPath(u), u \neq v$. a_i differentiates whether it is at v (the last node in $RootPath(v)$) or not. If not at v, a_i finds the next node $succ$ on $RootPath(v)$ and moves to $succ$, i.e., in round $r+1$, a_i will be positioned on node $succ$. If a_i is at v (that is, $v_i = v$), then it finds a node $succ \in N_r(v_i) \backslash N_r^{occupied}(v_i)$ such that $succ$ can be reached from v_i using the smallest port among the ports used to connect v_i to the nodes in $N_r(v_i) \backslash N_r^{occupied}(v_i)$. a_i then moves to $succ$. Concurrently with a_i, all other robots on the nodes of $RootPath(v)$ move so their $succ$ nodes. This process repeats every round until there

is no multiplicity node in graph $G_{r'}$ at some round r'. Fig. 4(b) shows how a robot is slided along the paths computed (from a multiplicity node to an empty node) in Fig. 4(a). The hashed nodes receive a robot each and the nodes in the disjoint path remain occupied. This guarantees that at least a previously-non-occupied empty node is occupied through sliding.

Analysis of the Algorithm. We analyze Algorithm 4 for correctness, runtime, and memory requirement at each robot.

Lemma 6: Algorithm 4 correctly solves DISPERSION.

Proof. We have from Lemmas 3 and 5 that there is at least a $RootPath()$ in a connected component CG_r^ϕ that leads to an empty node. Furthermore, from Lemma 4, all the robots that belong to CG_r^ϕ compute the same set of $RootPath()$s. Moreover, except the node $v_{root} \in CG_r^\phi$ in each $RootPath()$s, all other nodes of CG_r^ϕ belong to at most one root path. Therefore, robots can figure out how to move to the next node in the root path they belong to. Additionally, only at most $count(v_{root}) - 1$ robot and at least one robot moves from v_{root} at any round. When there is exactly one robot on v_{root} at some round r, it never becomes root of $ST_{r'}^\phi$ in any round $r' > r$. Therefore, the nodes of G_r which are occupied (with at least a robot) in the beginning of round r stay occupied (with at least a robot) at the end of round r and one more not-previously-occupied empty node in G_r is occupied with at least a robot. Therefore, the occupied set of nodes in G_r grow until the occupied set has size exactly k. □

The following lemma is crucial for the runtime proof.

Lemma 7: Consider any n-node dynamic graph G_r at round $r \geq 0$. If $k \leq n$ robots are positioned on m nodes of G_r in the beginning of round r, then at the end of round r, the robots are positioned on at least $m + 1$ nodes of G_r.

Proof. Consider any connected component CG_r^ϕ at any round $r \geq 0$. If there is (at least) a multiplicity node in CG_r^ϕ, since $k \leq n$ and G_r is connected, there must be a node $v \in CG_r^\phi$ which has a neighbor $v^{empty} \in N_r(v)$ that is not-previously occupied. Furthermore, v^{empty} is not in any connected component since there is a distance of at least 2-hop between the nodes of two connected components. Therefore, there is at least a root path in $DisjointPathSet_r^\phi()$ for CG_r^ϕ which can lead to v^{empty}. When robots slide on that path, then a robot will reach to v^{empty} and all other nodes on that path will also have a robot on it coming from the previous node on the path. The root node v_{root} will never be vacant since it will only slide at most $count(v_{root} - 1$ robots over $count(v_{root} - 1$ disjoint paths at round r. If $count(v_{root}) = 1$ at the end of round r, then it does not become a root of $ST_{r'}^\phi$ at any round $r' > r$. In the worst-case, only single empty node may be occupied in every round since, all robots slided from different root paths may reach that node. □

Lemma 8: A robot stores $O(\log k)$ bits in Algorithm 4.

Proof. A robot needs to differentiate itself from other robots. Since there are k robots, a robot needs to store $\log k$ bits to be able to do this. The computation within a round happens in temporary memory. With 1-neighborhood knowledge and global communication, the information needed to computing disjoint paths and sliding is available in each round. □

*Theorem 4 (***main result):** *Given $k \leq n$ robots placed initially arbitrary on the nodes of any n-node graph G_r that dynamically changes in every round $r \geq 0$ following the 1-interval connected dynamic graph model, Algorithm 4 solves* DISPERSION *in $\Theta(k)$ rounds with $\Theta(\log k)$ bit at each robot in the synchronous setting with global communication and 1-neighborhood knowledge.*

Proof. Follows immediately combining Lemmas 7 and 8. □

VII. ACCOMMODATING CRASH FAULTS

Crash fault is modeled such that a robot which has experienced (crash) fault behaves as if it has vanished from the system. Basically, it does not participate in the communication, does not move, and no robot can know where it is positioned. The non-faulty robots behave as described in Section II. The dispersion problem for crash-faulty robots, called FAULTY-DISPERSION, can be defined as follows.

