Optimal Randomized Complete Visibility on a Grid for Asynchronous Robots with Lights

Gokarna Sharma[1], Ramachandran Vaidyanathan[2], and Jerry L. Trahan[2]

[1]Kent State University, Kent, OH 44242, USA
sharma@cs.kent.edu
[2]Louisiana State University, Baton Rouge, LA 70803, USA
{vaidy,jtrahan}@lsu.edu

Abstract. We consider the distributed setting of N autonomous mobile robots that operate in *Look-Compute-Move* (LCM) cycles and communicate with other robots using colored lights (the *robots with lights* model). This model has *obstructed visibility* where a robot cannot see another robot if a third robot is positioned between them on the straight line connecting them. In this paper, we consider an infinite grid embedded in the 2-dimensional Euclidean plane that restricts the movements of each robot to one of the four neighboring grid points. Grid setting is appealing as it naturally discretizes the 2-dimensional Euclidean plane and finds applications in many real-life robotic systems. The COMPLETE VISIBILITY problem is to reposition N autonomous robots (starting at arbitrary, but distinct, initial positions) so that each robot is visible to all others. The objective in this problem is to simultaneously minimize (or provide trade-off between) two fundamental performance metrics: (i) Time to solve COMPLETE VISIBILITY and (ii) Number of distinct colors used by each robot light. We provide the first $\mathcal{O}(\max\{D, N\})$-time algorithm for COMPLETE VISIBILITY in the asynchronous setting, where D is the diameter of the initial configuration. The number of colors used in our algorithm depends on whether leader election is required or not: (i) 17 colors if leader election is not required and (ii) 50 colors if leader election is required. We also prove a time lower bound of $\Omega(N)$ even if unlimited number of colors are provided and leader election is not required, i.e., our algorithm is time-optimal when $D = \mathcal{O}(N)$. The best previously known algorithm for this problem has time $\Omega(\max\{DN, N^2\})$ using 11 colors without leader election, which our algorithm improves significantly with respect to time.

1 Introduction

The classical model. The classical model of distributed computing by mobile robots models each robot as a point in the plane [7]. The local coordinate system of a robot may not be consistent with that of other robots. The sensory capability of a robot, generally called *vision*, allows a robot to determine the positions of other robots in its own coordinate system. The robots are *autonomous* (no external control), *anonymous* (no unique identifiers), *indistinguishable* (no external identifiers), and *disoriented* (no agreement on local coordinate systems). They execute the same algorithm. Typically, robots have *unobstructed* visibility, that is, robots are transparent such that three collinear robots can see each other. Each robot proceeds in *Look-Compute-Move* (LCM) cycles: When a robot becomes active, it first gets a snapshot of its surroundings (*Look*), then computes

a destination based on the snapshot (*Compute*), and finally moves to the destination (*Move*). Moreover, the robots are *oblivious* and *silent* – each robot has no memory of its past LCM cycles and they do not communicate directly [7].

Robots with lights. While silence has advantages, many other situations, e.g., hostile environments, assume direct communication. The *robots with lights* model [3,7,11] incorporates direct communication – each robot has a visible light that can assume colors from a constant-sized set; robots explicitly communicate with each other using these colors. The colors are *persistent*; i.e., the color is not erased at the end of a cycle. Except for the lights, the robots are oblivious as in the classical model. Also, robots have *obstructed visibility* – robot b in a line of robots a, c blocks a (c) from seeing c (a).

Complete visibility. The fundamental COMPLETE VISIBILITY problem is defined as follows: Given any initial configuration of N autonomous mobile robots located at distinct points, they reach a configuration in which each robot is at a distinct point from which it can see all others (i.e., no three robots are collinear). Initially, some robots may be obstructed from the view of others (obstructed visibility). The robots do not know the total number of robots, N, and they do not have any agreement on a global coordinate system. Solving COMPLETE VISIBILITY enables solutions to many other robot coordination problems under obstructed visibility, e.g., gathering, pattern formation, and leader election. Indeed, after solving COMPLETE VISIBILITY, each robot is able to see all the robots, a scenario similar having unobstructed visibility. COMPLETE VISIBILITY has been receiving much attention recently. The objective is to simultaneously minimize (or provide trade-off between) two fundamental performance metrics: (i) Time to solve COMPLETE VISIBILITY and (ii) Number of distinct colors used by each robot light. A series of papers resulted a deterministic $\mathcal{O}(1)$-time, 47-color algorithm in the 2-dimensional Euclidean plane in the asynchronous setting [15].

Infinite grid. In this paper, we consider solving COMPLETE VISIBILITY in the robots with lights model on an *infinite grid* embedded in the 2-dimensional Euclidean plane that restricts the movements of each robot in each move to one of the four neighboring grid points. Grid setting naturally discretizes the 2-dimensional Euclidean plane and finds applications in real-life robot navigation systems, such as industrial Automated Guided Vehicles [10] and Coverage Path Planning [14]. The movement of robots along grid lines from one grid point to its adjacent point can be easier to implement for robots with weak capabilities as they may not be able to execute accurate movements in arbitrary directions or by arbitrarily small amounts [1]. Even with advanced robot navigation capabilities, the resolution of its plane of movement is finite, and consequently, a countable plane (rather than a real plane) better models the robots' movements; indeed, many algorithms for the real plane assume infinite resolution by, for example, relying on sufficient room to accommodate any number of robots between two given robots (regardless of how close they are to each other). Further, the asynchronous setting admits differences between individual robots, whether that is in the form of minor variations (in a homogeneous swarm) or large-scale differences (in a heterogeneous swarm).

Previous work. Adhikary *et al.* [1] gave the first (deterministic) algorithm for robots with lights to solve COMPLETE VISIBILITY on an infinite grid using 11 colors in the asynchronous setting. They proved the correctness but gave no runtime except a proof of finite time termination. Our analysis of their algorithm gives $\Omega(\max\{DN, N^2\})$

Algorithm	Time (in epochs)	Number of colors	Remarks
[1]	$\Omega(\max\{DN, N^2\})$ (**Th. 2**)	11	leader election not needed, deterministic
Lower bound	$\Omega(N)$ (**Th. 3**)	any	any
Th. 1	$\mathcal{O}(\max\{D, N\})$	17	if leader election not needed, deterministic
Th. 1	$\mathcal{O}(\max\{D, N\})$	50	if leader election needed, randomized

Table 1. COMPLETE VISIBILITY for $N \geq 1$ robots on an infinite grid in the asynchronous setting, where D is the diameter of the initial configuration. Our algorithm has three stages, Stage 1 to Stage 3. Stage 1 is needed if and only if a leader needs to be elected. If a leader is already provided (or identified or not needed), then Stage 1 is not needed. Irrespective of leader election, the time of our algorithm remains the same. Stage 1 uses 38 colors, Stage 2 uses 9 colors, and Stage 3 uses 3 colors. Therefore, if leader election is not needed, then the number of colors are 17 (sum colors for Stages 2 and 3, plus we need 5 colors to represent 4 different leaders and an initial color), and with leader election, the number of colors are 50. With leader election, the algorithm terminates with probability at least $1 - \frac{1}{2^{\max(D,N)}}$, and hence randomized.

runtime, where D is the diameter of the initial configuration. Our goal in this paper is to develop a fast COMPLETE VISIBILITY algorithm on an infinite grid.

Contributions. We consider the same robot model as in Di Luna *et al.* [5], namely, robots are oblivious except for a persistent light that can pick a color at a time from a constant-sized set. Robots can have their visibility obstructed by other robots in the line of sight, robots do not know the total number of robots, N, and the robots are disoriented (no agreement on global coordinate system). Moreover, the robot setting is *asynchronous* – they have no common notion of time and robots perform their LCM cycles at arbitrary times. The robots operate on an infinite grid, where each grid point connects to four other (neighboring) grid points, each at unit distance away (to its "North," "South," "East," and "West" in its local coordinate system). A single robot movement is restricted to be to one of its neighbors at unit distance. Two robots should not occupy the same grid point simultaneously (this would constitute a *collision*). Runtime is measured in *epochs* – the interval where every robot completes at least one LCM cycle. We prove the following main result which, to our knowledge, is the first algorithm with this runtime for COMPLETE VISIBILITY on an infinite grid.

Theorem 1 (main result). *For any initial configuration of $N \geq 1$ robots with lights on distinct points on an infinite grid,* COMPLETE VISIBILITY *can be solved in $\mathcal{O}(\max\{D, N\})$ time (where D is the diameter of the initial configuration) using*

- *17 colors through a deterministic algorithm if leader election is not required*
- *50 colors through a randomized algorithm which terminates with probability at least $1 - \frac{1}{2^{\max(D,N)}}$ if leader election is required (i.e., a leader needs to be identified)*

in the asynchronous setting without collisions.

We also prove the following two results which show the significance of Theorem 1:

a. A runtime of $\Omega(\max\{DN, N^2\})$ for the only previous deterministic algorithm of Adhikary *et al.* [1] for COMPLETE VISIBILITY on an infinite grid (**Theorem 2**).
b. A time lower bound of $\Omega(N)$ for COMPLETE VISIBILITY on an infinite grid (**Theorem 3**), which holds even if unlimited number of colors are available and leader election is not required (i.e., a unique leader is already provided or identified).

In summary, Theorem 1 improves significantly with respect to runtime compared to Adhikary *et al.* [1]. In fact, our algorithm is asymptotically time-optimal for $D = \mathcal{O}(N)$ (Theorem 3). Additionally, our time lower bound shows that irrespective of (i) whether leader election is needed or not and (ii) whether there is a limit of the number of colors, the time bound can not be better than $\mathcal{O}(N)$. Therefore, our result provides trade-off on the number of colors and nature of the algorithm (determinisitic/randomized) based on whether leader election is needed for being able to solve the problem optimizing time.

