Optimizing Robot Dispersion on Grids: with and without Fault Tolerance

Rik Banerjee [*], Manish Kumar,[†] and Anisur Rahaman Molla[‡]

Abstract

The introduction and study of dispersing mobile robots across the nodes of an anonymous graph have recently gained traction and have been explored within various graph classes and settings. While optimal dispersion solution was established for *oriented* grids [Kshemkalyani et al., WALCOM 2020], a significant unresolved question pertains to whether achieving optimal dispersion is feasible on an *unoriented* grid. This paper investigates the dispersion problem on unoriented grids, considering both non-faulty and faulty robots. The challenge posed by unoriented grids lies in the absence of a clear sense of direction for a single robot moving between nodes, as opposed to the straightforward navigation of oriented grids.

We present three deterministic algorithms tailored to our robot model. The first and second algorithms deal with the dispersion of faulty and non-faulty robots, ensuring both time and memory optimization in oriented and unoriented grids, respectively. Faulty robots that are prone to crashing at any time, causing permanent failure. In both settings, we achieve dispersion in $O(\sqrt{n})$ rounds while requiring $O(\log n)$ bits of memory per robot. The third algorithm tackles faulty robots prone to crash faults in an unoriented grid. In this scenario, our algorithm operates within $O(\sqrt{n}\log n)$ time and uses $O(\sqrt{n}\log n)$ bits of memory per robot. The robots need to know the value of n for termination.

Keywords: Mobile agents, Mobile robots, Grid graph, Mess network, Crash-fault robots, Robot's dispersion, Distributed algorithm

1 Introduction

The distribution of autonomous mobile robots for achieving coverage across an area is a highly pertinent challenge within distributed robotics, as highlighted in [5, 6]. More recently, this issue has been framed in the context of graphs in the following manner: In a scenario where k robots are initially situated on the nodes of an n-node graph, the robots undertake autonomous repositioning to achieve a final configuration wherein each robot occupies a distinct graph node (referred to as the *dispersion* problem) [1]. This problem holds practical significance across various applications, such as the repositioning of self-driving electric cars (analogous to robots) to available charging stations (equivalent to nodes). This assumption involves the cars utilizing intelligent communication methods to locate unoccupied charging stations [1, 8]. Furthermore, the problem's importance stems from its interconnectedness with numerous other extensively researched challenges in autonomous robot coordination, including exploration, scattering, and load balancing [1, 8].

The dispersion of mobile robots has garnered attention across various graph classes, including trees[1, 8], rings[1, 8, 15], arbitrary graphs [1, 11, 9, 12, 13, 16], dynamic graphs [10], directed graphs [7]. In the grid graph, the problem was explored by Kshemkalyani et al. [9], but they considered oriented grid (called planar

Algorithm	Grid Type	Faulty	Time	Memory (in bits)
Kshemkalyani et al. [9]	Oriented	No	$O(\min(k, \sqrt{n}))$	$O(\log k)$
Algorithm in Section 4	Oriented	Yes	$O(\sqrt{n})$	$O(\log n)$
Algorithm in Section 5	Unoriented	No	$O(\sqrt{n})$	$O(\log n)$
Algorithm in Section 6	Unoriented	Yes	$O(\sqrt{n}\log n)$	$O(\sqrt{n}\log n)$

Table 1: Results for the dispersion of $k \leq n$ robots and up to $f \leq n$ faulty robots on an n-node square grid.

grid) and non-faulty robots exclusively. Orientation plays a pivotal role in the symmetric graphs, Barrière et al. studied the scattering of autonomous mobile robots in the grid [2] and Becha et al. constructed a sense of direction by mobile robots in a torus [3].

Oriented vs Unoriented Grid: In an oriented grid (see Figure 1), the ports are organized in such a way that allows a single robot to traverse the grid along a path with a clear sense of direction. When a robot enters a node via an incoming port, it simply needs to select the second port (i.e., leave one port after the incoming port and select the next port) as its outgoing port to continue moving in the same direction. In Figure 1, as a robot enters from port 1 at node u, it has the sense that the straight path leads from port 3 to node v. However, this straightforward approach is not applicable in an unoriented grid where the ports are interconnected arbitrarily, as depicted in Figure 2. In such a scenario, robots cannot distinguish whether they are moving in the same direction (i.e., along a row or column) or traversing in a cycle or zigzag manner across the unoriented grid. In Figure 2, a robot enters from the port 1 at node x. It is tough to decide based on the edges which path leads in the straight direction to y. In the grid, it appears that port 4 leads to the y unlike port 3. Consequently, it's not feasible to adapt the algorithm proposed in [9] to work in an unoriented grid.
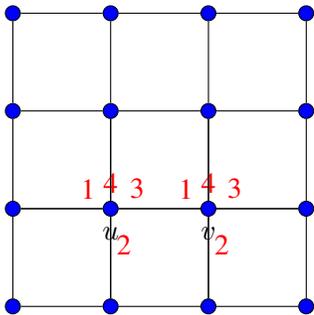

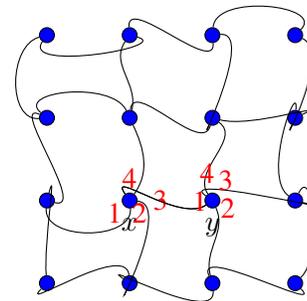Figure 1: 16 nodes oriented square grid.

Figure 2: 16 nodes unoriented square grid.

We provide three novel deterministic algorithms for dispersion on oriented and unoriented grid graphs. Our first algorithm works with faulty robots in an oriented grid graph. The second algorithm works with non-faulty robots and the third algorithm works with faulty robots in an unoriented grid graph, in which, a faulty robot can crash at any time during the execution of the protocol and never respond after crashing. Among the best-known results on dispersion [13, 4], the paper [13] considered non-faulty robots, and [4] considered faulty robots. However, applying the arbitrary graph results of [13, 4] to grid graphs yields memory-optimal solutions but not the time-optimal ones. Our results and a comparison with the closely related work [9] are given in Table 1 for reference.

Our Contributions. We present algorithms for the dispersion of k robots in any arbitrary anonymous square grid in three different models, namely, faulty robots' dispersion in oriented grid, non-faulty robots' dispersion in unoriented grid, and faulty robots' dispersion in unoriented grid. All these algorithms are deterministic. Specifically, we have the following three contributions when the square grid size, n, is known.

1. **Faulty robots' dispersion on an oriented grid:** We present dispersion of k robots in an oriented square grid of n nodes having f faulty robots such that $f \leq k \leq n$ that terminates in $O(\sqrt{n})$ rounds and uses memory bits $O(\log n)$ at each robot.

2. **Non-faulty robots' dispersion on an unoriented grid:** We propose dispersion of k robots in an unoriented square grid of n nodes such that $k \leq n$ that terminates in $O(\sqrt{n})$ rounds and uses $O(\log n)$ bits of memory at each robot.

3. **Faulty robots' dispersion on an unoriented grid:** We present dispersion of k robots in an unoriented square grid of n nodes in the presence of any number of faulty robots that terminates in $O(\sqrt{n} \log n)$ rounds and uses $O(\sqrt{n} \log n)$ bits of memory at each robot.

Notice that $\Omega(\sqrt{n})$ is the trivial lower bound for round complexity. A robot requires the diameter of the graph time (round) to reach from one end to the other, where $2\sqrt{n}$ is the diameter of the grid. Therefore, our first two results are time optimal, while the third result is time optimal up to $\log n$ factors. While $\Omega(\log k)$ is the lower bound for each robot's memory (in bits)[1].

