Dispersion of Mobile Robots on Grids

Ajay D. Kshemkalyani[1], Anisur Rahaman Molla[2], and Gokarna Sharma[3](✉)

[1] University of Illinois at Chicago, Chicago, USA
ajay@uic.edu
[2] Indian Statistical Institute, Kolkata, India
molla@isical.ac.in
[3] Kent State University, Kent, USA
sharma@cs.kent.edu

Abstract. The dispersion problem on graphs asks $k \leq n$ robots initially placed 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 many other fundamental robot coordination problems, such as exploration, scattering, load balancing, relocation of self-driven electric cars (robots) to recharge stations (nodes), etc. The objective in this problem 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. The existing algorithms for trees and arbitrary graphs either minimize time or memory but not both. In this paper, we consider for the very first time the dispersion problem on a grid graph embedded in the Euclidean plane and present solutions that simultaneously minimize both the metrics. The grid graph is appealing as it naturally discretizes the 2-dimensional Euclidean plane and finds applications in many real-life robotic systems. Particularly, we provide two deterministic algorithms on an anonymous grid graph that achieve simultaneously optimal bounds for both the metrics.

1 Introduction

The dispersion of autonomous mobile robots to spread them out evenly in a region is a problem of significant interest in distributed robotics, e.g., see [4,5]. 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) [1]. This problem has many practical applications, for example, in relocating self-driven 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 [1,6]. 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 [1,6].

M. S. Rahman et al. (Eds.): WALCOM 2020, LNCS 12049, pp. 183–197, 2020.
https://doi.org/10.1007/978-3-030-39881-1_16

Table 1. Results on Dispersion for $k \leq n$ robots on n-node square grids (for grids, maximum degree $\Delta = 4$ and diameter $D = O(\sqrt{n})$). Time always remains sub-optimal on a grid applying the arbitrary graph algorithms for Dispersion from the literature. Theorem 1 is simultaneously time-memory optimal for grid when $k = \Omega(n)$ and Theorem 2 is for any $k \leq n$.

Algorithm	Memory/robot (in bits)	Time (in rounds)	Communication model
Lower bound	$\Omega(\log k)$	$\Omega(\sqrt{k})$	Local/global
Applying first algorithm of [6]	$O(k)$	$O(n)$	Local
Applying second algorithm of [6]	$O(D) = O(\sqrt{n})$	$O(4^D) = O(4^{\sqrt{n}})$	Local
Applying third algorithm of [6]	$O(\log k)$	$O(nk)$	Local
Applying algorithm of [7]	$O(\log n)$	$O(k \log k)$	Local
Theorem 1 (this paper)	$O(\log k)$	$O(\min(k, \sqrt{n}))$	**Local**
Applying algorithm of [8]	$O(\log k)$	$O(k)$	Global
Theorem 2 (this paper)	$O(\log k)$	$O(\sqrt{k})$	**Global**

The objective in this problem 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 trees and arbitrary graphs giving algorithms with increasingly better bounds on both the metrics [1,6–8]. However, the existing algorithms (for trees and arbitrary graphs) are only able to minimize either time or memory but not both (details in **related work**). Particularly, some of the existing algorithms obtained optimal memory bounds but no algorithm established optimal time bounds. Therefore, the following question naturally arises: *Is it possible to solve* Dispersion *on some graph classes simultaneously minimizing both time and memory?* In this paper, we answer this question in the affirmative, considering for the very first time Dispersion on a grid graph embedded in the Euclidean plane. Surprisingly, we are not only able to simultaneously minimize both the metrics but also able to achieve optimal bounds for both. Grid graph setting is simple yet appealing as it naturally discretizes the 2-dimensional Euclidean plane and finds applications in many real-life robotic systems, for example, see [9,12].

Specifically, we provide two novel deterministic algorithms for Dispersion on an anonymous grid graph (Table 1). Our first algorithm works in the *local communication* model where a robot can only communicate with other robots that are present at the same node. Our second algorithm works in the *global communication* model where a robot can communicate with any other robot in the graph possibly at different nodes (but the graph structure is not known to robots). The global communication 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 neighboring nodes of G cannot figure out just by communicating which edge of the nodes leads to each other. Therefore, the robots still need to explore through the edges as in the local model. The global communication model has been of much interest in the past in distributed robotics, e.g., see [2,3,11]. Among the previous works [1,6–8] on Dispersion, papers [1,6,7] considered the local communication model and [8] considered the global communication model. Applying the arbitrary graph results of [6–8] to a grid, memory bound obtained is optimal but time remains sub-optimal.

Overview of the Model and Results. We consider a system of $k \leq n$ robots are operating on an n-node graph G. G is assumed to be a connected, undirected graph with m edges, diameter D, and maximum degree Δ. In addition, G is *anonymous*, i.e., nodes have no unique IDs and hence are indistinguishable but the ports (leading to incident edges) at each node have unique labels from $[1, \delta]$, where δ is the degree of that node. The robots are *distinguishable*, i.e., they have unique IDs in the range $[1, k]$. The robot activation setting is *synchronous* – all robots are activated in a round and they perform their operations simultaneously in synchronized rounds. Runtime is measured in rounds (or steps). We establish the following result in the local model.