Techniques. The lower bound proof uses an argument based on the well-known pigeonhole principle. For the upper bound, our proposed algorithm has three stages, Stages 1–3, that execute one after another. Stage 1 is needed only when a leader needs to be identified (i.e., a leader is not provided already). Stage 1 (if executed) elects two robots as the first and second leaders satisfying some properties. Stage 2 moves all the robots to position them on consecutive grid points on the line connecting the leaders. Stage 3 moves all robots except the first leader to position themselves on grid points solving COMPLETE VISIBILITY. We will show that each stage finishes in $\mathcal{O}(\max\{D, N\})$ time, giving overall $\mathcal{O}(\max\{D, N\})$ runtime. For leader election, Stage 1 finishes with probability at least $1 - \frac{1}{2^{\max(D,N)}}$ (and hence the algorithm is in overall randomized) and Stages 2 and 3 are deterministic. For no leader election, Stage 1 is not needed and Stages 2 and 3 are deterministic (and hence the algorithm is in overall deterministic). The total number of colors used throughout the stages with leader election is 50 and without leader election is 17. Keys to Stage 1 are *corner moving* and *leader election* procedures that allow to elect two robots as leaders guaranteeing desired properties. Key to Stage 2 is a *line formation* procedure that positions all N robots on consecutive positions on a line connecting the leaders. Key to Stage 3 is a *placement* procedure that positions the robots in grid points guaranteeing COMPLETE VISIBILITY.

Remark on leader election requirement. Suppose robots have a compass, i.e., robots agree on North (top), South (bottom), East (right), and West (left) directions. This immediately provides the leader (pick the southmost robot; break tie with picking the leftmost among all southmost ones). Even in this case, we show the time lower bound of $\Omega(N)$ (which holds irrespective of colors, leader, etc. with argument based only on pure pigeonhole principle). This can again be done if robots agree only on one-axis (say North and South), picking all the Southmost robots as leaders. With leader not provided, Stage 1 elects leader in $\mathcal{O}(\max\{D, N\})$ time with probability at least $1 - \frac{1}{2^{\max(D,N)}}$. Note that since robots are indistinguishable, deterministic leader election is not possible. Therefore, after having leader(s), Stages 2 and 3 run deterministic. Even after having leader, Stage 2, if naively done sequentially, needs $\mathcal{O}(\max\{DN, N^2\})$ time, which we solve in $\mathcal{O}(\max\{D, N\})$ time through a smart parallel approach.

Closely related work. Many papers focused on solving COMPLETE VISIBILITY on the 2-dimensional Euclidean plane. Di Luna *et al.* [5] gave the first algorithm for robots with lights. Subsequent papers [4,13] minimized the number of colors. Recently, Vaidyanathan *et al.* [18] presented an algorithm with runtime $\mathcal{O}(\log N)$ using 12 colors in the fully synchronous setting. Sharma *et al.* [16] presented an algorithm with runtime $\mathcal{O}(1)$ using 12 colors in the semi-synchronous setting. Sharma *et al.* [15,17] then presented an algorithm with runtime $\mathcal{O}(1)$ using 47 colors in the asynchronous setting.

Furthermore, COMPLETE VISIBILITY on a grid has been considered in the literature as the NO-THREE-IN-LINE problem [6,8,9], where the goal is to select grid points from a $N \times N$ grid so that no three points among the selected ones are collinear. The maximum number of such grid points is $2N$ for $N \leq 46$, and for $N >> 46$, the best solution known is $(\frac{3}{2} - \epsilon)N$, for any $\epsilon > 0$ [9]. However, these works do not deal with relocating robots to place themselves on the grid points satisfying a NO-THREE-IN-LINE configuration. This becomes further challenging when (i) robots do not know N, (ii) robots and grid both are anonymous, and (iii) robots are disoriented.

Roadmap. Section 2 provides details on the model, some basic results, the runtime of the previous algorithm, and the runtime lower bound of $\Omega(N)$. We present our $\mathcal{O}(\max\{D, N\})$-time algorithm and its analysis in Section 3 proving Theorem 1. We then conclude in Section 4 with a short discussion.

2 Model and Preliminaries

Robots. We consider a system of N robots (agents) $\mathcal{Q} = \{r_0, r_1, \cdots, r_{N-1}\}$. Each robot is a (dimensionless) point that can move in an infinite 2-dimensional grid $\mathbb{Z}^2$ embedded in the plane. All the robots are initially positioned on grid points. The robots do not have access to any global coordinate system and they do not know N. Robots agree on the location of grid points, but each robot's orientation of up/down, left/right in the grid is local and is independent of the orientation of other robots. A robot r_i can see, and be visible to, another robot r_j iff there is no third robot r_k in the line segment joining r_i and r_j. Each robot has a light that can assume one color at a time from a constant-sized set. We will often use r_i to denote a robot as well as its position.

Robot movement. A robot's movement is restricted to grid lines. In other words, from a robot's current grid point, it can move only to one of the four neighboring grid points (that are a unit distance away) in one move. For simplicity in analysis, we assume that the moves are *instantaneous*, i.e., the robots are always seen at grid points (not on grid edges). Numerous grid and graph algorithms make this assumption (for instance, [1,7]). Our algorithm and analysis will work without this assumption as well as seen on an edge means that the move has not been completed and the robots can wait for that.

Look-Compute-Move. At any time, each robot r_i is either active or inactive. When a robot r_i becomes active, it performs the "Look-Compute-Move" cycle described as follows: (i) *Look:* For each robot r_j that is visible to it, r_i can observe the position of r_j on the grid and the color of the light of r_j. Robot r_i can also observe its own color and position. Each robot observes position in its own frame of reference. That is, two different robots observing the position of the same point may produce different coordinates. A robot observes the positions of points accurately within its own reference frame. (ii) *Compute:* Robot r_i may perform an arbitrary computation using only the colors and positions observed during the "look" portion of that cycle. This includes determination of a (possibly) new position (at a neighboring grid point) and light color for r_i for the start of the next cycle. (iii) *Move:* Robot r_i changes its light to the new color (if computed) and moves to its new position.

Robot activation and synchronization. In the fully synchronous setting ($\mathcal{FSYNC}$), every robot is active in every synchronized LCM cycle. In the semi-synchronous setting ($\mathcal{SSYNC}$), at least one robot is active in each synchronized LCM cycle, and over an infinite number of LCM cycles, every robot is active infinitely often. In the asynchronous

setting ($\mathcal{ASYNC}$), robots have no common notion of time. No limit exists on the number and frequency of LCM cycles in which a robot can be active except that every robot is active infinitely often. Complying with the $\mathcal{ASYNC}$ setting, we assume that a robot "wakes up" and performs its Look phase instantaneously. An arbitrary amount of time may elapse between Look/Compute and Compute/Move phases.

Runtime. For the $\mathcal{FSYNC}$ setting, time is measured in rounds. Since a robot in the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ settings could stay inactive for an indeterminate time, we use the idea of an epoch to measure time. An *epoch* is the smallest time interval within which each robot is active at least once [2]. Let t_0 denote the start time. Epoch i ($i \geq 1$) is the time interval from t_{i-1} to t_i where t_i is the first time instant after t_{i-1} when each robot has finished at least one complete LCM cycle. Therefore, for the $\mathcal{FSYNC}$ setting, a round is an epoch. We will use "time" generically to mean rounds for the $\mathcal{FSYNC}$ and epochs for the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ settings.

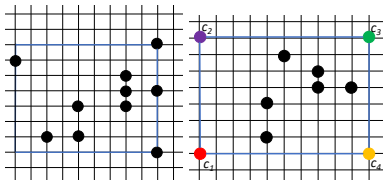

Fig. 1. Examples: (a) SER, (b) FCC

Smallest enclosing rectangle (SER) and four-corners configuration (FCC). A *smallest enclosing rectangle* (SER) of a configuration is an axis-aligned rectangle with minimum area such that all robots are on its perimeter or in its interior. A *four-corners configuration* (FCC) is a configuration in which one robot is at each of the four corners and the remaining $N - 4$ robots are in the interior. For $N = 10$, Fig. 1 shows a SER (left) and a FCC (right).

Line and line segment. For any pair of points a, b, we denote the line segment connecting them by $\overline{ab}$ and its length by length($\overline{ab}$). We denote a line with one endpoint a and passing through b by $\overrightarrow{ab}$. When we refer to grid points on $\overrightarrow{ab}$, we always refer to points on $\overrightarrow{ab}$ starting from a towards b (and possibly beyond b).

Configuration. A *configuration* $\mathbf{C}_t = \{(pos_i^t, col_i^t) \ : \ 0 \leq i < N\}$ specifies the robot positions pos_i^t and their colors col_i^t at any time $t \geq 0$. A configuration, $\mathbf{C}_t(r_i)$, for a robot $r_i \in \mathcal{Q}$ at time t is a restriction of $\mathbf{C}_t$ to those robots that are visible to r_i at t. Clearly, $\mathbf{C}_t(r_i) \subseteq \mathbf{C}_t$. We will sometimes write $\mathbf{C}, \mathbf{C}(r_i)$ to denote $\mathbf{C}_t, \mathbf{C}_t(r_i)$.

Basic results. The following results will be used in the algorithm in Section 3.

Lemma 1. *Let L_i be an axis-aligned grid line passing through robot $r_i \in \mathcal{Q}$ such that: (a) there is at least one robot on each side of L_i, and (b) r_i can see a robot colored* `red` *on only one side of L_i (with no visible* `red`*-robots on the other side). Let LP_i be a line perpendicular to L_i passing through r_i. Let p_{up} and p_{down} be the grid points on LP_i adjacent to r_i's position where p_{up} is on the side with* `red` *robots. Robot r_i can correctly move to p_{up} or p_{down} as needed.*

Proof. Since robots on one side of L_i are colored differently than the other side, r_i can differentiate the neighboring points p_{up} or p_{down} on LP_i as needed. □

Lemma 2. *Let L_i be an axis-aligned grid line passing through robot $r_i \in \mathcal{Q}$. Let LP_i be a line perpendicular to L_i passing through r_i. Let LP_f and LP_j be lines parallel to LP_i and on opposite sides of LP_i. Let L_α be a line parallel to L_i such that a robot*

each from LP_f and LP_j is on it with color different from other robots on those lines. Let there be no robots between LP_f and LP_i and between LP_i and LP_j on the side of L_i that includes L_α, and let there be no robot on LP_i or LP_j beyond L_α. Let p be the grid point on LP_i adjacent to r_i's position and on the side of L_i opposite that of L_α. Robot r_i can correctly move to p.