Challenges and Techniques. Recall that robots in the oriented grid possess a sense of direction. Firstly, a robot can reach the boundary nodes (nodes with degree 3) from the internal of the grid by moving in a straight direction. But if the grid is unoriented in that case the robots have no sense of the direction, therefore, a robot can not reach the boundary in optimal time. Secondly, a robot can reach the corner of the grid by following the boundary. In this way, robots can find each other by sending a single robot to find the appropriate corner to gather at a single corner. On the other hand, if the robots are faulty then gathering at a single corner becomes challenging due to the faulty nature of the robots. The single robot can crash and corner robots might wait for an indefinite time.

We overcome these challenges as follows. In the non-faulty unoriented setup, two or more robots ensure their direction of movement in a straight line. Firstly, the group of robots (2 or more) moves in the direction of the minimum port number. Now the challenge is to ensure the movement in a straight direction. In this, all robots except one remain stationary, and one robot r_u explores the shortest possible ways to reach its initial position in the grid where the rest of the robots are placed. The robot r_u reaches the initial position in two ways, using the shortest path. These two paths lie at $180°$ to each other. This helps to determine the direction of the robots in a straight line. The detailed description is given in Section 5. A similar problem is tackled in the faulty unoriented setup by sending half of the robots to explore the other two paths to reach their initial position (detailed description is given in Section 6). In this way, robots reach the corner with the help of the boundary. Secondly, the faulty corner robots find each other by sending half of their available corner robots in one direction and reach their initial position after a round trip of the grid. By using this strategy, either half of the robots crash or find the appropriate corner (based on ID) to move. In the worst case, every time half of the corner robots may crash which causes an overhead of $O(\log n)$ round trip of the grid and there would not be left any robot to disperse. This problem is optimally tackled in the Section 4 in two different way based on the value of n, i.e., even or odd. For odd value of n all the robots gathers at the center of the grid. On the other hand, for even value of n, due to unavailability of unique center of grid, robots partition the whole grid in 4 parts and disperse accordingly. Finding an appropriate node in the unoriented grid might be another challenge after gathering at the corner in a faulty setup since a singleton robot might be left in the internal grid. Doing that effectively is also challenging, we discussed one of the approaches in Section 6.

The rest of the paper is organized as follows.

Paper Organization. Section 2 states our distributed computing model. Section 3 discusses the closely related work. Section 4 discuss the dispersion of faulty robots on oriented grid. In Section 5, the dispersion of non-faulty robots is presented on the unoriented grid. In section 6, the dispersion of faulty robots on an unoriented grid is discussed. Finally, Section 7 concludes the paper with some interesting problems.

2   Distributed Computing Model

Graph: We consider an unweighted, undirected graph $G = (V, E)$ which is a square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes embedded in 2-dimensional Euclidean plane such that $\mid V \mid = n$ and $\mid E \mid = m$, where V is the set of nodes and E is the set of edges. G is a connected graph having nodes with either degrees 2 or 3 or 4. Nodes with degrees 2, 3, and 4 are considered to be corner nodes, boundary nodes, and internal nodes, respectively. A square grid consists of 4 corner nodes, $4\sqrt{n} - 8$ boundary nodes, and $n - 4\sqrt{n} + 4$ internal nodes. These nodes are memoryless and resourceless means, unable to store any information and perform computation on them. Furthermore, nodes are anonymous such that nodes do not have IDs (identifiers) but each incident edge is uniquely identified by a labeled port number from $[1, \delta]$ where δ is the degree of the node. The nodes connected via an edge are termed neighboring nodes and are considered to be one hop away from each other. There are two port numbers assigned to any edge (e) corresponding to two nodes $u, v \in G$, that connect these two nodes. Moreover, there is no relationship between any two port numbers of an edge.

Robots: The set of robots $R = \{r_1, r_2, \ldots, r_k\}$ represents a collection of $k \leq n$ robots that are located across the graph G at one or more nodes. Robots do not stay at the edge and stay only on the nodes of the graph G. Two or more robots situated at a node are termed as co-located robots and these co-located robots can communicate with each other. This model is known as *local communication model* [1, 8]. On the other hand, if the robots are allowed to communicate with any other robots in the graph (need not be co-located), the model is known as the *global communication model* [12, 9]. However, in this paper, we consider only the local communication model. Each robot contains a unique ID consisting of $O(\log k)$ bits. A robot can move from one node to another from the port if the nodes are connected to each other via an edge. Each robot consists of some memory to store information and computation. We considered the computation time to be negligible as compared to the movement time of the robots from one node to another. A robot performs the "Communicate-Compute-Move" operation which is defined below.

Cycle: We consider a synchronous setting, every robot is synchronized to a common clock and movement from one node to another is complete in a one-time cycle or round. A robot $r_i \in R$ remains active in the "Communicate-Compute-Move" (CCM) cycle in a synchronous setting. Following are the three operations carried out by the robots:

- **Communicate:** r_i can interact with the co-located robots and view the memory of a different robot, say $r_j \in R$.

- **Compute:** r_i can perform any required computation by using the data gathered during the "communicate" phase. This involves choosing a (potentially) port to leave v_i and selecting the data to be saved in the robot r_j.

- **Move:** r_i writes new information (if any) in the memory of a robot r_j at v_i, and exits v_i using the computed port to reach to a neighbor node of v_i.

Crash Faults: We consider the crash failure setup where a robot may fail by *crashing* at any time during the execution of the algorithm. The crashed robot is not recoverable and once a robot crashes it immediately loses all the information stored in itself, as if it was not present at all. Further, a crashed robot is not visible or sensible to other robots. We assume there are at most f faulty robots such that $f \leq k$.

Time and Memory Complexity: The time complexity conveys the number of discrete rounds or cycles taken before achieving dispersion. Memory complexity is the number of bits required to store each robot to successfully execute the algorithm. Our goal is to solve dispersion as fast as possible and keep the memory

per robot as low as possible.

Below, we formally define non-faulty and faulty robots' dispersion problems in a graph. We study the problems on an oriented as well as unoriented grid graph that too with and without fault in this paper.

Definition 1 (NON-FAULTY ROBOTS DISPERSION). *Given $k \leq n$ robots, initially, placed arbitrarily over the nodes of an n-node graph, the robots re-position themselves autonomously such that each node has at most one robot on it and subsequently terminate.*

Definition 2 (FAULTY ROBOTS DISPERSION). *Given $k \leq n$ robots, up to $f \leq k$ of which are faulty, initially placed arbitrarily over the nodes of an n-node graph, the (non-faulty) robots re-position themselves autonomously such that each node has at most one non-faulty robot on it and subsequently terminate.*

3   Related Work

In this section, we discuss the work related to deterministic algorithms for the dispersion of robots.

In 2018, Moses Jr. et al. introduced the dispersion very first time [1], where they presented dispersion for several types of graphs. They showed a lower bound of $\Omega(\log n)$ for each robot's memory. Later in 2019, the lower bound was made more specific w.r.t. k (number of robots) with $\Omega(\log(\max(k, \Delta)))$ in [11] where Δ is the highest degree of the graph. They showed that for an arbitrary graph, a deterministic algorithm requires $\Omega(D)$ rounds, where D is the diameter of the graph. They also presented two techniques for the dispersion of robots on an arbitrary graph, one takes $O(\log n)$ memory and $O(mn)$ time while the other requires $O(n \log n)$ memory and $O(m)$ time.

In 2019, Kshemkalyani and Ali discussed several techniques for both synchronous and asynchronous models [8]. They used $O(k \log \Delta)$ memory and $O(\min(m, k\Delta))$ rounds to solve the dispersion problem in the synchronous model. They presented many algorithms for the asynchronous model, one of which requires $O(\Delta D)$ rounds and $O(D \log \Delta)$ memory while another requires $O(\max(\log k, \log \Delta))$ memory with $O((m - n)k)$ time complexity. In the same year 2019, Kshemkalyani et al. improved the time complexity to $O(\min(m, k\Delta) \log k)$ at the cost of $O(\log n)$ memory [11], but the knowledge of m, n, k, and Δ were required by the robots in advance. In 2020, Takahiro et al. achieved the same complexity without the knowledge of the parameter m, n, k, and Δ [16]. Later in 2021, Kshemkalyani and Sharma improved the time complexity to $O(\min(m, k\Delta))$ with the graph-specific termination of the algorithm [13].