Theorem 1. *Given any initial configuration of $k \leq n$ mobile robots on an anonymous n-node square grid graph G, Dispersion can be solved in $O(\min(k, \sqrt{n}))$ time with $O(\log k)$ bits memory at each robot in the local communication model.*

Theorem 1 is simultaneously time-memory optimal when $k = \Omega(n)$ since a time lower bound of $\Omega(D)$ trivially holds for Dispersion of n robots on any graph [1] and the diameter of a n-node grid is $D = \Omega(\sqrt{n})$. Furthermore, $\Omega(\log k)$ bits are necessary at a robot to distinguish the robots from each other. We also extend Theorem 1 on a rectangular grid G. We establish the following result in the global model.

Theorem 2. *Given any initial configuration of $k \leq n$ mobile robots on an anonymous n-node square grid graph G, Dispersion can be solved in $O(\sqrt{k})$ time with $O(\log k)$ bits memory at each robot in the global communication model.*

Theorem 2 is simultaneously time-memory optimal for Dispersion for any $k \leq n$. We also extend Theorem 2 for Dispersion on a rectangular grid graph G.

Challenges and Techniques. The solutions proposed in the literature in the local [1,6,7] and global [8] communication models for Dispersion on arbitrary graphs and trees (grids were not considered before) heavily use a DFS (depth first search) traversal approach. The best bound so far for grid in the local model is obtained while applying the arbitrary graph algorithm of [7] (the bounds obtained in [7] for arbitrary graphs are given in related work): $O(k \log k)$ time with $O(\log k)$ bits at each robot (see Table 1). However, for a grid, the time lower bound is $\Omega(D) = \Omega(\sqrt{k})$ for any $k \leq n$ and memory lower bound is obviously $\Omega(\log k)$ at each robot to distinguish the robots from each other (consider the case of all k robots are at a single node of G initially). Therefore, we develop two techniques for grids, one specific to the local model and another to the global model. The technique for the local model uses the grid properties (instead of a DFS traversal). Particularly, the grid technique in the local model uses the idea of repositioning first all robots to the boundary nodes of G using the grid properties (despite the grid being anonymous), then collect them to a boundary corner node of G, and finally distribute them to the nodes of G leaving one robot on each node. The technique for the global model uses the grid properties with a limited use of the DFS traversal tailored specifically to the information available to robots during execution due to the global model. Particularly, the grid technique in the global model uses the DFS traversal for $O(\sqrt{k})$ time, then forms a square boundary of perimeter $4(\sqrt{k}-1)$, and disperses all k robots within that boundary in $O(\sqrt{k})$ time. Although the techniques above seem rather straightforward in high level, we need to overcome many challenges for them to correctly work while putting together to solve Dispersion.

Related Work. As discussed above, for Dispersion, there are three previous studies [1,6,7] in the local communication model. For $k = n$, Augustine and Moses Jr. [1] 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 any 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. Kshemkalyani and Ali [6] provided an $\Omega(k)$ time lower bound for 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. Recently, Kshemkalyani *et al.* [7] provided an algorithm for arbitrary graphs that runs in $O(\min(m, k\Delta) \cdot \log k)$ time using $O(\log n)$ bits at each robot. There is one previous study [8] for Dispersion in the global communication model, which 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 [7] in the local model by an $O(\log k)$ factor. Randomized algorithms for Dispersion are presented in [10] where the random bits are used to reduce the memory requirement. Other closely related works to Dispersion are omitted due to space constraints.

Roadmap. We discuss model details in Sect. 2. We present an algorithm in the local model in Sect. 3. We present an algorithm in the global model in Sect. 4. Finally, we conclude in Sect. 5 with a short discussion on possible future work. Due to space constraints, many details and proofs are deferred to a full version.

2 Model Details and Preliminaries

Grid Graph. Let $G = (V, E)$ be an n-node grid graph embedded in the 2-dimensional Euclidean plane, i.e., $|V| = n$ and $|E| = m = O(n)$. G is assumed to be connected, unweighted, and undirected with no holes. G is *anonymous*, i.e., nodes do not have identifiers but, at any node, its incident edges are uniquely identified by a *label* (aka port number) in the range $[1, \delta]$, where $\delta \leq 4$ is the *degree* of that node. The *maximum degree* of G is $\Delta = 4$, which is the maximum among the degree δ of the nodes in G. Any edge e connecting two nodes $u, v \in G$ has two port numbers associated with it, one at the end of e towards u and another at the end of e towards v. We assume that there is no correlation between two port numbers of an edge. For a grid graph G, the nodes on 2 boundary rows and 2 boundary columns are called *boundary nodes* and the 4 corner nodes on boundary rows (or columns) are called *boundary corner nodes.* In an n-node square grid graph G, there are exactly $4\sqrt{n} - 4$ boundary nodes. Any number of robots are allowed to move along an edge at any time. The grid nodes do not have memory, i.e., they are not able to store any information.