Proof. Since a robot r_i on LP_i always sees robots on LP_f and LP_j colored different from other robots, by Lemma 1, r_i can correctly move to point p. □

Lemma 3. *Let $\overline{AB}$ be an axis-aligned grid line joining two grid points A, B. Suppose $\mathsf{length}(\overline{AB}) = x$ units. For any $1 \leq y \leq x+1$, y robots placed arbitrarily on the points of $\overline{AB}$ can be relocated to be positioned consecutively on $\overline{AB}$ starting from A (or B) in $2x+2$ epochs.*

Proof. If there are $y = x+1$ robots on $\overline{AB}$, no relocation is necessary. If $y = 1$, then it needs x epochs. Suppose the only robot is at B and has to move to A. If there are $2 \leq y \leq x$ robots, then a robot may not be able to make its first move for $y-1$ epochs (suppose y robots are on consecutive positions). After that, it can move in each epoch as the robots are moving in the same direction and when the next point is empty and a robot moves into it, then the new next point will be empty in the next epoch as well. Therefore in total, $y \leq 2x+2$ epochs. □

Time complexity of the previous algorithm. Adhikary *et al.* [1] constructed the only previous algorithm for COMPLETE VISIBILITY on an infinite grid. They provided no runtime. We prove here the following theorem.

Theorem 2 (runtime of Adhikary *et al.* [1]). *The algorithm of Adhikary et al. [1] for* COMPLETE VISIBILITY *on an infinite grid has runtime $\Omega(\max\{DN, N^2\})$.*

Proof. Their algorithm has three phases, Phase 1 to Phase 3, that execute sequentially.

- **Phase 1 (interior depletion)** moves all the robots in the interior in the initial configuration to form a SER configuration so that all N robots are positioned on the boundary of SER. The robots already on the boundary of the SER do not move and the robots in the interior of the SER sequentially move to position themselves on the boundary of the SER. If all positions on the SER boundary are occupied, then the robots on the boundary of the SER move to the exterior to create empty positions. After all the robots positioned on the boundary of SER, Phase 1 finishes.
- **Phase 2 (corner creation)** If the four robots on the SER are not already on the four corner points of the SER, then the robots on the boundary of the SER move to place robots on the four corners of the SER. Phase 2 then finishes.
- **Phase 3 (symmetric movements)** moves the robots on the SER, two at a time (from each side of the SER), in the exterior of the SER to form a COMPLETE VISIBILITY configuration. After two robots (from a side of SER) are placed on their positions, the next two from that side will move, and so on. The moves of two robots from four sides of SER run independently in parallel. The four robots on the corners of the SER will not move, and they act as references to position the $N-4$ robots appropriately to achieve a COMPLETE VISIBILITY configuration. If

two robots moved simultaneously before are placed at distance d from the Phase 2 SER (in the exterior of the Phase 2 SER), the two robots moved simultaneously after are placed at distance $d' > d$.

We now discuss why the runtime is $\Omega(\max\{DN, N^2\})$. In Phases 1 and 3, there are configurations such that these phases need time $\Omega(DN)$ and $\Omega(N^2)$, respectively. For time $\Omega(DN)$ for Phase 1, consider the initial configuration of $\lfloor N/2 \rfloor$ robots in the interior on a line (that is not vertical or horizontal axis-aligned line of the grid and the remaining $\lceil N/2 \rceil$ robots on a SER. This gives $\lfloor N/4 \rfloor$ rectangle layers in the interior of SER with two robots serving as the corners of each rectangle layer. Suppose the distance between robots on the SER boundary to any robot in the interior of it is $D = \Omega(N)$. In this configuration, the robots in the interior of SER need to move to the SER boundary. Since the robots on a side move sequentially, only two robots on a rectangle layer in the interior of SER can move to SER at any round. Since there are $\lfloor N/2 \rfloor$ robots in the interior and the distance from each of them to G' is $\Omega(D)$, the total time is $\Omega(DN)$.

For time $\Omega(N^2)$ for Phase 3, consider a SER with $N/4$ robots on each side. Only two robots from each side move simultaneously at any time. After they are settled, then two other robots move. This way this process repeats $N/8$ times. Two robots settle at distance 1 from the Phase 2 SER, two robots settle at distance 2 from the Phase 2 SER, and so on. This way, the last two robots have to move $N/8$ distance. Therefore, the total time is $\geq (1 + 2 + \ldots + N/8) \geq N/8(N/8 + 1) \cdot 1/2 = \Omega(N^2)$. □

$\Omega(N)$ time lower bound. We prove an $\Omega(N)$ time lower bound for COMPLETE VISIBILITY on an infinite grid in the $\mathcal{FSYNC}$ (and hence the $\mathcal{ASYNC}$) setting). The lower bound construction is as follows (see Fig. 2). Suppose $\sqrt{N} \geq 4$ is an even integer. Consider an $\frac{N}{2} \times \frac{N}{2}$ axis-aligned sub-grid G having $N^2/4$ grid points within the infinite grid. Let P be the perimeter of G. Consider also a $\sqrt{N} \times \sqrt{N}$ axis-aligned sub-grid G' having N grid points with perimeter P'. Suppose G' is positioned in the middle of G, i.e., the distance L from any grid point of G' on its perimeter P' to the closest grid point of G on its perimeter P is exactly $\frac{N}{4} - \frac{\sqrt{N}}{2}$ (note that $\sqrt{N}$ is an even integer). Suppose N robots are positioned initially on the N grid points of G'. We show that at least one robot in G' needs to move $\Omega(N)$ times.

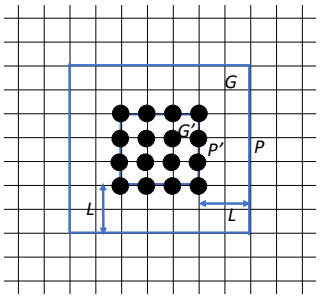

Fig. 2. Lower bound example

Theorem 3 (lower bound). *For the initial configuration above, any algorithm for* COMPLETE VISIBILITY *needs $\Omega(N)$ rounds in the $\mathcal{FSYNC}$ setting, which holds even if unlimited number of colors are available and a unique leader is provided.*

Proof. Consider any vertical (or horizontal) line segment L' connecting two perimeter points of G. On L', there will be exactly $\frac{N}{2}$ grid points of G. If there are three (or more) robots on L', then the robot in the middle obstructs the remaining two robots to be able to see each other. Therefore, in a complete visibility configuration, there cannot be more than two robots on any line segment L'.

In the construction above, there are exactly N robots on the grid points of G'. Furthermore, there are exactly $\frac{N}{2}$ vertical line segments and $\frac{N}{2}$ horizontal line segments in G, totaling N line segments. Applying the pigeonhole principle, we can place only at most $2 \times \frac{N}{2} = N$ robots on N line segments of G. The argument is that if we try to put $N' > N$ robots in G, then some vertical or horizontal line segment of G will contain three (or more) robots, violating the COMPLETE VISIBILITY condition.

With N robots placed on $\frac{N}{2}$ horizontal (vertical) line segments in G, there are exactly two robots on each line segment. Consider any vertical or horizontal line segment L'' on the perimeter P of G. Two robots of G' must move to L''. In the initial configuration, the distance from any robot on G' to L' is $L \geq \frac{N}{4} - \frac{\sqrt{N}}{2}$. Therefore, any robot in G' that moves to L'' needs to move at least $L \geq \frac{N}{4} - \frac{\sqrt{N}}{2} \geq \frac{N}{8} = \Omega(N)$ rounds to reach its position on L'', for any even $\sqrt{N} \geq 4$. Note also that the N robots in G' do not need to move to the nodes of G as described above, but if they move otherwise it will only increase the minimum number of rounds required.

In the pigeonhole principle argument above, it is not difficult to see that having unlimited number of colors and a unique leader provided does not have an impact. □

3 $\mathcal{O}(\max\{D, N\})$ Time Algorithm

In this section, we describe an $\mathcal{O}(\max\{D, N\})$ time algorithm for COMPLETE VISIBILITY on an infinite grid in the $\mathcal{ASYNC}$ setting for the robots with lights. For simplicity in the analysis, we first consider the cases of $D = \mathcal{O}(N)$. The algorithm and analysis extend immediately, replacing N by D, for cases where $D = \Omega(N)$. The algorithm consists of three stages, Stages 1–3 (see Fig. 3).

- **Stage 1 (leader election)** is to position robots on the corners of a FCC and elect two robots A and B on a side of the FCC to be leaders. A is elected first and is called the "first leader". B is elected after A and is called the "second leader". The remaining two robots C, D on a side of the FCC are also ranked first and second. Stage 1 is not needed if leader election capability is already provided or available.
- **Stage 2 (line formation)** moves the $N-2$ non-leader robots to position themselves on the consecutive grid points on line $\overrightarrow{AB}$ starting from A towards B, except one robot that will be positioned on $\overrightarrow{AB}$ at the grid point that is at distance $N'-1$ from A, where $N' \geq N$ is the first prime number greater than or equal to N.
- **Stage 3 (placement)** moves all robots (except A) from $\overrightarrow{AB}$ perpendicular to $\overrightarrow{AB}$ to position them on grid points forming a COMPLETE VISIBILITY configuration.