In 2020, the dispersion problem is studied for faulty robot configurations. In [14], Molla et al., dispersed the robots in weak Byzantine settings (robots that behave randomly but cannot change their IDs) and examined the problem of unidentifiable rings. In [15] authors suggested several dispersion methods, some of which were tolerant of strong Byzantine robots (robots that behave arbitrarily and can even modify their IDs). Their algorithms are primarily based on the concept of assembling the robots at a root vertex, using them to create an isomorphic map of G, and then scattering them throughout G by a predetermined protocol. Recently, Chand et. al. [4] provided two algorithms on the mobile robots' dispersion on arbitrary graphs in the presence of crash fault robots. The first algorithm which has a rooted initial configuration has a time complexity of $O(k^2)$. On the other hand, the second algorithm which has an arbitrary initial configuration has a time complexity of $O((f + l) \cdot \min(m, k\Delta, k^2))$ when all the parameters are known to the robots, where l is the cluster of robots in initial configuration.

In the grid graph, the dispersion problem was studied by Kshemkalyani et al. in [9]. They consider an oriented (called planar) grid and only non-faulty robots. In an oriented grid a single robot can move through a path of the grid in the same direction using the ports' ordering at a node. This property makes it easier for them to solve the dispersion problem in $O(\min(k, \sqrt{n}))$ time with each robot requiring only $O(\log k)$ bits of memory in the local communication model. They also studied the problem in the global communication

model, reducing the round complexity to $O(\sqrt{k})$ rounds. We study the dispersion problem on an unoriented grid, considering both faulty and non-faulty robots in the local communication model. A quick comparison of results is provided in Table 1.

4 Dispersion of the Faulty Robots on Oriented Grid

In this section, we present a deterministic algorithm for the dispersion of faulty robots arbitrarily placed in the oriented grid, in which robots have a sense of direction. Our primary goal is to achieve the round complexity as close as possible to non-faulty robots' dispersion in the oriented square grid and keep the memory as low as possible. Specifically, we achieve the optimal round and memory complexity, i.e., $O(\sqrt{n})$ messages and $O(\log n)$ bits memory at each robot.

4.1 Algorithm

The dispersion algorithm designed for non-faulty robots [9] cannot be readily adapted for faulty robots in oriented grids. Due to faulty robots, dispersion is more challenging as compared to non-faulty robots in an oriented square grid. The main idea is to reach the corner of the grid and then disperse from there but finding a single corner, efficiently, for gathering is challenging due to the faulty nature of the robots. Therefore, based on the size of the grid robots meet in the middle or partition the whole grid into four parts and disperse. Let us call this algorithm as *Oriented Grid Dispersion with Faulty Robots*. Below, we explain the algorithm in detail by first gathering the robots at the corners and then based on the n whether odd or even, executing the protocol.

Gathering at the corners of the grid: Each robot r_u (with degree 4) initiates from the minimum port number and reaches the boundary (node with degree 3) of the grid by moving in a straight line. It takes at most $\sqrt{n}$ rounds. Robot r_u at the boundary node moves along the boundary of the grid by initiating the minimum port number and reaches the corner (node with degree 2) of the grid, which takes at most $\sqrt{n}$ rounds. In $2\sqrt{n}$ rounds, all the robots gather across the four corners. Each robot requires $O(\log n)$ bits of memory to keep the account of number of rounds that have passed and their own ID.

Dispersion for even number of node: A corner keeps $n/4$ lowest ID robots at its corner and sends the remaining to another corner from the minimum port number along the boundary. If any corner receives some extra robots from some corner then send those robots to the opposite corner. To move over the boundary of the grid and access all the corners takes $3\sqrt{n}$ rounds in the worst case. Therefore, after $5\sqrt{n}$ rounds ($2\sqrt{n}$ rounds used to reach the corner), corner with C_r robot sends $\frac{2C_r}{\sqrt{n}}$ (at most $\sqrt{n}/2$) robot in each column with the help of the boundary nodes (including itself) in the direction of the minimum port number, if available. Each robot r_u takes at most $5.5\sqrt{n}$ rounds to reach their respective column based on their IDs, lower the ID nearer the column number. In the next $\sqrt{n}/2$ rounds, each robot disperses on the grid in their respective column in the order of lower to higher ID. Therefore, it takes $6\sqrt{n}$ rounds to disperse the robots in the square grid when n is even. In this way, a corner covers at most $1/4^{th}$ grid, i.e., $\sqrt{n}/2 \times \sqrt{n}/2$ to disperse the $n/4$ robots (if available) at those nodes.

Dispersion for an odd number of node: After $2\sqrt{n}$ rounds, corner robots move to the middle of the boundary node by initiating in the direction of the minimum port number, i.e., $\lfloor \sqrt{n}/2 \rfloor$ hop away from the corner. It takes $\sqrt{n}/2$ rounds. After $2.5\sqrt{n}$ rounds ($2\sqrt{n}$ rounds used to reach the corner), robots at the middle of the boundary move $\lfloor \sqrt{n}/2 \rfloor$ hop away from the boundary node at $90°$ and reach the center of the grid in another $\sqrt{n}/2$ rounds. Since n is odd, therefore, $\sqrt{n}$ is also odd. Notice that $\sqrt{n}$ is an integer. This implies, that there exists a unique node at equal distance between any two non-diagonal corner nodes. Furthermore, if any two robots are situated in the middle of the boundary node at opposite boundary then also there exists a unique middle node between them. Therefore, all the non-faulty robots meet at the center

of the grid which is equidistant from all the corners. Henceforth, there exists a unique center node in the grid. After $3\sqrt{n}$ rounds, the centered robots collectively move to the corner of the grid as discussed above in *gathering at the corners of the grid*. Notice that it takes $\sqrt{n}$ rounds to reach the corner of the grid from the center of the grid since robots move in straight lines and reach the middle of the boundary in $\sqrt{n}/2$ rounds and further reach at the corner of the grid in next $\sqrt{n}/2$ rounds. After $4\sqrt{n}$ rounds, a single corner possesses S_r robots and sends $\frac{S_r}{\sqrt{n}}$ robot in each column with the help of the boundary nodes (including itself) in the direction of the minimum port number, if available. Each robot r_u takes at most $\sqrt{n}$ additional rounds to reach their respective corner based on their IDs, lower the ID nearer the column number. In the next $\sqrt{n}$ rounds, each robot is dispersed on the grid in their respective column in the order of lower to higher ID. Therefore, it takes overall $6\sqrt{n}$ rounds to disperse the robots in the square grid when n is odd.

Algorithm 1 ORIENTED GRID DISPERSION WITH FAULTY ROBOTS

Require: A square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes, where f robots are faulty such that $f \leq k \leq n$, k is the number of robots. The robots are distributed on the grid.
Ensure: Dispersion of the robots over the nodes.

1: Each robot r_u (with degree 4) reaches at the boundary (node with degree 3) of the grid by moving in straight line. Robot r_u placed at boundary node, reaches the corner (node with degree 4) of the grid. Takes $O(\sqrt{n})$ rounds.
2: **if** n is even **then**   takes $O(\sqrt{n})$ rounds.
3:     r_u keeps $n/4$ robot at its corner and sends the remaining robots to another corner.
4:     After $5\sqrt{n}$ rounds, corner with C_r robots sends $\frac{2C_r}{\sqrt{n}}$ robots in half ($\sqrt{n}/2$) of the column, if available.