Robots. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ be a set of $k \leq n$ robots residing on the nodes of G. For simplicity, we sometime use i to denote robot r_i. No robot resides on the edges of G, but one or more robots can occupy the same node of G. 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, it is aware of the port of u it used to leave u and the port of v it used to enter v. Furthermore, it is assumed that each robot is equipped with memory to store information, which may also be read and modified by other robots present on the same node.

Communication Model. We have two communication models: local and global. In the local communication model, a robot can only communicate with other robots present on the same node. In contrast, in the global communication model, a robot is capable to communicate with any other robot in the graph G, irrespective of their positions in the graph. However, they will not have the position information as graph nodes are anonymous and there is no correlation between two port numbers of an edge.

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

- *Communicate:* r_i can communicate with and observe the memory of some other robot $r_j \in \mathcal{R}$ (at the same node or a different node) depending on the communication model used. Robot r_i can also observe its own memory.
- *Compute:* r_i may perform an arbitrary computation using the information observed during the "communicate" portion of that cycle. This includes determination of a (possibly) port to use to exit v_i and the information to store in the robot r_j at v_i.
- *Move:* At the end of the cycle, r_i writes new information (if any) in the memory of r_j at v_i, and exits v_i using the computed port to reach to a neighbor v_i' of v_i. After entering v_i', it will keep track of the port at v_i' from which it entered v_i'.

Time and Memory Complexity. We consider the synchronous setting where every robot is active in every CCM cycle and they perform the cycle in a syn-

Algorithm 1: *Grid_Disperse*(k) in the local communication model

```
1 Input: An $n$-node square grid $G$ with $k \leq n$ robots positioned arbitrarily on its
  nodes.
2 if $k \geq \sqrt{n}$ then
3     Stage 1: the robots not already on the boundary nodes move to the
      boundary nodes;
4     Stage 2: the robots not on the boundary corner nodes move to the corner
      nodes;
5     Stage 3: the robots not on the boundary corner nodes move to a corner node;
6     Stage 4: the robots distribute on the nodes of a grid side, at most $\sqrt{k}$ on
      each node;
7     Stage 5: the robots disperse with one robot on a node;
8 else
9     The robots move in a direction from their position to find a free node to
      settle;
```

chrony. Therefore, time is measured in *rounds* or *steps* (a cycle is a round or step). Another parameter is memory which comes from a single source – the number of bits stored at each robot.

Mobile Robot Dispersion. DISPERSION can be formally defined as follows.

Definition 1 (DISPERSION). *Given any n-node anonymous grid $G = (V, E)$ 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.*

The goal is to optimize two performance metrics: (i) **Time** – the number of rounds (steps), and (ii) **Memory** – the number of bits stored at each robot.

3 Algorithm in the Local Communication Model (Theorem 1)

We present and analyze an algorithm, *Grid_Disperse*(k), that solves DISPERSION for $k \leq n$ robots on n-node square grid graphs in $O(\min(k, \sqrt{n}))$ time using $O(\log k)$ bits at each robot in the local communication model. The high level pseudocode is given in Algorithm 1. We discuss algorithm *Grid_Disperse*$(k), k = \Omega(n)$, here. *Grid_Disperse*$(k), k \leq o(n)$, and the algorithm for rectangular grid graphs for any $k \leq n$ are omitted here and discussed in a full version due to space constraints.

High Level Overview of the Algorithm. *Grid_Disperse*$(k), k = \Omega(n)$, for n-node square grid graphs has five stages, Stages 1–5 (Algorithm 1), which execute sequentially one after another. The goal in Stage 1 is to move all the robots (not already on boundary nodes) to position them on the boundary nodes of G. The goal in Stage 2 is to move the robots, now all on the boundary nodes, to the four boundary corner nodes of G. The goal in Stage 3 is to collect all robots, now on four corner nodes, at one corner node of G. The goal in Stage 4 is to distribute robots equally on the nodes of a boundary row or column of G. The goal in Stage 5 is to distribute the robots, now on the nodes of a boundary row or column, so that each node of G has exactly one robot positioned on it. We will show that each stage can be performed correctly in $O(\sqrt{n})$ rounds, giving overall $O(\sqrt{n})$ time complexity for *Grid_Disperse*(k). Algorithm *Grid_Disperse*(k) for $k \leq o(n)$ differentiates the cases of $\sqrt{n} \leq k \leq o(n)$ and $k < \sqrt{n}$ and handles them through separate algorithms. For $\sqrt{n} \leq k \leq o(n)$, we provide a $O(\sqrt{n})$-time algorithm, and for $k < \sqrt{n}$, we provide a $O(k)$-time algorithm, giving overall $O(\min(k, \sqrt{n}))$ runtime for any $k \leq n$. We then extend all these ideas for Dispersion on rectangular grid graphs.