At the initial configuration $\mathbf{C}_{init}$, all robots have color `start`. Each robot r_i works autonomously having only the information $\mathbf{C}(r_i)$. Though robots are oblivious, the colors and configurations that the robots take synchronize execution of the stages so that robots execute stages sequentially one after another. The algorithm uses 50 colors and runs for a total of $\mathcal{O}(\max\{D, N\})$ epochs with probability at least $1 - \frac{1}{2^{\max(D,N)}}$ with leader election and uses 17 colors and runs for $\mathcal{O}(\max\{D, N\})$ epochs deterministically (with probability 1) without leader election.

We differentiate initial configurations $\mathbf{C}_{init}$ into two categories, collinear initial configurations $C_{col,init}$, and non-collinear initial configurations $C_{non,init}$. In both

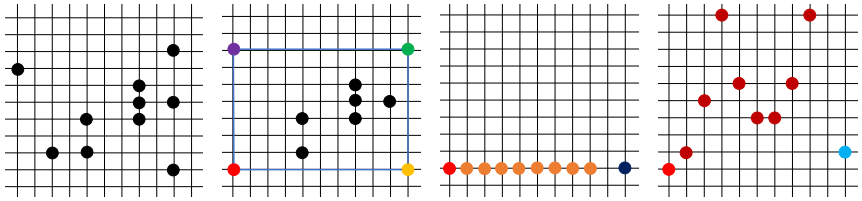

Fig. 3. A non-collinear initial configuration $C_{non,init}$ and the three stages of the algorithm: (left) $C_{non,init}$, (second left) after Stage 1, (third left) after Stage 2, and (right) after Stage 3. The configuration after Stage 3 is a COMPLETE VISIBILITY configuration.

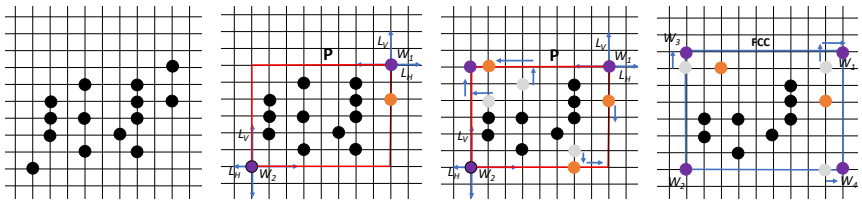

Fig. 4. An illustration of how an FCC is formed in Stage 1.1 starting from any $C_{non,init}$: (left) $C_{non,init}$, (middle two) during Stage 1.1, and (right) after Stage 1.1. $\mathbf{P}$ is a SER (rectangular convex hull) of $C_{non,init}$.

$C_{col,init}$ and $C_{non,init}$, Stage 3 is common. For $C_{col,init}$, Stages 1 and 2 can be executed together. Therefore, we describe Stages 1 and 2 for $C_{non,init}$ in Sections 3.1 and 3.2, respectively, Stages 1 and 2 together for $C_{col,init}$ in Section 3.3. Finally, we describe Stage 3 in Section 3.4 with correctness and runtime.

3.1 Stage 1 (Leader Election for Non-collinear Configurations)

The goal in Stage 1 is to form a four-corners configuration (FCC) among robots and to designate two of the corner robots A and B on one side as leaders. We first discuss how to elect A and B for any $C_{non,init}$ when $N \geq 4$. We then discuss a special case of $C_{non,init}$ when $N = 3$. We have two sub-stages, Stage 1.1 and Stage 1.2. In Stage 1.1, an FCC is formed with $N - 4$ robots being in its interior (Fig. 3(b)). In Stage 1.2, leaders A and B are elected among the four corner robots of the FCC. The remaining two robots C, D on the FCC are also ranked first and second.

Stage 1.1. Let $\mathbf{P}$ be a SER of the robots in $\mathcal{Q}$ in any given $C_{non,init}$ on an infinite grid. All the robots of $\mathcal{Q}$ are either on the perimeter of $\mathbf{P}$ or in the interior of $\mathbf{P}$ (Fig. 4(b)). A SER, $\mathbf{P}$, is an FCC if and only if there are exactly four robots on the four corners of $\mathbf{P}$ (and no other robot on the perimenter). The goal here is to form an FCC from the SER of $C_{non,init}$. A robot w is at a corner (or on a side) of the SER if all visible robots are within an axis-aligned angle of $90°$ (or $180°$ but not $90°$). In the following discussion, w will take actions based on sides and corners of an SER. These refer to the smallest enclosing rectangle of the configuration visible from w; this rectangle may be the same as or a subset of the SER of the entire configuration.

The algorithm is as follows for each robot $w \in \mathcal{Q}$.

- Let w be at a corner of the SER, then w changes its color to `rectangle-point`.
- Let w be a side robot on side S of the SER. If w sees a corner or side robot in one direction on S but nothing in the other direction, then it takes color `ready` (yellow

colored robot in Fig. 4(b)) and moves toward the empty corner. Suppose w sees no other robots on S and is closer to corner c_{near} of S than it is to corner c_{far} of S. Let I_S denote the set of interior robots closest to S. If I_S is empty or includes a robot whose projection to S is at equal or less distance to c_{far} than w, then w takes color `ready` and moves toward c_{near}; otherwise, w takes color `ready` and moves toward c_{far}. Now suppose w sees no other robots on S and is equidistant from both corners of S. If I_S has robots with projections to S in only one direction from w, then w takes color `ready` and moves toward the corner in the opposite direction; otherwise, w takes color `ready` and randomly chooses one corner and moves toward it.

- Let w be an interior robot closest to side S of the SER and at an extreme toward a corner of S of the set I_S of robots closest to S. If S has one corner robot but no side robots and w is the robot in I_S closest to the empty corner, then w moves toward S. If S has one side robot x but no corner robots and x is closer to corner c_{near} of S than it is to corner c_{far} of S, then w does the following. If the projection of w on S is closer to c_{far} than x is and w is closer to c_{far} than other robots in I_S, then w moves toward S. (Robot x will move toward c_{near} and w will move toward c_{far}.) If the projection of w on S is closer to c_{near} than x is and w is closer to c_{far} than other robots in I_S, then w moves toward S. (Robot x will move toward c_{far} and w will move toward c_{near}.) Fig. 4(c) illustrates these robots (yellow colored) moving to a side then toward a corner.
- Let w be a side robot not colored `ready`, in a configuration with no interior robots, and closest to side S of the SER that has only one side robot and no corner robots or only one corner robot and no side robots, then w moves into the interior. This puts w into a position from which it can later move to a corner of S.

If two robots were to head to the same rectangle point (one in horizontal direction and one in vertical direction) and they are at distance one from the rectangle point, then they use leader election to break symmetry (and avoid collision) so that one will continue moving towards the rectangle point. The robot that loses the leader election changes its color back to `start`.

If a robot w has color `rectangle-point` and sees a robot on one of its (axis-aligned) sides that does not have color `rectangle-point` (that is, a side robot), then w moves unit distance away from the side (toward the exterior) to be visible to the neighboring corner (see Fig. 4(d)).

For any $C_{non,init}$ with $N = 3$, each robot will figure out that there are only three robots in $\mathcal{Q}$ while executing Stage 1.1. The robots then terminate forming a triangular configuration.

Lemma 4. *At the end of Stage 1.1: (i) for $N = 3$, the robots terminate during Stage 1.1, (ii) for $N \geq 4$, the FCC is correctly formed with $N - 4$ robots in its interior. The corner robots of the FCC have color* `rectangle-point` *while the interior robots have color* `ready` *or* `start`*. Stage 1.1 finishes in $\mathcal{O}(N)$ epochs avoiding collisions.*

Proof. For the SER of the robot configuration (starting from $C_{non,init}$), consider each corner c. If a robot is at c, then it will take color `rectangle-point` and will remain at a corner of the SER. (It may move, but will remain at a corner of the new

configuration.) If no robot is at c, then consider one side S of that corner. If S has two or more side robots on it, then the one closest to c will take color `ready` and move toward c. It will either reach c and take color `rectangle-point` or it will lose a randomized selection to a robot colored `ready` on the adjacent side and that robot will reach c and take color `rectangle-point`. If c has no robot and S has only one side robot r, then if the corner at the other end of S has a robot then r will take color `ready` and move toward c as above, otherwise, an interior robot will move to S and the algorithm will proceed as above for two or more side robots. Each corner robot colored `rectangle-point` will move to see the two neighboring corners without being blocked by side robots. A robot will move at most twice after taking color `rectangle-point`. Collision avoidance is obvious since robots never move to the same grid point and towards each other (except for two robots moving to the same corner, and these use leader election to avoid collision). For runtime, $\mathcal{O}(N)$ epochs are enough since even if the robots move sequentially, only at most eight robots compete to become rectangle points and four are successful. □

Stage 1.2. After an FCC is formed, the goal is to elect two adjacent robots among the four on rectangle points of FCC as the first and second leaders, A and B, respectively. First, A is picked. After A, leader B is picked, which is a neighbor of A on SER. A is colored `leader1` and B is colored `leader2`. Two remaining rectangle points C, D are colored `defeated1` and `defeated2`, respectively. C (colored `defeated1`) is the neighboring rectangle point of A in FCC.

Leader Election. In the context of robots with lights, the *leader election* problem is as follows: Given a non-empty subset S of robots (indicated by their relative positions and light colors), select exactly one robot from S as the leader. On termination, the leader is colored `leader` and all other robots of S are colored `non-leader`.

Leader Election under Complete Visibility. Here we assume that all robots in the set S are visible to each other, so each competing robot is aware of $M = |S|$, the number of competing robots. The algorithm executes the following for each robot at each round.

- Toss a coin to select light color `leader` (resp., `non-leader`) with probability $\frac{1}{M}$ (resp., $1 - \frac{1}{M}$).
- If there is exactly one robot in S with color `leader`, then terminate; otherwise repeat the previous step.