5: **end if**
6: **if** n is odd **then**   Takes $O(\sqrt{n})$ rounds.
7:     After $2\sqrt{n}$ rounds, corner robots move to the middle of the boundary, i.e., $\lfloor \sqrt{n}/2 \rfloor$ hop away from the corner.
8:     After $2.5\sqrt{n}$ rounds, robots at the middle of the boundary move $\lfloor \sqrt{n}/2 \rfloor$ hop away from the boundary at $90°$ and reach the center of the grid.
9:     After $3\sqrt{n}$ rounds, the centered robots move to the corner of the grid.
10:     After $4\sqrt{n}$ rounds, single corner possess S_r robots and sends $\frac{S_r}{\sqrt{n}}$ robot in each column, if available.
11: **end if**
12: Each column disperses the robots across the column if available.   Takes $O(\sqrt{n})$ rounds.
13: All the robots settled at a unique node.

From the above discussion, we have the following results.

Theorem 1. *Consider any oriented square grid of n nodes having f faulty robots such that $f \leq k \leq n$ and k is the number of robots, in which, each robot has memory access $O(\log n)$ bits then DISPERSION can be solved deterministically in $O(\sqrt{n})$ rounds.*

Remark 1. *Algorithm 1 can be modified to work without the knowledge of n. After Line 1 (in at most $\sqrt{n}$ rounds), each robot r_u moves from one corner to another corner and learns the value of $\sqrt{n}$, so n. Therefore, the extra cost added to the Algorithm 1 is of $\sqrt{n}$ round which keeps the overall round complexity unchanged, $O(\sqrt{n})$.*

4.2 Extension to Rectangular Grid

We discussed the dispersion of faulty robots on the oriented square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes in Algorithm 1. The Algorithm 1 can be modified to the dispersion of the faulty robots on the rectangular grid of $n = \ell \times \frac{n}{\ell}$, where ℓ is the length and n/ℓ is the width of the rectangle such that $\ell, n/\ell > 1$. In a square grid, length and width are of the same dimension, i.e., $\sqrt{n}$. In a rectangular grid, w.l.o.g, we consider that $\ell > n/\ell$. Henceforth, any operation that takes $\sqrt{n}$ rounds in a square grid would take at most ℓ rounds in a rectangular grid in every line of the Algorithm 1. After reaching at the corner (in line 1), it takes $\ell + n/\ell$ rounds to know the dimension of the grid. A robot r_u can do this by moving from one corner to the diagonal corner. Therefore, the only change in the Algorithm 1 is in the line 4 and line 10 where $\frac{2C_r}{\ell}$ and $\frac{S_r}{\ell}$ robots are sent, respectively, across the longer column of the rectangular grid. This conveys that the rectangular grid's dispersion round and memory complexity would be $O(\ell)$ and $O(\log n)$, respectively. Since the diameter of the rectangular grid is $O(\ell)$, therefore, our algorithm is optimal w.r.t. round and memory complexity. The correctness and complexity proof directly follows from the square grid.

5 Dispersion of Non-Faulty Robots on Unoriented Grid

In this section, we present a deterministic algorithm for dispersing $k \le n$ robots initially placed arbitrarily on the nodes of an n-node unoriented square grid. The algorithm is both time and memory optimal, achieving dispersion within $O(\sqrt{n})$ rounds while utilizing $O(\log n)$ bits of memory for each robot. Note that all robots considered in this algorithm are non-faulty.

5.1 Algorithm

The dispersion algorithm designed for oriented grids [9] cannot be readily adapted for unoriented grids. For unoriented grids, one can employ the DFS (Depth First Search) traversal based algorithm [13] which works in an arbitrary graph and solves the dispersion problem in $O(\min(m, k\Delta))$ rounds, where m is the number of edges, Δ is the highest degree in the graph and $k \le n$ is the number of robots. However, this algorithm takes $O(n)$ rounds, given that $m = O(n)$ and $\Delta = 4$ in a grid graph. Therefore, the challenge lies in reducing the time complexity from $O(n)$ to $O(\sqrt{n})$.

In the unoriented grid, the main challenge is the sense of direction as compared to the oriented grid as discussed before. Thus, a single robot situated at the internal nodes cannot move on a path in the same direction of the grid. However, we show that two or more robots working together can navigate along a path in the same direction. This ability to move along a path is crucial for achieving dispersion within $O(\sqrt{n})$ rounds since the length of a path in the square grid is $\sqrt{n}$. The outline of the algorithm is, first we gather the robots at a corner node of the grid. For this, a single robot at the node remain settle at its node. The robots in a group of two or more on the internal nodes (i.e., the node with degree four) move to the boundary nodes first and then move to a corner node. Then from the four corner nodes, the robots gathered at one corner node. Suppose the robots are gathered in the top-left corner of the grid. Notice that all the k robots may not be gathered, as some singleton robots may be settled already. Then the algorithm sends some robots to each column in parallel to count the number of single robots settled at each row. The robots back and report the requirement of robots at each row to the gathered robots in the top-left corner. Then appropriate number of robots are sent to each column from the corner node. Then robots are dispersed along each row in parallel. Our approach takes $O(\sqrt{n})$ rounds to gather the non-singleton robots at a corner and another $O(\sqrt{n})$ rounds to disperse from the corner, including counting the singleton robots. We call this algorithm as *Unoriented Grid Dispersion*. Below, we explain our algorithm the Algorithm 5.1 in detail, breaking it into 3 stages.

Stage 1 - Gathering at corners of the grid: Initially, robots are arbitrarily placed in the unoriented square grid. Nodes with degree four (internal nodes) having two or more robots start moving in a straight line and

---

Algorithm 2 UNORIENTED GRID DISPERSION

Require: A square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes and $k \leq n$ robots are distributed across the grid nodes.
Ensure: Dispersion of the robots over the nodes.

    Stage 1: Gathering at corners of the grid
1: Each robot r count the round number.
2: **while** round number $< 59\sqrt{n}$ **do**
3:     **if** robot r is at the corner (node with degree 2) **then**
4:         Stay at corner.
5:     **else if** robot r is at the boundary node (node with degree 3) **then**
6:         Reach the corner node and stay there.
7:     **else if** Non-Singleton robots at the internal node **then**
8:         Move in a direction and reach the boundary of the grid.   *Movement of the internal robots is explained in the Straight line movement of the internal robots.*
9:     **else if** Singleton robot at the internal node **then**
10:         Stay there.
11:     **end if**
12: **end while**
    Stage 2: Gathering at a single corner
13: **while** round number $< 77\sqrt{n}$ **do**   *$59\sqrt{n} + 18\sqrt{n}$ rounds required for Stage 1 and Stage 2.*
14:     Robots gathered (more than one) at the corner, send the highest ID robot via edges in one direction, and find the minimum robot's ID.
15:     All the corner robots gathered at the minimum robot's ID corner.
16: **end while**
    Stage 3: Dispersion from the gathered corner node
17: **if** number of robots at the corner is not more than the boundary and corner nodes **then**
18:     Disperse the robots along the boundary node and corner.
19: **else**
20:     Send a pair of robots to each column and count the number of robots required by each column.
21:     Send the required number of robots to each column (if available) and disperse.
22: **end if**
23: All the robots settled at a unique node.

---

reach the boundary node (node with degree three) of the grid – straight line movement of the internal robots is explained later. If robot r is already at the boundary node of the grid, then r moves across the boundary node of the grid and reaches the corner (node with degree two) – the movement of the robot across the boundary nodes of the grid is explained later. Robots at the corner (node with degree two) remain at the corner. As discussed in the straight line movement of the internal robots and movement of the robot across the boundary node of the grid takes $O(\log n)$ memory (in bits).