*Definition 6 (*FAULTYDISPERSION*): Given an n-node 1-interval connected dynamic graph $G = (V, E)$ having $k \leq n$ mobile robots positioned initially arbitrarily on its nodes, out of which $f \leq k$ can experience crash-fault, the robots reposition autonomously to reach a configuration where each non-faulty robot is on a distinct node of G.*

We show here that Algorithm 4 can be extended to solve FAULTYDISPERSION in $O(k - f)$ rounds with $O(\log k)$ bits at each robot in the global model with 1-neighborhood knowledge, for $f \leq k$ faulty robots. We assume that a robot can crash at any time, except that if it starts moving from a node then it does not crash until it reaches the neighbor destination node. In other words, we assume that the moves are *instantaneous*.

We describe here how Algorithm 4 behaves in a round r in case of crashes. The change in the connected component construction is that CG_r^ϕ may be partitioned into multiple components due to robot crashes in round r. The overall implication is that instead of $CG_r = \{CG_r^1, \ldots, CG_r^\beta\}$ in the fault-free case, we may have $CG_r = \{CG_r^1, \ldots, CG_r^{\beta'}\}, \beta' \geq \beta$, if crashes happen before *Communicate* Phase. However, the robots can compute the (sub) connected components they belong to without any problem. Note that a connected component may be partitioned in to two or more connected sub-components and being able to compute the sub-component the robot belongs to is enough. The robots do not need to be aware of the crashed robots to achieve this. The rest of the algorithm (spanning tree construction, disjoint path, computation, and sliding) now follows the procedure as in fault-free case described in Section VI. If crash happens after *Communicate* phase, then there is no problem until disjoint paths computation. During sliding, the faulty robot does not slide to the successor node (but it receives a robot). The node that becomes empty (due to a crash robot) then behaves like an previously unoccupied empty node for round $r + 1$ and hence the algorithm can proceed as usual. The algorithm stops in

some round r' in which there is no multiplicity node (of non-faulty robots of course due to the fault model).

Theorem 5: Given $k \leq n$ robots, out of which $f \leq k$ may experience crash fault, placed initially arbitrary on the nodes of any n-node graph G_r that dynamically changes in every round $r \geq 0$ following the 1-interval connected dynamic graph model, Algorithm 4 solves DISPERSION *in $O(k-f)$ rounds with $\Theta(\log k)$ bit at each robot in the synchronous setting with global communication and 1-neighborhood knowledge.*

Proof. The memory bound of $O(\log k)$ bits follows directly from fault-free case. We also have the memory lower bound of $\Omega(\log k)$ bits from [2, 23]. This is combination gives $\Theta(\log k)$ memory bound. For the runtime, notice that if a robot crashes, then the algorithm behaves like there are only $k-1$ robots. Therefore, 1 crash-fault during the execution of the algorithm means that the algorithm needs at least one less round. Therefore, for $f \leq k$ crash-faults, The algorithm needs $O(k-f)$ rounds. □

VIII. Concluding Remarks

We have presented two impossibility results (one for local communication model with 1-neighborhood knowledge and another for global communication model without 1-neighborhood knowledge), one time lower bound of $\Omega(k)$ rounds, and a deterministic algorithm with runtime $O(k)$ rounds for fault-free case and $O(k-f)$ rounds algorithm for crash faults for solving DISPERSION of $k \leq n$ robots on n-node 1-interval connected dynamic anonymous graphs in global communication model with 1-neighborhood knowledge. The impossibility results are surprising because they show that dispersion in dynamic graphs is in sharp contrast to dispersion in static graphs – it is always solvable in static anonymous graphs. Our impossibility results use combinatorial arguments and our algorithms use the novel idea of sliding robots to occupy empty nodes. For future work, an interesting direction is to extend our algorithms for T-interval connected dynamic graphs with $T > 1$. Second interesting direction will be to consider other dynamic graph models, such as temporal and time-varying models [12, 15]. Third interesting direction will be to consider byzantine faults. Fourth interesting direction will be to extend our work to solve DISPERSION in dynamic graphs in semi-synchronous and asynchronous settings.

Acknowledgements: A. R. Molla was supported, in part, by DST Inspire Faculty research grant DST/INSPIRE/04/2015/002801, Govt. of India.

References

[1] A. Agarwalla, J. Augustine, W. K. Moses, Jr., S. K. Madhav, and A. K. Sridhar. Deterministic dispersion of mobile robots in dynamic rings. In *ICDCN*, pages 19:1–19:4, 2018.