It is well-known that the above algorithm terminates with a leader in $O(\sigma \log M)$ rounds with probability $1 - \frac{1}{M^\sigma}$. Conversion of this algorithm to the $\mathcal{ASYNC}$ model is simply a matter of using lights (states) to synchronize robots into simulated rounds. In each simulated round (see Fig. 5), a robot will cycle through a `ready` state, one of two "toss" states (`toss-L` and `toss-NL`) and one of two "wait" states (`wait-L` or `wait-NL`). From a "wait" state the robot either goes back to the `ready` state or terminates in one of two "terminal" states (indicated by lights `leader` and `non-leader`).

Robots start at the `ready` state. When a robot is in the `ready` state and it sees all robots in either the `ready` or one of the "toss" states, it tosses a coin (as indicated in the algorithm above) and goes to state `toss-L` (resp., `toss-NL`) with probability $\frac{1}{M}$ (resp., $1 - \frac{1}{M}$); these transition conditions are denoted by C_1 and C_2, respectively in Fig. 5. No robot proceeds from a "toss" state until all robots are either in a "toss" state

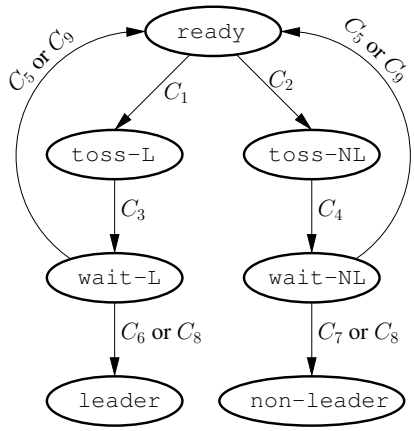

Fig. 5. State diagram for $\mathcal{ASYNC}$ leader election when competing robots are visible to each other. The conditions C_1–C_9 are explained in the text describing the algorithm.

or a "wait" state. When a robot in in a `toss-L` (resp., `toss-NL`) state and it sees all robots in either a "toss" state or a "wait" state, then it moves to state `wait-L` (resp., `wait-NL`); this is indicated by conditions C_3 and C_4 in Fig. 5. No robot proceeds from a "wait" state until all robots are in a "wait," "terminal" or `ready` state. Specifically, if a robot is in state `wait-L` (resp., `wait-NL`), and no robot is in a "toss" state, then it proceeds as follows. If there is a robot in state `ready`, then it too moves to state `ready` (Condition C_5 in Fig. 5). If there is a robot in a "terminal" state, then the robot moves to `leader` (resp., `non-leader`) (conditions C_6 (resp., C_7) in Fig. 5). If no robot is in `ready` or a "terminal" state, then the robot moves to state `leader` by condition C_8 (resp., `non-leader` by condition C_9) if it sees exactly one robot (including itself) in state `wait-L`; otherwise, it moves to state `ready` (condition C_9). The correctness of this algorithm follows from the fact that no new round starts until all robots have completed the current round. In fact, the time complexity of this $\mathcal{ASYNC}$ algorithm is the same as in the $\mathcal{FSYNC}$ model, with a round replaced by an epoch.

Lemma 5. *Leader election among a set of M robots that are visible to each other can be performed on the $\mathcal{ASYNC}$ model with lights in $O(\sigma \log M)$ epochs, with probability $1 - \frac{1}{M^\sigma}$.*

For this paper we will require a special case of Lemma 5 with $M = 2$. For an overall problem size of N and "with high probability (whp)" indicating a probability of at least $1 - N^{-c}$ for constant $c > 1$, we use $\sigma = c \log N$ to obtain the following result.

Corollary 1. *Leader election among a set of 2 robots that are visible to each other can be performed on the $\mathcal{ASYNC}$ model with lights in $O(\log N)$ epochs, with probability $1 - \frac{1}{N^c}$.*

Leader Election on a Non-Empty Rectangle. Here $S = \{0, 1, 2, 3\}$ is a set of four robots such that for each $0 \leq i < 4$, robot i can see itself and robots $(i \pm 1) (\mathrm{mod}\ 4)$. (In the following discussion we will write $i \pm 1$ to mean $(i \pm 1) (\mathrm{mod}\ 4)$. Also the index i itself is only for our reference; the robots themselves are anonymous.) We will call the

robots in set $\{i-1, i, i+1\}$ as the "neighbors" of i. As before, we consider the $\mathcal{FSYNC}$ case first. The basic algorithm is also the same as the one before (ALOHA-based).

The robots will start with color `ready`, flip a coin in the first round and color themselves accordingly. We will call this new color as the "toss color." Specifically, the robots begin by coloring themselves `zero` (or `one`) with probability $\frac{1}{4}$ (or $\frac{3}{4}$). If there is exactly one robot with toss color `one`, then that will (subsequently) become the leader.

In the second round, each robot i will generate a composite color (x_i, y_i), where $x_i \in \{\texttt{zero}, \texttt{one}\}$ is its toss color. The "view color" $y_i \in \{0, 1, 2, 3\}$ of robot i is the number of robots among its neighbors $i-1, i, i+1$ that have "toss color" `one`.

At the third round, each robot can determine whether a leader has been found or another iteration of coin-flips is needed. The following lemma develops the conditions to make this determination.

Lemma 6. For $S = \{0, 1, 2, 3\}$ and any robot $i \in S$, the number of robots in S with toss color `one` is 0 (resp., 1, ≥ 2) iff the highest view color among the neighbors of i is 0 (resp., 1, ≥ 2).

Proof. We first prove that if the total number of robots with toss color `one` is $c \in \{0, 1, \geq 2\}$ then the largest view color that robot i sees is c.

- Case $c = 0$: Every robot must have a view color of 0.
- Case $c = 1$: If robot i is the robot with toss color `one`, then robots $i-1, i, i+1$ all have a view color 1. If robot $i-1$ or $i+1$ is the robot with toss color `one`, then that robot and robot i both have a view color of 1 while the other robot of the pair has a view color of 0. If robot $i+2$ is the robot with toss color `one`, then robots $i-1$ and $i+1$ have view color 1 while robot i has view color 0. Across these instances, the largest view color that robot i sees is 1.
- Case $c \geq 2$: If at least two of the neighbors of robot i have toss color `one`, then i has a view color at least 2. If only one of the neighbors of robot i has toss color `one`, then $i+2$ must have toss color `one`. Then if $i-1$ or $i+1$ is the neighbor with toss color `one`, then that robot will have view color 2; otherwise, i is the neighbor with toss color `one`, so $i-1$ and $i+1$ will have view color 2. Across these instances, the largest view color that robot i sees is at least 2.

We now prove the converse that if the largest view color that robot i sees is $c \in \{0, 1, \geq 2\}$, then the total number of robots with toss color `one` is c. Again we consider some cases.

- Case $c = 0$: If robot i sees a maximum view color of 0, then neighbor $j \in \{i-1, i, i+1\}$ has view color $y_j = 0$, so each robot $k \in \{j-1, j, j+1\}$ must have a toss color of `zero`. With addition and subtraction modulo 4, the set $\{j-1, j, j+1 \ : \ i-1 \leq j \leq i+1 \text{ and } 0 \leq i < 4\} = \{i-2, i-1, i, i+1, i+2 \ : \ 0 \leq i < 4\} = \{0, 1, 2, 3\}$ is the entire set of robots under consideration.
- Case $c = 1$: Suppose robot i sees a maximum view color of 1. If i has a toss color of `one`, then each of robots $i \pm 1$ must have a toss color of `zero` and so must robot $i+2$; otherwise, the view color of $i \pm 1$ would be > 1. On the other hand, if i has a toss color of `zero`, then at most one of $i \pm 1$ has toss color `one`. If one does, then

$i+2$ has toss color `zero`; if neither does, then $i+2$ has toss color `one` because i sees a view color 1. These two observations imply that if i that sees a maximum view color of 1, then exactly one among the four robots of S has toss color `one`.

– Case $c > 1$: If a robot i sees a view color $c \geq 2$, then at least two of its neighbors or its neighbors' neighbors must have a toss color of `one`. This set $\{j, j+1, j-1 \ : \ i-1 \leq j \leq i+1\} = \{i-2, i-1, i, i+1, i+2\} = \{0, 1, 2, 3\}$ is the entire set of robots under consideration.

□

Lemma 6 establishes the criterion to terminate the leader election algorithm with an appropriate `leader` or `non-leader` flag or to iterate again. The extension of this algorithm to the $\mathcal{ASYNC}$ model follows the idea of Fig. 5 with the wait states replaced by the states corresponding to the composite colors.

Lemma 7. *By the end of Stage 1.2, two robots A, B on adjacent rectangle points of an FCC are elected as the first (A) and the second (B) leaders and colored* `leader1` *and* `leader2`*, respectively, and the rest two robots C, D are colored* `defeated1` *and* `defeated2`*, respectively. Stage 1.2 finishes in $\mathcal{O}(\log N)$ epochs avoiding collisions.*

3.2 Stage 2 (Line Formation for Non-collinear Configurations)

At the end of Stage 1, A, B, C, D are the rectangle grid points (or robots) of an FCC colored `leader1`, `leader2`, `defeated1`, and `defeated2`, respectively. The point pairs (A, B), (B, D), (A, C) and (C, D) share sides of the FCC. All remaining $N-4$ robots are in the interior of the FCC colored `start` or `ready` (see Fig. 3(b)). The goal in Stage 2 is to position all N robots on $\overrightarrow{AB}$ at distances $0, 1, 2, \ldots, N-2$, and $N'-1$ from A, where $N' \geq N$ is the smallest prime number greater or equal to N (Fig. 3(c)). Robot A does not move and robot B will be at distance either $N'-1$ or one of $1, 2, \ldots, N-2$. Note that N is not known to the robots.