Stage 2 - Gathering at a single corner: After $59\sqrt{n}$ rounds (shown in Lemma 2), all the internal non-singleton robots, initially, placed at the node with degree four would gather at the corner. In between, all the robots at the boundary nodes would also gather at the corner. These robots would be distributed across the four corners (gathering might not be at a single corner). Now, we would gather them at, a single corner, the corner which has a minimum ID robot. All the corners that have more than one robot would send the highest ID robot available at that node across the boundary node of the grid and reach their initial corner after exploring the rest of the three corners. Notice that the minimum ID robot does not move from its position, therefore, each moving robot gets to know about the minimum ID robot and all the robots gather at that corner. These movements of the robots take place across the boundary nodes of the grid which take $3\sqrt{n}$ rounds to reach one corner to another (discussed in Lemma 2). In the gathering at a single corner, the robot needs to keep account of the boundary nodes' traversal and the minimum ID robot's hop distance. This takes $O(\log n)$ memory (in bits).

Stage 3 - Dispersion from the single corner: In $77\sqrt{n}$ rounds (from the Lemma 2 and Lemma 3), all the internal non-singleton robots and non-internal robots on the grid gather at the single corner. If the number of robots at the corner is not more than $4\sqrt{n} - 4$ then disperse the robots along the boundary node and corner of the grid since there is no robot at the boundary nodes of the grid. On the other hand, if the number of robots at the corner is more than $4\sqrt{n} - 4$, then send a pair of robots to each column (except the boundary node of the grid) and count the number of robots required. The corner node considers the minimum port number at the boundary and other nodes in the direction as a column of the grid. The count can be done with the help of two robots by the straight movement of the robots in the column (explained in the straight line movement of the internal robots). These pairs of robots move in the column of the grid and know the number of robots required at each column. After having the round trip of their respective column, these pairs of robots consider the port number from where they entered their respective column to reach their initial corner position and move in a straight line across the boundary nodes. After knowing the required number of robots for each column, the required number of robots is sent to the corresponding column (if available) and dispersed. Notice that the first and last column requires $\sqrt{n}$ robots due to the unavailability of any robot on those columns. In this stage, pair robots keep the account of their column number and count of the required number of robots which takes $O(\log n)$ memory (in bits).

Straight Line Movement of the internal Robots: Let us consider there exists an internal node with t number of robots such that $t > 1$ and the minimum ID robot is r_m. Now, all the robots move across the minimum port number, i.e., port 1, in a round. Now we want to make sure that the robots move in a straight line, therefore, these robots should not move across the same edge that they just crossed (say, backward port). This implies three ports' choices remain in which these robots can move. Now, the maximum ID robots (say, r) take the next minimum available port and explore it till four port distances away from the r_m in all the possible combinations (by giving priority to minimum port number) which are $3 * 3 * 3 = 27$ total different nodes. This round trip would take at most $2 * 27 = 54$ rounds from one hop away from the r_m while moving one hop away from the r_m takes 2 more rounds. Therefore, the total rounds in this round trip are 56. Notice that there exist precisely two ways for r to reach the initial position (position where r_m stays) of the robots. These two positions would be at $180°$ to each other. Further, let us consider two cases: (i) one of the ports from which the r reaches r_m is the backward port, then the other port would be the desired one in a straight line. Therefore, all the robots would proceed from that port in a straight line. (ii)

None of the ports from which r reached r_m is the backward port, therefore, none of the ports among these two ports are at $180°$ of the backward port. As a consequence, one of the remaining ports is the backward port and the other one is at $180°$ of the backward port. In this way, we find the desired port to proceed in a straight line. Now, all the robots move in a straight line from the appropriate port and repeat the process till all of them reach the boundary node of the grid. Notice that some findings may take less than 56 rounds when the robot r explores the edge of the grid. Although we can consider that particular path to reach the boundary of the grid which is explored at $90°$ of our straight line direction, we would prefer to move in the straight line. Observe that it would not affect our overall round complexity but would keep the simple fact in consideration, i.e., the straight-line movement of the robots. Consequently, to move a single hop in a straight line takes at most 56 rounds. In the case of memory, the maximum ID robot keeps account of at most 28 ports to move one hop in a straight line and maximum ID and r_m, therefore, $O(\log k)$ (in bits) is used.

Movement of the robot across the boundary nodes of the grid: Let us consider a robot r is placed at the boundary node of the grid. Robot r moves one hop from the minimum port then there arise two cases that do not lead to the corner node: (i) robot r reaches the boundary node of the grid or (ii) robot r reaches the node whose degree is four. In the first case, r explores the minimum port among the rest of the ports (excluding the port due to which r reached this boundary node of the grid) and finds the boundary node of the grid. In the second case, r comes back to the boundary node explores the next available minimum port (minimum unexplored port), and finds the boundary node. Similarly, r repeatedly explores the boundary nodes and reaches the corner of the grid. Notice that the boundary nodes' degree is three and only one port leads to the non-side node of the grid, therefore, r may select at most one non-side node of the grid before finding the correct boundary node of the grid. Therefore, robot r takes at most 3 rounds to move in a specific direction towards the corner of the grid. Also, the robot keeps an account of the last two traversed ports which takes constant memory.

Lemma 2. *Any boundary node robot takes at most $3\sqrt{n}$ rounds to reach the corner, while the group of robots (non-singleton robots) placed inside the grid reach the corner of the grid in at most $59\sqrt{n}$ rounds.*

Proof. Non-singleton robots first move from the internal node to the boundary node of the grid. Further, they move from the boundary node of the grid to the corner of the grid. As we have seen in *straight line movement of the internal robots,* that to move in a straight line there are at most 56 rounds required to move one hop. Also, there exists at most $\sqrt{n}$ such hop in a straight line of the grid. Therefore, any group of robots requires $56\sqrt{n}$ rounds to move from the internal node to the boundary node of the grid.

Further, *movement of the robot on the edge of the grid* takes 3 round to move one hop in a specific direction, and there exist at most $\sqrt{n}-1$ such hop. Hence, the number of rounds required to move across the boundary nodes before reaching a corner node is at most $3\sqrt{n}$.

Therefore, overall rounds required to reach the corner node by non-singleton robots from the inside of the grid are $3\sqrt{n} + 56\sqrt{n} = 59\sqrt{n}$, i.e, $O(\sqrt{n})$ rounds. □

Lemma 3. *Gathering all the corner robots at a single corner node takes at most $18\sqrt{n}$ rounds.*

Proof. Firstly, the robots with the highest ID at their respective corner take a round of the whole grid via boundary nodes and corner edges. This takes four sides' exploration of the corner robots. It takes $3\sqrt{n}$ round for one boundary of the grid (shown in Lemma 2). Therefore, four sides take $12\sqrt{n}$ rounds. In this way, each node gets to know about the minimum ID robot present at the corner. Now, that minimum ID robot can be at most two sides ($2\sqrt{n}$ hops) away from any corner. Therefore, to reach that particular corner takes at most $6\sqrt{n}$ rounds. Hence, overall rounds required to gather at a single corner are at most $18\sqrt{n}$. □

Lemma 4. *Robots gathered at a corner node take at most $118\sqrt{n}$ rounds to disperse the robots on the square grid.*

Proof. Let us call that corner C_0 where all the robots gather. if the number of robots is not more than $4\sqrt{n} - 4$ then these robots would disperse across the boundary nodes which takes at most $12\sqrt{n}$ rounds, i.e, $O(\sqrt{n})$ rounds.