[2] J. Augustine and W. K. Moses Jr. Dispersion of mobile robots: A study of memory-time trade-offs. In *ICDCN*, pages 1:1–1:10, 2018.

[3] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *ICALP*, pages 121–132, 2008.

[4] E. Bampas, L. Gasieniec, N. Hanusse, D. Ilcinkas, R. Klasing, and A. Kosowski. Euler tour lock-in problem in the rotor-router model: I choose pointers and you choose port numbers. In *DISC*, pages 423–435, 2009.

[5] L. Barriere, P. Flocchini, E. Mesa-Barrameda, and N. Santoro. Uniform scattering of autonomous mobile robots in a grid. In *IPDPS*, pages 1–8, 2009.

[6] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. *ACM Trans. Algorithms*, 4(4):42:1–42:18, Aug. 2008.

[7] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.*, 7(2):279–301, Oct. 1989.

[8] S. Das, D. Dereniowski, and C. Karousatou. Collaborative exploration of trees by energy-constrained mobile robots. *Theory Comput. Syst.*, 62(5):1223–1240, 2018.

[9] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. *Theor. Comput. Sci.*, 609:171–184, 2016.

[10] D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznański. Fast collaborative graph exploration. *Inf. Comput.*, 243(C):37–49, 2015.

[11] Y. Elor and A. M. Bruckstein. Uniform multi-agent deployment on a ring. *Theor. Comput. Sci.*, 412(8-10):783–795, 2011.

[12] T. Erlebach and J. T. Spooner. Faster Exploration of Degree-Bounded Temporal Graphs. In *MFCS*, volume 117, pages 36:1–36:13, 2018.

[13] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theor. Comput. Sci.*, 469:53–68, 2013.

[14] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

[15] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Mobile Entities*, volume 1 of *Theoretical Computer Science and General Issues*. Springer, 2019.

[16] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[17] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theor. Comput. Sci.*, 345(2-3):331–344, Nov. 2005.

[18] T. Gotoh, P. Flocchini, T. Masuzawa, and N. Santoro. Tight bounds on distributed exploration of temporal graphs. In *OPODIS*, pages 22:1–22:16, 2019.

[19] T. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *WAFR*, pages 77–94, 2002.

[20] T.-R. Hsiang, E. M. Arkin, M. A. Bender, S. Fekete, and J. S. B. Mitchell. Online dispersion algorithms for swarms of robots. In *SoCG*, pages 382–383, 2003.

[21] D. Ilcinkas and A. M. Wade. Exploration of the t-interval-connected dynamic graphs: the case of the ring. *Theory Comput. Syst.*, 62(5):1144–1160, 2018.

[22] A. D. Kshemkalyani and F. Ali. Efficient dispersion of mobile robots on graphs. In *ICDCN*, pages 218–227, 2019.

[23] A. D. Kshemkalyani, A. R. Molla, and G. Sharma. Fast dispersion of mobile robots on arbitrary graphs. In *ALGOSENSORS*, pages 23–40, 2019.

[24] A. D. Kshemkalyani, A. R. Molla, and G. Sharma. Dispersion of mobile robots in the global communication model. In *ICDCN*, pages 12:1–12:10, 2020.

[25] A. D. Kshemkalyani, A. R. Molla, and G. Sharma. Dispersion of mobile robots on grids. In *WALCOM*, pages 183–197, 2020.

[26] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.

[27] G. A. D. Luna, S. Dobrev, P. Flocchini, and N. Santoro. Live exploration of dynamic rings. In *ICDCS*, pages 570–579, 2016.

[28] A. Menc, D. Pajak, and P. Uznanski. Time and space optimality of rotor-router graph exploration. *Inf. Process. Lett.*, 127:17–20, 2017.

[29] A. R. Molla and W. K. M. Jr. Dispersion of mobile robots: The power of randomness. In *TAMC*, pages 481–500, 2019.

[30] C. Ortolf and C. Schindelhauer. Online multi-robot exploration of grid graphs with rectangular obstacles. In *SPAA*, pages 27–36, 2012.

[31] P. Poudel and G. Sharma. Time-optimal uniform scattering in a grid. In *ICDCN*, pages 228–237, 2019.

[32] M. Shibata, T. Mega, F. Ooshita, H. Kakugawa, and T. Masuzawa. Uniform deployment of mobile agents in asynchronous rings. In *PODC*, pages 415–424, 2016.

[33] R. Subramanian and I. D. Scherson. An analysis of diffusive load-balancing. In *SPAA*, pages 220–225, 1994.