Algorithm for Square Grid Graphs, $k = \Omega(n)$. We describe in detail how Stages 1–5 of *Grid_Disperse*(k) are executed for square grids. The high level pseducode is in Algorithm 1. Figure 1 illustrates the working principle of *Grid_Disperse*(k) for $n = k = 49$. Each robot $r_i \in \mathcal{R}$ stores five variables $r_i.round$ (initially 0), $r_i.stage$ (values 1 to 5, initially *null*), $r_i.port\_entered$ (values 1 to 4, initially *null*), $r_i.port\_exited$ (values 1 to 4, initially *null*), and $r_i.settled$ (values 0 and 1, initially 0). We assume that in each round r_i updates $r_i.round \leftarrow r_i.round + 1$. Moreover, we denote the rounds of each stage by $\alpha.\beta$, where $\alpha \in \{1, 2, 3, 4, 5\}$ denotes the stage and β denotes the round within the stage. Therefore, the first round $(\alpha+1).1$ for Stage $\alpha+1$ is the next round after the last round of Stage α.

Stage 1. The goal in Stage 1 is to reposition the robots that are not already on boundary nodes of G to the boundary nodes of G. The robots that are already on (or reach during Stage 1) the boundary nodes do nothing until Stage 1 finishes. The idea here is, for each robot independently, to choose a port of a non-boundary node to exit in the current round based on the port from which it entered that non-boundary node in the previous round. Pick a robot $r_i \in \mathcal{R}$ at some node $v \in G$ which is not the boundary node. In round 1.1, r_i does not have information on the port of v from which it entered v (r_i has not moved yet). Therefore, it randomly picks one of the four ports at v and exits v. Suppose, in the beginning of round 1.2, r_i reaches a neighbor of v, say w. If w is a boundary node, we are done as r_i can differentiate a boundary node from a non-boundary node (a boundary node has three ports instead of four ports at a non-boundary node). While reaching w, the model provides r_i with the information on the port p at w from which r_i entered w. That means, in round 1.2 and after, r_i has information on port from which it entered the node in the previous round where it is positioning in the current round. Therefore, in round 1.2, r_i moves as follows. Besides port p, w has three other ports. r_i orders (in clockwise or counterclockwise direction) the three remaining ports at w starting from p. It then picks the second port in either order and exits w using that port. The process is then repeated by r_i in round 1.3 and after until it reaches a boundary node.

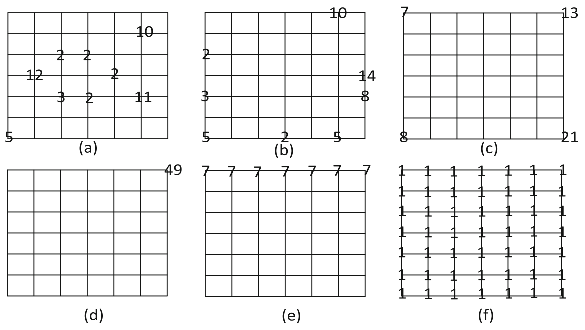

Fig. 1. An illustration of the five stages of algorithm *Grid_Disperse*(k) for $n = k = 49$: (**a**) An initial configuration, (**b**) Stage 1 that moves robots to boundary nodes of G, (**c**) Stage 2 that moves robots to four boundary corner nodes of G, (**d**) Stage 3 that moves the robots to one boundary corner node of G, (**e**) Stage 4 that distributes robots equally in a row (or a column) with each node having $\sqrt{n}$ robots, and (**f**) Stage 5 that distributes robots so that each node of G having exactly one robot each. The numbers denote the number of robots positioned at that node.

Lemma 1. *Pick a robot r_i at non-boundary node $v \in G$. Let r_i moves to a neighbor node w of v in round 1.1. Let $L_{\overrightarrow{vw}}$ be a line with one end v and passing through w. In round 1.2 and after (until r_i reaches a boundary node), r_i moves following the nodes of G that are on $L_{\overrightarrow{vw}}$ in the same direction in each round.*

Proof. Let the four ports of v be p_{v1}, p_{v2}, p_{v3}, and p_{v4}. Suppose r_i exits v using p_{v1} in round 1.1 and reaches node w in the beginning of round 1.2. If w is a boundary node, we are done. Otherwise, it remains to show that in round 1.2 and after, r_i always moves on the nodes of $L_{\overrightarrow{vw}}$ in the same direction. Let p_{w1} be the port at w from which r_i entered w in round 1.1. The three remaining ports at w are p_{w2}, p_{w3}, and p_{w4}. Since r_i picks second port in the clockwise (or counterclockwise) order in round 1.2 and after, the port r_i picks at w is always opposite port of port p_{w1} that it used to enter w from v in round 1.1. Notice that r_i is aware of p_{w1} while at w. Therefore, in the beginning of round 1.3, r_i reaches a neighbor node of w on $L_{\overrightarrow{vw}}$ (opposite of v). Continuing this way, the move decision in round 1.3 (and after) resembles the approach of round 1.2, takes r_i farther way from v on $L_{\overrightarrow{vw}}$ in each subsequent round. □

Lemma 2. *At the end of Stage 1, all $k = \Omega(n)$ robots in any initial configuration are positioned on boundary nodes of G. Stage 1 finishes in $\sqrt{n} - 1$ rounds.*

Stage 2. The goal in Stage 2 is to collect all $k = \Omega(n)$ robots on boundary nodes of G to four boundary corners of G. Let L_{ab} be a boundary row or column of G passing through boundary corners a, b of G. For the rest three boundary row or columns the ideas are analogous and run in parallel. In Stage 2, *Grid_Disperse*(k) collects the robots on the nodes on L_{ab} to either node a or node b or some on a and some on b. The idea here is to move each robot independently in a direction on the boundary row or column of the grid until the robot reaches a boundary corner node. In this process, while selecting a port to leave from a current node to the next node in the boundary row or column, the robot may reach to a non-boundary node, in which case it will return back to the boundary node in the next round using the same port from which it entered the non-boundary node. Details are in a full version. We have the following lemma.