On the other hand, if the number of robots at C_0 is more than $4\sqrt{n} - 4$ then the robots in pairs explore all the columns. To reach a particular column take $3\sqrt{n}$ (due to the movement of the robot across the boundary node of the grid) and then to take the round trip of a particular column take $2 * 56\sqrt{n}$ rounds. After that reporting about the required number of robots at a particular column to C_0 takes at most $3\sqrt{n}$ rounds. Now these robots reach a particular column in at most $3\sqrt{n}$ rounds and settle at their respective node in $56\sqrt{n}$. This implies the total number of rounds is $(3 + 112 + 3)\sqrt{n} = 118\sqrt{n}$. $\qquad\square$

Lemma 5. *The Algorithm 5.1 takes $195\sqrt{n}$ rounds and $O(\log n)$ bits of memory at each robot.*

Proof. From reaching the corner, gathering at a single corner, and dispersing all the robots in Lemma 2, Lemma 3 and Lemma 4, respectively, take $59\sqrt{n} + 18\sqrt{n} + 118\sqrt{n} = 195\sqrt{n}$ rounds. Similarly, Stages $1, 2$, and 3 show total memory required is $O(\log n)$ bits. $\qquad\square$

From the above discussion, we conclude the following results.

Theorem 6. *Consider any unoriented square grid of n nodes having k robots such that $k \leq n$ where each robot has the memory $O(\log n)$ (in bits) then DISPERSION can be solved deterministically in $O(\sqrt{n})$ rounds.*

5.2 Extension to Rectangular Grid

We discussed the dispersion of non-faulty robots on the square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes in Algorithm 5.1. The Algorithm 5.1 can be modified to the dispersion of the non-faulty robots on the rectangular grid of $n = \ell \times \frac{n}{\ell}$, where ℓ is the length and n/ℓ is the width of the rectangle, such that $\ell, n/\ell > 2$. For $n/\ell = 2$, a singleton robot can not decide the direction of the movement on the boundary since all the neighboring nodes have the same degree 3. In a square grid, length and width are of the same dimension, i.e., $\sqrt{n}$. In a rectangular grid, w.l.o.g, we consider that $\ell > n/\ell$. Henceforth, any operation that takes $\sqrt{n}$ rounds in a square grid would take at most ℓ rounds in a rectangular grid. Therefore, we can modify the Algorithm 5.1 for the rectangular grid by replacing the $\sqrt{n}$ to ℓ. This conveys that the rectangular grid's dispersion round and memory complexity would be $O(\ell)$ and $O(\log n)$. The farthest distance between two nodes in a rectangle is $\ell + n/\ell$ hopes, i.e., the diameter of the rectangle. Hence, our modified algorithm for rectangular grids has optimal round complexity and memory complexity. The correctness and complexity proof directly follows from the square grid. For termination, knowledge of ℓ is required.

Remark 2. *For $n/\ell = 2$, the dispersion can be solved in a rectangular grid if the values of n and ℓ are known. In that case, each boundary robot settles at the empty node, and if there is no empty node then movement in a straight line is executed similar to the movement on the internal node, to break the symmetry.*

6 Dispersion of Faulty Robots on Unoriented Grids

In this section, we present a deterministic algorithm for the dispersion of faulty robots, arbitrarily placed on the nodes of a unoriented grid. The algorithm achieves dispersion within $O(\sqrt{n}\log n)$ rounds while utilizing $O(\sqrt{n}\log n)$ bits of memory for each robot.

6.1 Algorithm

In the presence of faulty robots, dispersion becomes more challenging. We can not adopt the procedure discussed in Section 5. The reason behind this is: Firstly, in the case of straight-line movement of the internal robots, there might be the case that there are $O(n)$ robots present at the internal node. And after constant rounds, the highest ID robot crash before deciding the direction of movement. This may happen $O(n)$ times, therefore, the round complexity's order becomes linear. Secondly, after reaching the corner, it would be costlier (in terms of rounds) to find the minimum ID robot at the corner of the grid. Since there might be the case that every robot that goes for the round trip via boundary nodes of the grid crashes every time. In the worst case, there might be $O(n)$ robots at the corner and each failure would cost $\Omega(\sqrt{n})$ rounds. Therefore, their consecutive failure would cost sub-quadratic round complexity. Thirdly, after gathering at a single corner (non-singleton internal robots and non-internal robots), it would be costlier to send a pair of robots to the columns and count the required number of robots in each column. If $O(n)$ robots are gathered at a single corner then sending them in pair and waiting for the $O(\sqrt{n})$ rounds would further take $O(\sqrt{n})$ rounds extra, in worst case. Each round would be $O(\sqrt{n})$ rounds costly. Consequently, round complexity would be linear.

Our approach for the faulty robots' dispersion is similar to the approach, in 3 stages, of the non-faulty robots' dispersion. We deal with the above-discussed challenges in detail and improve our algorithm stage-by-stage.

Stage 1 - Gathering at corners of the grid: Initially, robots are arbitrarily placed in the unoriented square grid. Node with degree four (internal node) having two or more robots start moving in a straight line and reach the boundary node (node with degree three) of the grid – straight line movement of the internal faulty robots is explained later. If the robot r is already at the boundary node of the grid, then r moves across the boundary edge of the grid and reaches the corner (node with degree two) as explained in the movement of the robot across the boundary edges of the grid (see Section 5). During the exploration of the boundary nodes and reaching the corner, faulty robots might crash and not reach the corner, unlike non-faulty robots. Robots at the corner (node with degree two) remain at the corner.

Stage 2 - Gathering at a single corner: After $56 \log n + 59\sqrt{n}$ rounds (shown in Lemma 7), all the internal non-singleton non-crashed robots, initially, placed at the node with degree four would gather at the corner. In between, all the robots at the boundary of the grid would also gather at the corner. These robots would be distributed across the four corners (gathering might not be at a single corner). After that, robots gather at a single corner that has a minimum ID robot known as whether crashed or non-crashed. All the corners which have more than two robots would send half of the higher ID robots (say, seeker robots) available at that node across the boundary edge of the grid from the minimum port number and reach their initial corner after exploring the rest of the three corners. The seeker robots and non-seeker robots of a corner node move at the corner which has/had the minimum ID robot according to both seeker and non-seeker robots. There might be a case that either all the seeker robots or non-seeker robots crashed, in that case, the remaining robots of a corner repeat the process to find the minimum ID robot on the boundary of the grid. This would be repeated until the seeker robots sent from the minimum port numbers meet the non-seeker robots after a round trip with the same known minimum ID robot, or at most one robot is remaining at the corner. These movements of the robots take place across the boundary edges of the grid. The boundary edge robots of the grid take $3\sqrt{n}$ rounds to reach the corner and there would be $O(\log n)$ round trip in worst case. Therefore, round complexity for this stage is $O(\sqrt{n} \log n)$ (as shown in Lemma 8).

Stage 3 - Dispersion from the single corner: In $12\sqrt{n} \log n + 65\sqrt{n} + 56 \log n$ rounds (from Lemma 7, and Lemma 8), all the internal non-singleton robots and non-internal robots on the grid would gather at the single corner. Let us consider C_{r_m} as the number of robots at the single corner and R_0 as the number of columns requiring no robots. Initially, R_0 is 0. If the number of robots at the corner is not more than

---

Algorithm 3 UNORIENTED GRID DISPERSION WITH FAULTY ROBOTS

Require: A square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes and k is the number of robots. The robots are distributed in the grid.

Ensure: Dispersion of the robots over the nodes.