A simple approach is to move robots sequentially. All robots on a parallel line closest to $\overrightarrow{AB}$ first move to $\overrightarrow{AB}$. After that the robots on the next parallel line move to $\overrightarrow{AB}$, and so on. This approach, however, needs $\mathcal{O}(N^2)$ time (note we consider $D \leq N$; with $D > N$, the bound becomes $\mathcal{O}(DN)$); at most $N-2$ lines have at least a robot on each, and the robots on each line need to traverse distance $\mathcal{O}(N)$ to be positioned on $\overrightarrow{AB}$. This will make the overall runtime $\mathcal{O}(N^2)$. Therefore, the challenge is to devise an approach that finishes Stage 2 in $\mathcal{O}(N)$ epochs.

Our approach runs in seven sub-stages, Stages 2.1–2.7, each taking $\mathcal{O}(N)$ time, giving overall $\mathcal{O}(N)$ runtime for the algorithm. Fig. 6 illustrates the ideas behind each sub-stage. The main challenge we address here is how to make moves in parallel.

We need some notations. For a robot w, let L_H (L_V) denote the horizontal (vertical) line through it, in w's local coordinate system. Without loss of generality, the description below refers to L_H when describing a condition that could apply to either line. Moreover, let L_0^A (L_0^B) be the line perpendicular to $\overrightarrow{AB}$ passing through A (B). Let the length of the line segment $\overline{AB}$ joining leaders A and B be $\mathsf{length}(\overline{AB}) = m$ units, i.e., there are $m-1$ grid points on $\overrightarrow{AB}$ between A and B. Let L_i^A be the line perpendicular

to $\overrightarrow{AB}$ passing through the grid point on $\overrightarrow{AB}$ at distance i from A. Since $1 \leq i < m$, there will be $m-1$ perpendicular lines between A and B, with L_1^A closest to L_0^A and L_{m-1}^A closest to L_0^B. Let $\mathsf{length}(\overline{AC}) = n$ units. Let LP_i^A be a line parallel to $\overrightarrow{AB}$ at distance i from $\overrightarrow{AB}$ towards $\overrightarrow{CD}$. LP_1^A is closest to $\overrightarrow{AB}$ and LP_{n-1}^A is closest to $\overrightarrow{CD}$.
Stage 2.1. All the robots in the interior of the FCC are on the grid points of lines $L_1^A, \ldots, L_{m-1}^A$. Each line $L_i^A, 1 \leq i \leq m-1$, has no, one, or more robots positioned on its grid points. Let those be called no-robot, one-robot, or multi-robot lines, respectively, depending on the number of robots on them. Stage 2.1 moves the robots on one-robot lines to position them on $\overrightarrow{AB}$ (Fig. 6(b)) in $\mathcal{O}(N)$ epochs. The flow of Stage 2.1 is to sweep down each occupied row of the configuration until reaching $\overrightarrow{AB}$, moving each robot on a one-robot line down one row and coloring each robot on a multi-robot line as `multi-robot`. Let w be a robot with color `start` or `ready` that sees only colors `defeated1`, `defeated2`, or `multi-robot` on one side of L_H. If w is the only robot on L_V and is not at a side of the visible SER, then it moves perpendicularly to L_H in the direction opposite of the side with colors `defeated1`, `defeated2`, or `multi-robot` (Lemma 1). If w is the only robot on L_V and is at a side of the visible SER, then it changes color to `on-AB`. If w is not the only robot on L_V, then it changes color to `multi-robot`. Stage 2.1 finishes when all robots on one-robot lines reach $\overrightarrow{AB}$ and assume color `on-AB`.

Lemma 8. *During Stage 2.1, the robots in the interior of the FCC that are on one-robot lines move to grid points on $\overrightarrow{AB}$ and take color* `on-AB`. *The remaining interior robots, on multi-robot lines, take color* `multi-robot` *and do not move. Stage 2.1 finishes in $\mathcal{O}(N)$ epochs avoiding collisions.*

Proof. Let LP_β^A be the line closest to $\overrightarrow{CD}$ that has (at least) a robot on it at the start of Stage 2.1. Let w be a robot on LP_β^A. The robots from $\overrightarrow{AB}$ to $LP_{\beta-1}^A$ are colored different than C, D, hence from Lemma 1, w can move to $LP_{\beta-1}^A$ if it is on a one-robot line or it can take color `multi-robot` if it is on a multi-robot line. Robot w can wait on $LP_{\beta-1}^A$ until there is no robot on LP_β^A or all on it are colored `multi-robot`. After this, the conditions on Lemma 1 are again satisfied for robots on $LP_{\beta-1}^A$, and they can move to $LP_{\beta-2}^A$ or take color `multi-robot`. This process then continues until all robots on one-robot lines reach $\overrightarrow{AB}$ and all robots on multi-robot lines have color `multi-robot`. Regarding runtime, since we assume $D = \mathcal{O}(N)$, any side of the FCC is $\mathcal{O}(N)$ long, and hence w needs to move for $\mathcal{O}(N)$ epochs to reach $\overrightarrow{AB}$. Since the moves of robots on one-robot lines are happening in parallel (with synchronization between two consecutive lines finishing in one epoch), Stage 2.1 finishes in $\mathcal{O}(N)$ epochs. Collisions are avoided since there is no other robot on the one-robot lines. □

Stage 2.2. At the beginning of Stage 2.2, all robots in the interior of the FCC are positioned on multi-robot lines and have color `multi-robot`. Let LP_β^A be the line closest to $\overrightarrow{CD}$ that has at least one robot colored `multi-robot` on it. For all the multi-robot lines that do not have a robot on LP_β^A, Stage 2.2 moves one robot each from those lines to position it on LP_β^A. At the end of Stage 2.2, the interior robots on LP_β^A have color

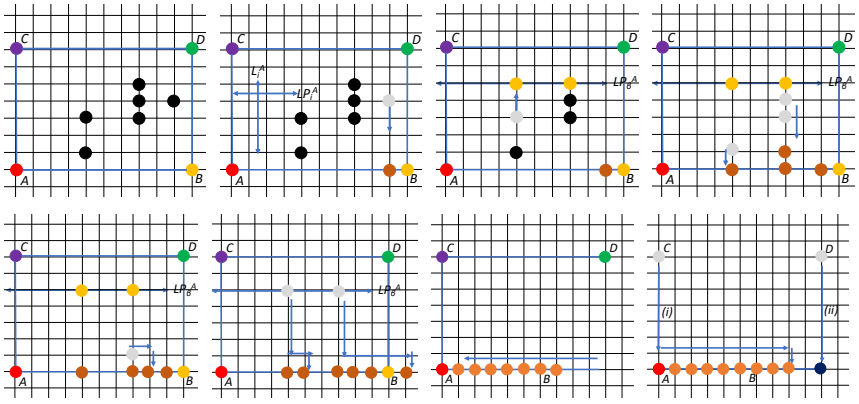

Fig. 6. An illustration of the configuration after Stage 1 and the seven sub-stages of Stage 2: (top left) after Stage 1, (top second left) after Stage 2.1, . . ., (bottom right) after Stage 2.7. The configuration shown for each sub-stage is achieved at the end of that sub-stage.

`multi-robot-2`, while the other interior robots have color `multi-robot-1`. The flow of Stage 2.2 is to sweep up each occupied row of the FCC from $\overrightarrow{AB}$ to LP_β^A. Stage 2.2 is done as follows. Let w be a robot with color `multi-robot` that sees only colors `multi-robot-1`, `leader1`, or `leader2` on one side of L_H. Call this side of L_H as the *bottom* side, and call the other side of L_H as the *top* side. Robot w checks the following conditions to determine whether to perform the corresponding action.

(a) If it sees a robot colored `multi-robot` on L_V in the top side, then w changes its color to `multi-robot-1` without moving.
(b) If it sees one or more robots colored `multi-robot` in the top side but no such robot on L_V in that side, then it moves perpendicularly to L_H into the top side, keeping its current color `multi-robot`. Robot w is now unit distance closer to LP_β^A.
(c) If it sees no robot in the top side except one each colored `defeated1` and `defeated2`, then w changes its color to `multi-robot-2` without moving. Robot w is in fact positioned on LP_β^A.

Case (c) makes sure that after w reaches LP_β^A, it stops moving and this way guaranteeing the end of Stage 2.2 for each robot on multi-robot lines. Fig. 6(c) illustrates what is achieved after Stage 2.2.

Lemma 9. *Let LP_β^A be the line parallel and closest to line $\overrightarrow{CD}$ of the FCC that has at least a robot colored* `multi-robot` *positioned on it at the end of Stage 2.1. At the end of Stage 2.2, a robot colored* `multi-robot-2` *from each multi-robot line is positioned on LP_β^A. The other robots take color* `multi-robot-1` *(without moving). Stage 2.2 finishes in $\mathcal{O}(N)$ epochs avoiding collisions.*

Proof. Stage 2.1 swept toward $\overrightarrow{AB}$, whereas Stage 2.2 sweeps toward $\overrightarrow{CD}$. At first, the robots closest to $\overrightarrow{AB}$ see the robots on one side colored differently than the robots on the other side. This satisfies the conditions of Lemma 1, so they either move keeping their color `multi-robot` or change their color to `multi-robot-1` without moving.

This continues for each occupied row. After reaching LP^A_β, a robot only sees robots C, D and can change its color to `multi-robot-2` and stay on LP^A_β. Collisions are avoided since only one robot on each multi-robot line moves and the moving robot is the one that is closest to LP^A_β among the robots on that line. Regarding runtime, the robots on each line can move independently in parallel, and in $\mathcal{O}(N)$ epochs, they reach LP^A_β, since any side of the FCC is $D \leq \mathcal{O}(N)$. □

Stage 2.3. At the beginning of Stage 2.3, one robot from each multi-robot line is positioned on LP^A_β, the line closest to $\overrightarrow{CD}$ that has robot(s) in it, and has color `multi-robot-2`. Stage 2.3 moves all the robots on each multi-robot line L^A_k, except on LP^A_β, to position them on consecutive positions on L^A_k starting from $\overrightarrow{AB}$ (including the grid point on $\overline{AB}$). Fig. 6(d) illustrates what is achieved at the end of Stage 2.3. Note that in the beginning of Stage 2.3, the position on $\overline{AB}$ of each multi-robot line L^A_k is empty so a robot on L^A_k can be positioned on it.