Lemma 3. *At the end of Stage 2, all $k = \Omega(n)$ robots in G are positioned on (at most) 4 boundary corner nodes of G. Stage 2 finishes in $3(\sqrt{n} - 1)$ rounds after Stage 1.*

Stage 3. The goal in Stage 3 is to collect all $k = \Omega(n)$ robots on a boundary corner node of G. Let a, b, c, d be the four boundary corner nodes of G. Suppose the smallest ID robot $r_1 \in \mathcal{R}$ is positioned on a. Pick any robot $r_i \neq r_1$. If r_i is already on a, it does nothing in Stage 3. Otherwise, it is on b, c, or d (say b) and it moves in Stage 3 to reach a. Since r_i needs to move on the boundary of G, the technique of Stage 2 can be modified for r_i so that it reaches a following the boundary nodes (details omitted).

Lemma 4. *At the end of Stage 3, all $k = \Omega(n)$ robots in G are positioned on a boundary corner nodes of G. Stage 3 finishes in $9(\sqrt{n} - 1)$ rounds after Stage 2.*

Stage 4. The goal is Stage 4 is to distribute $k = \Omega(n)$ robots (that are at a boundary corner node a after Stage 3) to a boundary row or column so that there will be no more than $\sqrt{n}$ robots on each node. In the first round, the smallest ID robot r_1 moves to pick a direction (i.e., row or a column). From the second round onwards, the idea similar to Stage 2 is used to move the robots on the boundary row or column leaving $\sqrt{n}$ robots on each node of that row or column. Details are omitted.

Lemma 5. *At the end of Stage 4, all $k = \Omega(n)$ robots in G are distributed on a boundary row or column of G so that there will be exactly $\sqrt{n}$ or less robots on a node. Stage 4 finishes in $3\sqrt{n} - 1$ rounds after Stage 3.*

Stage 5. The goal in Stage 5 is to distribute robots to nodes of G so that there will be at most one robot on each node. Let c be a boundary node with $\sqrt{n}$ or less robots on it and r_i is on c. In round 5.1, if r_i is the largest ID robot r_{max} among the robots on c, it settles at c assigning $r_i.settled \leftarrow 1$. Otherwise, in round 5.1, r_i moves as follows. While executing Stage 4, r_i stores the port of c it used to enter c (say $r_i.port\_entered = p_{c1}$) and the port of c used by the robot that left c exited through (say $r_i.port\_exited = p_{c2}$). Robot r_i then exits through port p_{c3}, which is not $r_i.port\_entered$ and $r_i.port\_exited$. This way r_i reaches a non-boundary node c'. All other robots except r_{max} also reach c' in the beginning of round 5.2. In round 5.2, the largest ID robot $r_{max'}$ settles at c'. The other at most $\sqrt{n} - 2$ robots exit c' using the port of c' selected through the port ordering technique described in Stage 1. This process then continues until a single robot remains at a node z, which settles at z.

Lemma 6. *At the end of Stage 5, all $k = \Omega(n)$ robots in G are distributed such that there is exactly one robot on a node of G. Stage 5 finishes in $\sqrt{n}$ rounds after Stage 4.*

Theorem 3. *$Grid\_Disperse(k)$ solves* Dispersion *correctly for $k = \Omega(n)$ robots on an n-node square grid graph G in $O(\sqrt{n})$ rounds with $O(\log n)$ bits at each robot.*

Proof. Each stage of $Grid\_Disperse(k)$ executes sequentially one after another. Therefore, the correctness of $Grid\_Disperse(k)$ follows combining the correctness proofs of Lemmas 2–6. The time bound of $O(\sqrt{n})$ rounds also follows summing up the $O(\sqrt{n})$ rounds of each stage. Regarding memory, variables *port_entered*, *port_exited*, *settled*, and *stage* take $O(1)$ bits ($\Delta = 4$ for grids), and *round* takes $O(\log n)$ bits. Moreover, two or more robots at a node can be differentiated using $O(\log k)$ bits, for $k = \Omega(n)$. Therefore, a robot needs in total $O(\log n)$ bits. □

We have the following theorem for $k \leq o(n)$. Details are omitted.