    Stage 1 - Gathering at corners of the grid:
1: Each robot r count the round number (say $Counter_r$).
2: Initially, $Counter_r = 0$.
3: **while** $Counter_r < (56 \log n + 59\sqrt{n})$ **do**
4:     **if** robot r is at the corner of the grid (node with degree 2) **then**
5:         Stay at corner.
6:     **else if** robot r is at node having degree 3 **then**
7:         Reach the corner node and stay there.
8:     **else if** Non-Singleton robots at the node having degree 4 **then**
9:         Move in a straight direction and reach the boundary of the grid.   Movement of the robots is explained in the *Straight Line Movement of the Faulty Robots.*
10:     **else if** Singleton robot r at the node with degree 4 **then**
11:         Stay there.
12:     **end if**
13: **end while**

    Stage 2 - Gathering at a single corner:
14: Robots gathered (more than one) at the corner (say, C_r robots).
15: **for** $12\sqrt{n} \log n + 6\sqrt{n}$ rounds **do**
16:     C_r robots send $\lceil C_r/2 \rceil$ higher ID robot (say, seeker robots) via boundary edges in minimum port direction to find the corner of minimum ID robot.
17:     **if** seeker and non-seeker robots meet with the same minimum robot's ID after round trip **then**
18:         All the corner robots (seeker and non-seeker) gathered at the minimum robot's ID corner.
19:     **else**
20:         Update C_r.
21:     **end if**
22: **end for**

    Stage 3 - Dispersion from the single corner:
23: Let the number of robots at the corner be C_{r_m}.
24: Consider the number of columns which does not require any robot as R_0. Initially, $R_0 = 0$.
25: **for** $236\sqrt{n} \log n$ rounds **do**   See in Lemma 10.
26:     **if** $C_{r_m} \leq$ number of boundary and corner nodes **then**
27:         Disperse the robots along the boundary and corner of the grid based on robots' ID.
28:         Exit the For Loop.
29:     **else**
30:         Send $\left\lfloor \frac{C_{r_m}}{\sqrt{n}-R_0-2} \right\rfloor$ robots to a column that requires robots (except the boundary edges of the grid) and disperse them in that column.
31:         The remaining robots report at the corner about the column which does not require robots.
32:         Update C_{r_m} and R_0.
33:     **end if**
34: **end for**
35: All the robots settled at a unique node.

$4\sqrt{n} - 4$ then disperse the robots along the boundary edges and corner of the grid based on their ID (smaller the ID nearer the position) since there is no robot on the boundary nodes of the grid. On the other hand, if the number of robots at the corner is more than $4\sqrt{n} - 4$, then send $\left\lfloor \frac{C_{r_m}}{\sqrt{n} - R_0 - 2} \right\rfloor$ robots to a column that requires robots (except the boundary edges of the grid) and disperse them in that column. The remaining robots report at the corner about the column which does not require robots and update C_{r_m} and R_0. This process is repeated until less than $4\sqrt{n} - 4$ robots are left, which would be dispersed across the boundary edges of the grid.

Straight Line Movement for the internal Faulty Robots: We traverse similarly with the faulty robots as we did with the non-faulty robots. As discussed earlier there might be the case that maximum ID robots might crash every time before reporting the appropriate port number (port in the same direction at 180°) for the traversal. It might increase the round complexity up to $O(n)$. Therefore, we consider half of the robots for finding the appropriate port number as compared to the maximum ID robot, and the other half stay at their place, waiting for the appropriate port number. If only one of the robots survived to report the appropriate port number then we know the appropriate port. Even if all the robots of a group crashed, then at most half of the robots would be left. Further, the remaining half of the robots partition themselves into two groups and repeat the process. Therefore, without moving on the right port (appropriate direction) these robots might crash $O(\log n)$ times, otherwise they will move in the appropriate direction (in a straight line). Consequently, round complexity would be $O(\sqrt{n} + \log n) = O(\sqrt{n})$. In other words, let us consider there exists an internal node with t number of robots such that $t > 1$ and the minimum ID robot known is r_m. Now, all the robots move across the minimum port number, i.e., port 1, and take one round. Now we want to make sure that the robots move in a straight line, therefore, these robots should not move across the same port which they just crossed (say, backward port). This implies there are three ports' choices remaining in which these robots can move. Now, $\lfloor t/2 \rfloor$ maximum ID robots take the next minimum available port and explore it till four port distances away from the group of r_m robots in all the possible combinations (by giving priority to minimum port number) which are $3 * 3 * 3 = 27$ total different nodes. This round trip would take at most $2 * 27 = 54$ rounds from one hop away from the r_m while moving one hop away from the r_m takes 2 more rounds. Therefore, the total rounds in this round trip are 56. Notice that there exist exactly two ways for the group of $t/2$ robots (even a single robot reporting would be enough to know the appropriate port number) to reach the initial position (position where other $t/2$ robots stay which may crash) of the robots. These two positions would be at 180° to each other. Further, let us consider two cases: (i) one of the ports from which the $t/2$ robots reached the initial position (where the other half of the robots are waiting for the appropriate port number) is the backward port, then the other port would be the desired one in a straight line. Therefore, all the non-crashed robots would proceed from that port in a straight line. (ii) None of the ports is the backward port, then none of the ports among these two ports are at 180° of the backward port. Therefore, one of the remaining ports is the backward port and the other one is at 180° of the backward port. In this way, we find the desired port to proceed in a straight line. Now, all the robots move in a straight line from the appropriate port and repeat the process till all of them reach the boundary edge of the grid. There might be two more possibilities with faulty robots based on the crash that $t/2$ robots crashed without reporting the port at 180° or the other half crashed, in that case, repeat the process by doing the partition of the remaining robots in two halves. There are only $\log n$ such partitions possible. Hence, to move a single hop in a straight line takes at most 56 rounds or at least half of the robots would crash. Therefore, it would take at most $56(\log n + \sqrt{n})$ rounds in reaching the boundary edge of the grid.

Lemma 7. *In stage 1, the group of robots (non-singleton non-crashed robots) placed inside the grid reach corners of the grid in at most $56 \log n + 59\sqrt{n}$ rounds.*

Proof. There are $54\sqrt{n}$ rounds required if both the groups do not crash fully in the straight line movement for the internal faulty robots. But if one of the groups crashes fully, that might happen at most $\log n$ times.

15

Hence, 56 rounds can be wasted without finding the appropriate direction $\log n$ times. Therefore, the total number of rounds required to reach the boundary edge of the grid is $56(\log n + \sqrt{n})$. In Section 5, we have seen that a non-faulty (robot which does not crash) single robot takes $3\sqrt{n}$ robot in the movement of the robot across the boundary edge of the grid. Therefore, non-singleton non-crashed robots reach at corners of the grid in at most $56 \log n + 59\sqrt{n}$ rounds from the internal node of the grid. $\qquad\square$

Lemma 8. *In stage 2, corner robots gather at a single corner in $12\sqrt{n} \log n + 6\sqrt{n}$ rounds.*

Proof. Firstly, half of the robots with the highest ID at their respective corner takes a round of the whole grid via boundary edges and corner edges. This takes four boundary edge explorations for the corner robots. It takes $3\sqrt{n}$ round for one boundary of the grid (discussed in Lemma 7). Therefore, four boundaries take $12\sqrt{n}$ rounds. In this way, each robot gets to know about the minimum ID robot present at the corner. On the other hand, there might be the case that either half of the robots crash fully, then the remaining half (at most) of the robots take another round trip across the boundary edges of the grid. There can be at most $\log n$ such incidents. Therefore, the overall round cost might be $12\sqrt{n} \log n$ to know the minimum ID robot's corner. Secondly, the minimum ID robot can be at most two boundaries ($2\sqrt{n}$ hops) away from any corner. Therefore, to reach the particular corner takes $6\sqrt{n}$ rounds. Hence, overall rounds required to gather at a single corner are $12\sqrt{n} \log n + 6\sqrt{n}$. $\qquad\square$

Lemma 9. *In Lemma 8, all the non-crashed corner robots gather at the single corner.*