Consider a multi-robot line L^A_k. Let L^A_j and L^A_l be two multi-robot lines closest to L^A_k, one on each side of L^A_k. Notice that all these lines are perpendicular to $\overrightarrow{AB}$. If L^A_k has no other multi-robot line on one or both sides of it, it can use $\overline{AC}$ (or $\overline{BD}$) as L^A_j (or L^A_l). Notice that the robots on L^A_k see all the robots on both L^A_j and L^A_l, in particular the robots with color `multi-robot-2`. They will use the robots colored `multi-robot-2` to orient themselves, determining the direction of $\overline{AB}$.

Stage 2.3 is done as follows. Let robot w on L^A_k have color `multi-robot`. If its neighbors on the line perpendicular to L^A_k have color `on-AB`, `leader1`, or `leader2`, then w changes its color to `on-AB`, as it has reached $\overline{AB}$. Otherwise, if the next grid point towards $\overline{AB}$ is empty, then w moves to that grid point, but if that grid point has a robot with color `multi-robot-consecutive` or `on-AB`, then w changes its color to `multi-robot-consecutive`.

Let robot w have color `multi-robot-2`. If all visible robots toward $\overline{AB}$ have color `multi-robot-consecutive`, `on-AB`, `leader1`, or `leader2`, then w changes its color to `multi-robot-3`. This color change initiates Stage 2.4. Note that all these robots are still on LP^A_β.

Lemma 10. *At the end of Stage 2.3, all the robots on multi-robot lines, except those colored* `multi-robot-3`*, are positioned on those lines in consecutive positions starting from* $\overrightarrow{AB}$ *and colored* `multi-robot-consecutive` *or* `on-AB`*. The robots that started with color* `multi-robot-2` *have changed color to* `multi-robot-3` *without moving. Stage 2.3 finishes in* $\mathcal{O}(N)$ *epochs avoiding collisions.*

Proof. Consider the robots on L^A_k. Each robot on L^A_k sees the robots on L^A_j and L^A_l colored `multi-robot-2` (`multi-robot-2` colored robots are positioned on LP^A_β). Since the robots on LP^A_β do not move in this sub-stage, the robots on L^A_k see the robots on L^A_j and L^A_l colored `multi-robot-2` at all times. Now, each robot w on L^A_k satisfies the conditions of Lemma 2 and hence they can move on L^A_k toward $\overline{AB}$. This continues until w is next to a robot with color `multi-robot-consecutive` or `on-AB` when it stops and takes color `multi-robot-consecutive` or until w reaches $\overline{AB}$ when it stops and takes color `on-AB`. Collisions are avoided since each

robots moves only if the neighboring grid point is empty. Runtime is $\mathcal{O}(N)$ rounds since there are at most $N-4$ robots on a multi-robot line and since $D = \mathcal{O}(N)$, they can relocate to consecutive positions in $\mathcal{O}(N)$ epochs (Lemma 3). The robots colored `multi-robot-2` can color `multi-robot-3` since after the robots on L_k^A moved to consecutive positions, a robot colored `multi-robot-2` on L_k^A sees the closest robot on L_k^A colored `multi-robot-consecutive`. □

Stage 2.4. At the beginning of Stage 2.4, all the robots in the interior of the FCC are positioned as follows for each multi-robot line: one robot is on LP_β^A colored `multi-robot-3`, one-robot is on $\overline{AB}$ colored `on-AB`, and the rest of the robots are on consecutive positions on those lines starting from $\overline{AB}$ colored `multi-robot-consecutive` (Fig. 6(d)).

Stage 2.4 moves all the robots colored `multi-robot-consecutive` to position them on $\overrightarrow{AB}$ colored `on-AB`. Note that some of these robots may be past B on $\overrightarrow{AB}$. The flow of this stage is for each multi-robot line to pipeline its robots down to LP_1^A then along this line in the direction from A to B until reaching an empty grid point on $\overrightarrow{AB}$. The robot will then drop onto $\overrightarrow{AB}$ and take color `on-AB`. The multi-robot lines do this in order starting from the line closest to B. Each robot with color `multi-robot-consecutive` can orient itself to recognize the direction toward $\overrightarrow{AB}$ by using robots colored `multi-robot-3`. Each robot on LP_1^A can recognize that fact because it can see robots with color `leader1` (A) and `leader2` (B).

Let w be a robot with color `multi-robot-consecutive`. If w is not on LP_1^A, then if the adjacent grid point toward $\overline{AB}$ is open, then w moves to that grid point. If w is on LP_1^A, then w checks if (i) the adjacent grid point on LP_1^A in the direction from A to B is open and (ii) no robot with color `multi-robot-consecutive` is present in the quadrant defined by the side of LP_1^A opposite to $\overline{AB}$ and the side of the line through it perpendicular to $\overline{AB}$ that is opposite from A. If these conditions are satisfied, then w changes its color to `moving-to-AB` and moves to that open grid point. This is the circumstance when all multi-robot lines toward B from w have already pipelined themselves onto LP_1^A and it is the turn of w's multi-robot line.

Let w be a robot with color `moving-to-AB`. If the adjacent grid point on $\overrightarrow{AB}$ is open, then change color to `on-AB` and move to that point. If the adjacent grid point on $\overrightarrow{AB}$ is not open and the adjacent grid point on LP_1^A in the direction from A to B is open, then w moves to that open point.

This process then continues for all multi-robot lines of Stage 2.3 with at least three robots on them (for a multi-robot-line with two robots, a robot reaches $\overrightarrow{AB}$ in Stage 2.3 and one is still on LP_β^A colored `multi-robot-3`). After reaching $\overrightarrow{AB}$, they assume color `on-AB`. Fig. 6(e) illustrates what Stage 2.4 achieves. Some of the robots that moved to $\overrightarrow{AB}$ during this substage may have moved beyond B, so the overall configuration may no longer be an FCC (though a SER still exists).

Lemma 11. *Stage 2.4 takes robots colored* `multi-robot-consecutive` *and positions them on* $\overrightarrow{AB}$ *with color* `on-AB`. *Stage 2.4 finishes in* $\mathcal{O}(N)$ *epochs avoiding collisions.*

Proof. The process starts from the multi-robot line closest to B and goes toward A. The robots on each line move to LP_1^A in a pipelined fashion (Lemma 3). After reaching LP_1^A, the robots see both A, B and can determine the direction to move, always in the direction from A to B. Collisions are avoided since robots only move if their next grid point is empty. Runtime of $\mathcal{O}(N)$ is also immediate. While some robot has color `multi-robot-consecutive`, one such robot changes color to `moving-to-AB` and starts moving along LP_1^A at least every other epoch. Also, with this pipelined movement along LP_1^A, each robot needs to move at most $\mathcal{O}(N)$ distance to find the first empty grid point on $\overrightarrow{AB}$ to jump from LP_1^A and take color `on-AB`. This is because $\mathsf{length}(\overline{AB}) = \mathcal{O}(N)$ and there are N robots. So within $\mathcal{O}(N)$ distance from A on $\overrightarrow{AB}$, all robots moving on Stage 2.4 are accommodated. □

Stage 2.5. At the beginning of Stage 2.5, the only robots in the interior of the SER are on LP_β^A with color `multi-robot-3`. Each such robot w moves toward $\overline{AB}$ until it reaches LP_1^A then waits until all other robots of LP_β^A reach LP_1^A. When all are at LP_1^A, these robots see both C, D and the robots on $\overrightarrow{AB}$. Robot w then moves on LP_1^A in the direction of A toward B whenever the next grid point is empty in a pipelined fashion then changes color to `on-AB` and jumps to $\overrightarrow{AB}$ whenever the first empty grid point is available on $\overrightarrow{AB}$. Fig. 6(f) illustrates this sub-stage.

Lemma 12. *Stage 2.5 moves all the robots colored* `multi-robot-3` *on* LP_β^A *to* $\overrightarrow{AB}$ *and colors them* `on-AB`*. Stage 2.5 finishes in* $\mathcal{O}(N)$ *epochs avoiding collisions.*

Proof. The robots on LP_β^A can move to LP_1^A in $\mathcal{O}(N)$ epochs. After reaching LP_1^A, they see both A, B at all times and can move on LP_1^A in the direction from A towards B in a pipelined fashion until the first empty grid point on $\overrightarrow{AB}$ is found to jump and color `on-AB`. This whole process again takes $\mathcal{O}(N)$ epochs. Collisions are avoided by not moving to the next point until it is empty. □

Stage 2.6. At the beginning of Stage 2.6, no robots are in the interior of the SER. The robots on $\overrightarrow{AB}$ may not be in consecutive positions starting from A. The goal in this stage is to move them on $\overrightarrow{AB}$ towards A so that they will occupy consecutive positions on $\overrightarrow{AB}$. Fig. 6(g) illustrates what this sub-stage achieves.

Let w be a robot on $\overrightarrow{AB}$ with color `on-AB` or `leader2`. It can see C and D and so determine the directions of A and B. If the adjacent grid point toward A on $\overrightarrow{AB}$ is empty, then w moves to that grid point. If the adjacent grid point toward A on $\overrightarrow{AB}$ is occupied by a robot with color `on-AB-consecutive` or `leader1`, then w changes color to `on-AB-consecutive`. The coloring process starts from A towards B and after the farthest robot on $\overrightarrow{AB}$ takes color `on-AB-consecutive`, Stage 2.6 finishes.