Theorem 4. *$Grid\_Disperse(k)$ solves* Dispersion *correctly for $k \leq o(n)$ robots on a square grid G in $O(\min(k, \sqrt{n}))$ rounds with $O(\log k)$ bits at each robot.*

Proof of Theorem 1: Theorems 3 and 4 together prove Theorem 1 for any $k \leq n$. □

Algorithm 2: *$Grid\_Disperse\_Global(k)$* in the global communication model

---

```
1 Input: An $n$-node square grid $G$ with $k \leq n$ robots positioned arbitrarily on its
  nodes.
2 Stage 1: For each node of $G$ with two or more robots, run a DFS traversal
  algorithm of Kshemkalyani et al. [8] for $W = 48\sqrt{k}\Delta$ rounds;
3 Stage 2: If a DFS traversal tree component formed in Stage 1 has $14\sqrt{k}$ or
  more nodes, then (i) divide the component into sub-components having $6\sqrt{k}$ to
  $14\sqrt{k} - 1$ nodes, and (ii) collect all the robots on each sub-component into a
  node in that sub-component;
4 Stage 3: The nodes in $G$ with at least $6\sqrt{k}$ robots positioned on them form a
  square boundary of length $4(\sqrt{k} - 1)$ having $4(\sqrt{k} - 1)$ robots on the boundary
  (the remaining nodes of $G$ has at most one robot on each);
5 Stage 4: If two or more square boundaries overlap, make them non-overlapping
  by coalescing some square boundaries with others;
6 Stage 5: If there are robots, if any, in the interior of each square boundary,
  collect those robots to a corner of that boundary such that only that corner has
  multiple robots;
7 Stage 6: Disperse the multiple robots on a single corner of each square
  boundary to the nodes in the interior of that boundary;
```

4 Algorithm in the Global Communication Model (Theorem 2)

We present and analyze an algorithm, *Grid_Disperse_Global*(k), that solves Dispersion for $k \leq n$ robots on n-node square grid graphs in $O(\sqrt{k})$ time with $O(\log k)$ bits at each robot in the global model. We discuss *Grid_Disperse_Global*(k) for square grids here. The algorithm for rectangular grid graphs is in a full version.

High Level Overview of the Algorithm. *Grid_Disperse_Global*(k) for square grid graphs has six stages, Stage 1–6, which execute sequentially one after another. *Grid_Disperse_Global*(k) works for any $k \leq n$. Each stage runs for $O(\sqrt{k})$ rounds, giving overall $O(\sqrt{k})$ runtime. Stage 1 runs a DFS traversal in parallel for the nodes of G with at least two robots in the initial configuration. Stage 1 forms one or more DFS traversal tree components. Each DFS traversal tree component is a graph. Stage 2 divides each DFS traversal tree component with $14\sqrt{k}$ or more nodes into sub-components containing between $6\sqrt{k}$ to $14\sqrt{k} - 1$ nodes. The robots on a sub-component (say CC) are then collected to a node (say v_{CC}) on that sub-component. Stage 3 is then initiated by the nodes of grid G (for example, node v_{CC} of sub-component CC) having at least $6\sqrt{k}$ robots. They create a square boundary of length $4(\sqrt{k} - 1)$ so that there will be exactly $4(\sqrt{k}-1)$ robots positioned on the square boundary. If there is overlapping between two or more square boundaries constructed in Stage 3, Stage 4 makes them non-overlapping through merging all the square boundaries to one. Stage 5 collects the robots, if any, that are in the interior of each square boundary to a boundary corner node of the square boundary (again the node is v_{CC} for the sub-component CC). Finally, Stage 6 first distributes the robots on the boundary corner node (v_{CC} for sub-component CC) equally to the nodes on a row or a column of the square boundary and then disperses those robots to the nodes inside the square boundary, one on each node, achieving a Dispersion configuration.

Algorithm for Square Grid Graphs, $k \leq n$. We now describe in detail how Stages 1–6 of *Grid_Disperse_Global*(k) are executed for n-node square grid graphs. The high level pseudocode is given in Algorithm 2. We also abstract some details on what variables are used and how they are used to provide better readability for the overall algorithm. The techniques are highly involved compared to Sect. 3.

Stage 1. Stage 1 is for the nodes of G which have at least two robots positioned on them in the initial configuration. Stage 1 runs a DFS traversal algorithm for arbitrary graphs developed Kshemkalyani *et al.* [8] in the global communication model. Kshemkalyani *et al.* [8] run this DFS traversal until a Dispersion configuration is achieved, and hence the runtime becomes $\min(2 \cdot 4m, 2 \cdot 2k\Delta)$ rounds for arbitrary graphs. In grid, running this algorithm until reaching a Dispersion configuration needs $16k$ rounds (in a n-node square grid $m \leq 4n$, $\Delta = 4$, and $k \leq n$), which is clearly sub-optimal. Recall that our goal is to achieve optimal runtime of $O(\sqrt{k})$ rounds in grids. Therefore, we run the DFS traversal of [8] in the global communication model only for $48\sqrt{k}\Delta$ rounds and derive some properties to achieve a $O(\sqrt{k})$-round algorithm for a grid.