Proof. We prove the lemma by contradiction. Let us suppose there exist two corners, C_1 and C_2 in Lemma 8 such that C_2's corner ID was minimum then C_1. But the corner C_0 figures out the minimum ID robot at corner C_1 this implies that C_0's seeker robots neither crossed the C_2's non-seeker robots nor met in the middle with the seeker robots of the C_2. This implies there are no non-seeker robots at C_2. Additionally, seeker robots of C_2 did not cross the non-seeker robots of the C_0. Consequently, there are no seeker robots from the corner C_2 exist in the grid. Therefore, there do not exist any C_2 robots with minimum ID. Hence, the lemma. $\qquad\square$

Lemma 10. *In stage 3, dispersion from the single corner takes $O(\sqrt{n} \log n)$ rounds.*

Proof. In Stage 3, if the number of robots at the single corner is less than $4\sqrt{n} - 4$ then these robots are placed across the boundary nodes of the grid in $4 \cdot 3\sqrt{n}$ rounds based on their ID. On the other hand, if the number of robots at the single corner is more than $4\sqrt{n} - 4$ then an equal number of robots are sent in each column. There can be two cases: either half or more robots crash before reporting or not. If half or more robots report at their corner then half or more of the column does not require any robots further. Now, the number of columns left for the dispersion of the robots in the grid is less than or equal to half. This case can be repeated at most $\log n$ times, then there would be no column/node for the robots' dispersion. In another case, if less than half of the column reports to their corner then more than half of the robots are settled in the grid from the corner. This case also can be repeated at most $\log n$ times, after that there would be no robot for dispersion. In the worst case, these two cases can arise alternatively, where this process can be repeated at most $2\log n$ times. As we know before reporting, there are rounds required $6\sqrt{n}$ rounds for moving on the boundary edge and $112\sqrt{n}$ for moving in the column of the grid. Therefore, dispersion from a single corner takes $236\sqrt{n} \log n$ rounds, i.e., $O(\sqrt{n} \log n)$. $\qquad\square$

Lemma 11. *Our algorithm for faulty setup requires $O(\sqrt{n} \log n)$ memory at each robot.*

Proof. Each robot stores its own ID, therefore, $\log k$ bits are required. In stage 1 $O(\log k)$ memory is required to store the initial node's minimum ID and some constant memory to track the port number before deciding the appropriate straight direction for the internal robots. In the case of boundary edge movement, only port numbering is stored, therefore, constant memory is required.

In stage 2, traversal takes place along the boundary edges, therefore, the minimum ID corner and its position along with some constant port numbering require $O(\log n)$ memory.

In stage 3, the distance from the corner and the number of robots required should be stored at each robot, along with the traversal at the boundary edge and inside the grid. Therefore, $2\sqrt{n}\log n$ bits are required to store the position of the column and the required number of robots. Consequently, all these stages require the $O(\sqrt{n}\log n)$ bits memory. $\qquad\square$

From the above discussion, we have the following results.

Theorem 12. *Consider any unoriented square grid of n nodes in the presence of any number of faulty robots where k is the number of robots, in which, each robot has access to memory $O(\sqrt{n}\log n)$ bits then DISPERSION can be solved deterministically in $O(\sqrt{n}\log n)$ rounds.*

6.2   Extension to Rectangular Grid

We discussed the dispersion of faulty robots on the square grid of $n = \sqrt{n} \times \sqrt{n}$ nodes in Algorithm 6.1. The Algorithm 6.1 can be modified to the dispersion of the faulty robots on the rectangular grid of $n = \ell \times \frac{n}{\ell}$, where ℓ is the length and n/ℓ is the width of the rectangle such that $\ell, n/\ell > 2$. For $n/\ell = 2$, a singleton robot can not decide the direction of the movement on the boundary since all the neighboring nodes have the same degree 3. In a square grid, length and width are of the same dimension, i.e., $\sqrt{n}$. In a rectangular grid, w.l.o.g, we consider that $\ell > n/\ell$. Henceforth, any operation that takes $\sqrt{n}$ rounds in a square grid would take at most ℓ rounds in a rectangular grid in Stage 1, Stage 2, and Stage 3 of the algorithm 6.1. In Stage 3, line 30 considers the longer edge with length ℓ for the dispersion of the robot across the columns. Therefore, we can modify the Algorithm 6.1 for the rectangular grid by replacing the $\sqrt{n}$ to ℓ. This conveys the rectangular grid's dispersion round and memory complexity would be $O(\ell\log n)$ and $O(\ell\log n)$. The correctness and complexity proof directly follows from the square grid. For termination, knowledge of ℓ is required.

Remark 3. *For $n/\ell = 2$, the dispersion can be solved in a rectangular grid if the values of n and ℓ are known. In that case, each boundary robot settles at the empty node, and if there is no empty node then movement in a straight line is executed similar to the movement on the internal node, to break the symmetry.*

7   Conclusion and Future Work

In this paper, we studied dispersion for distinguishable mobile robots on a port-labeled square grid in an arbitrary configuration: oriented with fault and unoriented with and without fault. We presented oriented and unoriented grids with and without fault, respectively, having the round complexity $O(\sqrt{n})$ and $O(\log n)$ bits memory. In contrast, the unoriented grid with faulty robots required both round and memory (in bits) $O(\sqrt{n}\log n)$. Some open questions that are raised by our work: i) What is the non-trivial lower bound in terms of k when $k = o(\sqrt{n})$, for the round complexity in both setups by keeping the memory $O(\log n)$? ii) Is it possible to improve the result in the faulty setup similar to the non-faulty setup? iii) Finally, whether similar bounds hold without the prior knowledge of n.

References

[1] John Augustine and William K. Moses Jr. Dispersion of mobile robots: A study of memory-time trade-offs. In *ICDCN 2018*.

[2] Lali Barriere, Paola Flocchini, Eduardo Mesa-Barrameda, and Nicola Santoro. Uniform scattering of autonomous mobile robots in a grid. *International Journal of Foundations of Computer Science*, 22(03):679–697, 2011.

[3] Hanane Becha and Paola Flocchini. Optimal construction of sense of direction in a torus by a mobile agent. *International Journal of Foundations of Computer Science*, 18(03):529–546, 2007.

[4] Prabhat Kumar Chand, Manish Kumar, Anisur Rahaman Molla, and Sumathi Sivasubramaniam. Fault-tolerant dispersion of mobile robots. In *CALDAM 2023*.

[5] Tien-Ruey Hsiang, Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, and Joseph S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *WAFR 2002*.

[6] Tien-Ruey Hsiang, Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, and Joseph S. B. Mitchell. Online dispersion algorithms for swarms of robots. In Steven Fortune, editor, *SCG 2003*.

[7] Giuseppe F. Italiano, Debasish Pattanayak, and Gokarna Sharma. Dispersion of mobile robots on directed anonymous graphs. In *SIROCCO 2022*.

[8] Ajay D. Kshemkalyani and Faizan Ali. Efficient dispersion of mobile robots on graphs. In *ICDCN 2019*.

[9] Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Dispersion of mobile robots on grids. In *WALCOM 2020*.

[10] Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Efficient dispersion of mobile robots on dynamic graphs. In *ICDCS 2020*.

[11] Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Fast dispersion of mobile robots on arbitrary graphs. In *ALGOSENSORS 2019*.

[12] Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Dispersion of mobile robots using global communication. *J. Parallel Distributed Comput.*, 2022.

[13] Ajay D. Kshemkalyani and Gokarna Sharma. Near-optimal dispersion on arbitrary anonymous graphs. In *OPODIS 2021*.

[14] Subhrangsu Mandal, Anisur Rahaman Molla, and William K. Moses Jr. Live exploration with mobile robots in a dynamic ring, revisited. In *ALGOSENSORS 2020*.

[15] Anisur Rahaman Molla, Kaushik Mondal, and William K. Moses Jr. Byzantine dispersion on graphs. In *IPDPS 2021*.

[16] Takahiro Shintaku, Yuichi Sudo, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Efficient dispersion of mobile agents without global knowledge. In *SSS 2020*.