Lemma 7. *Let $W = 48\sqrt{k}\Delta$. If the DFS traversal forming a DFS tree component stops before W rounds, there is exactly one robot on each node of that DFS tree component. IF the DFS traversal does not stop until W rounds, there will be $\geq 6\sqrt{k}$ nodes (with at least a robot on each node) on the DFS tree component.*

Stage 2. Consider a connected DFS tree component CC_i with ID CID_i formed in Stage 1. Stage 2 will run for CC_i if it has at least a node with two or more robots on it. For simplicity, we denote such DFS tree components by $CC_i(not\_settled)$. Robots in the component $CC_i(not\_settled)$ have knowledge of the component ID CID_i of $CC_i(not\_settled)$. We have from Lemma 7 that $CC_i(not\_settled)$ must have at least $6\sqrt{k}$ nodes. Suppose $CC_i(not\_settled)$ has $X_i \geq 6\sqrt{k}$ nodes. Then, $CC_i(not\_settled)$ must have $Y_i > X_i$ robots. If $CC_i(not\_settled)$ has $6\sqrt{k} \leq X_i < 14\sqrt{k} - 1$ nodes, we collect all the robots of $CC_i(not\_settled)$ to the node of $CC_i(not\_settled)$ that is CID_i. If there are $X_i \geq 14\sqrt{k}$ nodes on $CC_i(not\_settled)$, we partition $CC_i(not\_settled)$ into sub-components $CC_i^{sub,j}(not\_settled)$ such that each $CC_i^{sub,j}(not\_settled)$ has between $6\sqrt{k}$ and $14\sqrt{k} - 1$ nodes on it. After this, the robots on the nodes of each sub-component $CC_i^{sub,j}(not\_settled)$ are collected to a node in that sub-component. Note that we collect only for sub-components $CC_i^{sub,j}(not\_settled)$ which have $Y_i > X_i$ (or at least a node on that sub-component has two robots on it). It remains to discuss two techniques:

(a) How the partitioning of a connected DFS tree component $CC_i(not\_settled)$ into sub-components $CC_i^{sub,j}(not\_settled)$ is achieved.
(b) How the collection of the robots on the nodes of each sub-component $CC_i^{sub,j}(not\_settled)$ into a node in that sub-component is achieved.

The details are in a full version. We have the following lemma after Stage 2.

Lemma 8. *At the end of Stage 2, there is either no, one, or at least $6\sqrt{k}$ robots on the nodes of grid G. Stage 2 finishes in $O(\sqrt{k})$ rounds.*

Stage 3. Stage 3 is initiated in parallel by the grid nodes which have at least $6\sqrt{k}$ robots positioned on them at the end of Stage 2 (Lemma 8). Consider a node $w \in G$ which has at least $6\sqrt{k}$ robots on it at the end of Stage 2. The goal is to form a square boundary SB_w of length $4(\sqrt{k}-1)$ containing one robot each on the boundary nodes (except node w which will have at least $2\sqrt{k}+1$ robots). This boundary helps to disperse all k robots in the interior. Details are omitted on how the square boundary is formed.

Lemma 9. *At the end of Stage 3, for each node of G with at least $6\sqrt{k}$ robots on it at end of Stage 2, a square boundary of length $4(\sqrt{k}-1)$ is correctly formed with each boundary node having one robot positioned on it, with only exception of a node on the boundary which has at least $2\sqrt{k}+1$ robots on it. Stage 3 finishes in $O(\sqrt{k})$ rounds.*

Stage 4. The goal in Stage 4 is make the square boundaries SB_w non-overlapping. Consider a square boundary SB_i with ID SID_i (the smallest ID robot on node w for the square boundary SB_w). All the robots on the boundary of SB_i know they belong to the boundary SB_i. In the first round of Stage 4, if any robot r_i of SB_i has some other robot r_j with SID_j colocated on the same node, it broadcasts a $\mathit{Overlap}(SID_i, SID_j)$ message. This is to indicate that the square boundary SB_i has met the square boundary $SB_j, j \neq i$. All the robots listen to such broadcasts and build an undirected *square boundary overlapping graph* $SB\_Overlap(B, E')$, where B is the set of square boundary IDs, and edge (SID_i, SID_j) indicates that $\mathit{Overlap}(SID_i, SID_j)$ message has been received. There might be one or more $SB\_Overlap(B, E')$ components and Stage 4 runs in parallel for each of those. A maximal independent set (MIS) is computed for each component and the square boundaries which are not part of the MIS will merge with the neighboring square boundary that is part of the MIS. Details are omitted.

Lemma 10. *At the end of Stage 4, all square boundaries are non-overlapping. The* $4(\sqrt{k}-1)$ *boundary nodes in each square boundary have a robot each, except a node which has at least* $2\sqrt{k}+1$ *robots. Stage 4 finishes in* $O(\sqrt{k})$ *rounds.*

Stage 5. At the end of Stage 4, all the remaining square boundaries are non-overlapping. The only problem might be that there are robots in the interior of the square boundaries. This is particularly because of DFS traversal trees that stopped before W rounds in Stage 1 and the robots of square boundaries from Stage 4 not collected yet. The goal in Stage 5 is to collect those robots, if any, inside each square boundary SB_i to the SID_i node so that at the end of Stage 5, there are robots only the boundary of each SB_i. A challenge is how to collect robots if any in the interior of SB_i in $O(\sqrt{k})$ rounds. For this, $\sqrt{k}-2$ robots first move in a row or column of SB_i and then to the interior of SB_i until reaching the opposite row or column, collecting all the robots that are found on the traversal. They then return to SID_i with all the robots (if any) in the interior of SB_i. Details are in a full version. We have the following lemma.

Lemma 11. *At the end of Stage 5, there will be no robot in the interior of the non-overlapping square boundaries obtained in Stage 4. Only a corner of square boundaries have at least* $2\sqrt{k}+1$ *robots. Stage 5 finishes in* $O(\sqrt{k})$ *rounds.*

Stage 6. At the end of Stage 5, we have the following: (i) All square boundaries SB_i are non-overlapping, (ii) There is no robot in the interior of the square boundaries, and (iii) Each boundary node of SB_i has exactly one robot on it except one corner SID_i which has at least $T_i \geq 2\sqrt{k}+1$ robots. The goal in Stage 6 is to disperse $T_i - 1$ robots to the interior nodes in SB_i. Stage 6 uses the ideas as in Stages 4 and 5 of Sect. 3 modified appropriately. Details are in a full version.

Lemma 12. *At the end of Stage 6, all $k \leq n$ robots are distributed such that there is at most one robot on a node of G. Stage 6 finishes in $O(\sqrt{k})$ rounds.*

Theorem 5. *Grid_Disperse_Global(k) solves* DISPERSION *correctly for $k \leq n$ robots on an n-node square grid G in $O(\sqrt{k})$ rounds with $O(\log k)$ bits at each robot.*

Proof. The correctness and runtime of *Grid_Disperse_Global*(k) follows combining Lemmas 7–12. Regarding memory, *Grid_Disperse_Global*(k) uses $O(1)$ number of different variables to be stored by each robot, taking $O(\log k)$ bits for each variable. □

Proof of Theorem 2: Theorem 5 proves Theorem 2 for any $k \leq n$. □

5 Concluding Remarks

We have studied the robot DISPERSION problem on graphs with the object of simultaneously minimizing two fundamental performance metrics: time and memory at each robot. We have presented two algorithms for DISPERSION of $k \leq n$ robots considering for the very first time grid graphs that find applications in many real-life robotic systems and provide simultaneously optimal bounds for both the metrics. The first result is for the local communication model and the second result is for the global communication model. The existing results in the literature were only for trees and arbitrary graphs and they were not able to simultaneously minimize both the metrics.

For future work, it will be interesting to solve DISPERSION on grids with time $O(\sqrt{k})$ in the local model for any $k \leq n$. Another interesting direction will be to extend our algorithms to semi-synchronous and asynchronous settings.

References

1. Augustine, J., Moses Jr., W.K.: Dispersion of mobile robots: a study of memory-time trade-offs. In: ICDCN, pp. 1:1–1:10 (2018)
2. Das, S., Dereniowski, D., Karousatou, C.: Collaborative exploration of trees by energy-constrained mobile robots. Theory Comput. Syst. **62**(5), 1223–1240 (2018)
3. Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks **48**(3), 166–177 (2006)
4. Hsiang, T., Arkin, E.M., Bender, M.A., Fekete, S.P., Mitchell, J.S.B.: Algorithms for rapidly dispersing robot swarms in unknown environments. In: WAFR, pp. 77–94 (2002)
5. Hsiang, T.-R., Arkin, E.M., Bender, M.A., Fekete, S., Mitchell, J.S.B.: Online dispersion algorithms for swarms of robots. In: SoCG, pp. 382–383 (2003)
6. Kshemkalyani, A.D., Ali, F.: Efficient dispersion of mobile robots on graphs. In: ICDCN, pp. 218–227 (2019)
7. Kshemkalyani, A.D., Molla, A.R., Sharma, G.: Fast dispersion of mobile robots on arbitrary graphs. CoRR, abs/1812.05352 (2018). (Accepted to ALGOSENSORS 2019)

8. Kshemkalyani, A.D., Molla, A.R., Sharma, G.: Dispersion of mobile robots in the global communication model. CoRR, abs/1909.01957 (2019). (Accepted to ICDCN 2020)
9. Martnez, H., Cnovas, J.P., Zamora, M.A., Skarmeta, A.G.: i-Fork: a flexible AGV system using topological and grid maps. In: ICRA, pp. 2147–2152 (2003)
10. Molla, A.R., Moses Jr., W.K.: Dispersion of mobile robots: the power of randomness. In: Gopal, T.V., Watada, J. (eds.) TAMC 2019. LNCS, vol. 11436, pp. 481–500. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14812-6_30
11. Ortolf, C., Schindelhauer, C.: Online multi-robot exploration of grid graphs with rectangular obstacles. In: SPAA, pp. 27–36 (2012)
12. Sharma, G., Dutta, A., Kim, J.-H.: Optimal online coverage path planning with energy constraints. In: AAMAS, pp. 1189–1197 (2019)