Lemma 13. *Stage 2.6 relocates the robots on* $\overrightarrow{AB}$ *to consecutive points of* $\overrightarrow{AB}$ *with color* `on-AB-consecutive`*. Stage 2.6 finishes in* $\mathcal{O}(N)$ *epochs avoiding collisions.*

Proof. The correctness is immediate. There are $N - 2$ robots on $\overrightarrow{AB}$ at the beginning of Stage 2.6. Let B' be the robot that is farthest from A on $\overrightarrow{AB}$. B' knows that it is the

farthest. The total length of $\overline{AB'}$ from A until B' is at most $\mathcal{O}(N)$. Therefore. moving the robots toward A takes $\mathcal{O}(N)$ epochs since robots move in a pipelined fashion (Lemma 3). Collisions are avoided similarly as in previous sub-stages by not moving until the next grid point is empty. □

Stage 2.7. At the beginning of Stage 2.7, the robots on $\overrightarrow{AB}$ are in consecutive positions and only C, D are not on $\overrightarrow{AB}$. In this stage, C first moves to position itself on $\overrightarrow{AB}$ next to a robot colored `on-AB-consecutive`. D remains in its position until this is done. After that D can count the total number of robots, N, in the system. It computes the first prime number $N' \geq N$. It then moves first to line LP_1^A and then along line LP_1^A until the neighboring grid point on $\overline{AB}$ is at $N'-1$ distance away from A. It then jumps to $\overrightarrow{AB}$ and assumes color `endpoint1`. All N robots are now on $\overrightarrow{AB}$, with A colored `leader1`, the robot at distance $N'-1$ colored `endpoint1`, and the remaining $N-2$ have color `on-AB-consecutive`. All robots are in consecutive positions with the possible exception of the robot colored `endpoint1`. Fig. 6(h) illustrates this sub-stage. Stage 2.7 (and hence Stage 2) then finishes.

Lemma 14. *At the end of Stage 2, $N-1$ robots are on $N-1$ consecutive positions on $\overrightarrow{AB}$ starting from A, and one robot is on $\overrightarrow{AB}$ at distance $N'-1$ from A where $N' \geq N$ is the first prime number $\geq N$. Stage 2.7 finishes in $\mathcal{O}(N)$ epochs avoiding collisions.*

Proof. Similarly as in Lemma 13, C can move to $\overrightarrow{AB}$ in $\mathcal{O}(N)$ epochs. D can compute N as it sees all the robots on $\overrightarrow{AB}$. D can also compute what is its $N'-1$ position on $\overrightarrow{AB}$ until it is on $\overrightarrow{AB}$, which happens at the end when it jumps to the $(N'-1)$-th grid point from A towards B. Collision avoidance is immediate. □

3.3 Stages 1 and 2 (Leader Election and Line Formation for $C_{col,init}$)

We discussed in Sections 3.1 and 3.2 Stages 1 and 2 for $C_{non,init}$. For $C_{col,init}$, Stages 1 and 2 can be simplified significantly, which we do in this section. We first discuss how to handle collinear $C_{col,init}$ when $N \geq 4$. Let A, B be the endpoint robots on $C_{col,init}$. Let $r_1, \ldots, r_{N-2}$ be the $N-2$ robots between A and B on $\overline{AB}$, with r_1 closest to A and r_{N-2} closest to B. We have that $\mathsf{length}(\overline{AB}) = \mathcal{O}(N)$, since we are assuming that $D = \mathcal{O}(N)$. Robots A, B pick color `ready`. They do not move. The robots $r_1, \ldots, r_{N-2}$ do nothing. After A is colored `ready`, r_1 picks color `collinear-moving` and moves to either of the two neighboring grid points not on $\overline{AB}$. Robot r_{N-2} also does the same after B is colored `ready`. Let L_{up}, L_{down} be the two lines parallel to $\overline{AB}$ at unit distance each. Robots r_1, r_{N-2} reach to either of them, independently, after they move, so they may both be on the same line or one may be on L_{up} and the other on L_{down}. After that, each robot $r_2, \ldots, r_{N-3}$ picks color `collinear-moving` and moves to position itself on either of L_{up} or L_{down}.

After $r_1, \ldots, r_{N-2}$ move from $\overline{AB}$, A sees B (and vice-versa). Now A, B use the leader election procedure in Stage 1.2 to elect one as the first leader colored `leader1`. The other becomes the second leader colored `leader2`. Suppose A is the first leader colored `leader1` and B is the second leader colored `leader2`. A now elects two leaders C, D colored `defeated1` and `defeated2`, respectively, as follows. Let L_V

be the line perpendicular to $\overline{AB}$ closest to A such that it has at least a robot on it. The single robot on L_V colors itself `defeated1`, and we call it C. After that, the robot next to C on line L_{up} or L_{down} where C belongs colors itself as `defeated2`, and we call it D. D is already further away from A than C; D does not need to move. If no other robots are on L_{up} or L_{down} with C, then the closest robot on the opposite line takes color `defeated2` and moves perpendicularly to $\overline{AB}$ to the same line as C; call this robot as D.

Suppose both C, D are on L_{up} after all this (C, D on L_{down} is analogous). The robots on L_{down} pick color `on-AB-1` and move to $\overline{AB}$. The robots on L_{up} except C, D also move to $\overline{AB}$ picking color `on-AB-1`. After this, all $N-2$ robots except C, D are on $\overline{AB}$. This provides a scenario similar to the one that is achieved after Stage 2.5 (Section 3.2). The robots then execute Stages 2.6 and 2.7 to reach the configuration achieved after Stage 2.7 in Section 3.2. The resulting configuration is such that all N robots are on $\overrightarrow{AB}$ with A colored `leader1`, the robot at distance $N'-1$ from A is colored `endpoint1`, and the remaining $N-2$ robots are colored `on-AB-consecutive`.

We now discuss $C_{col,init}$ with $N \leq 3$. For $N = 3$, there is exactly one robot r_1 between A and B on $\overline{AB}$. A, B do not move. Therefore, when r_1 moves after it sees A or B colored `ready`, a non-collinear triangle configuration is achieved. A, B see each other and only one robot on L_{up} and L_{down} combined and hence terminate. r_1 can also terminate since it sees only A, B. If $N = 2$, A sees the light of the only other robot B `ready` and terminates. When $N = 1$, the only robot A can simply terminate since it sees no other robot.

Lemma 15. *For any $C_{col,init}$, the approach above achieves the following: (i) For $N = 1, 2$, robot(s) terminate without moving, (ii) for $N = 3$, the robots terminate in a triangular configuration; and (iii) for $N \geq 4$, the robots are positioned on a line $\overrightarrow{AB}$ satisfying Lemma 14. Runtime is $\mathcal{O}(N)$ epochs and no collisions.*

Proof. The correctness proof is immediate from the description and the proofs of Stages 2.6 and 2.7. Regarding runtime, the process until selecting A, B, C, D as leaders finishes in $O(1)$ epochs. The robots on L_{up} and L_{down}, except C, D move back to $\overline{AB}$ again in $\mathcal{O}(1)$ epochs. After that moving the robots already on $\overline{AB}$ to consecutive positions and after that placing C, D on $\overline{AB}$ using the techniques of Stages 2.6 and 2.7, respectively, takes $\mathcal{O}(N)$ epochs. Thus, overall, for any $C_{col,init}$, this approach finishes in $\mathcal{O}(N)$ epochs. Collision avoidance is also immediate. □

3.4 Stage 3 (Placement at COMPLETE VISIBILITY Points)

The configuration at the end of Stage 2 has all N robots on $\overrightarrow{AB}$, with A colored `leader1`, the robot at distance $N'-1$ colored `endpoint1`, and the remaining $N-2$ robots colored `on-AB-consecutive`. Let $w_i \neq A$ denote the robot in this configuration on $\overrightarrow{AB}$ at distance i, where $2 \leq i \leq N-2$ or $i = N'-1$. The goal in Stage 3 is to reposition the robots that are on $\overrightarrow{AB}$ at the end of Stage 2 to guarantee a COMPLETE VISIBILITY configuration. Let $(0, 0)$ denote the coordinates of A, so the initial coordinates of w_i are $(i, 0)$. We will establish that positioning each robot w_i at the target point of $(i, i^2 \mod N')$ achieves COMPLETE VISIBILITY.

Observe that the target points of A and $w_{N'-1}$ are $(0,0)$ and $(N'-1,1)$, respectively. Robots will use A and $w_{N'-1}$ at their target points to orient themselves and determine N'. Let LP_1^A denote the line parallel to $\overrightarrow{AB}$ on one side to be determined. The flow of Stage 3 is to move all robots, other than A, first to LP_1^A then to sweep up, one row at a time, with each robot stopping at its target point.

Every (infinite) line on the grid divides it into two "half-grids." A given line and a grid point not on the line, uniquely specifies a paricular half-grid. Coming back to the problem, recall that robot w_1 is adjacent to A. First w_1 changes color to `finalizing`, picks a grid point perpendicular to $\overrightarrow{AB}$, say γ, and moves to γ. This determines the half-grid to which the rest of the robots will move. Robot w_1 terminates, changing its color to `final`. We say that the coordinates of w_1 while terminating at grid point γ are $(1,1)$. Below, we will describe Stage 3 using the coordinate system determined by the positions of A and w_1. Other robots that can see landmarks in the configuration (such as A) will be able to discern this coordinate system. Let HP_γ denote the half-grid bordered by $\overrightarrow{AB}$ containing w_1. The other robots will move into half-grid HP_γ to reach on their target COMPLETE VISIBILITY positions.

When w_i becomes active and recognizes w_1 with color `finalizing` or `final`, it takes color `finalizing` (except $w_{N'-1}$, which takes color `endpoint`) and moves to its neighboring grid point on LP_1^A in HP_γ. Robot w_i can make this decision because it will only see robots toward HP_γ, not in the half-grid on the other side of $\overrightarrow{AB}$. Robot w_i's coordinates now are $(i,1)$. This way all robots that started on $\overrightarrow{AB}$ (except A) will eventually be positioned on LP_1^A between w_1 and $w_{N'-1}$ with color `finalizing`.

From here, Stage 3 sweeps up the rows parallel to $\overrightarrow{AB}$ as follows. Let w_i be a robot with color `finalizing`. Let L_H denote the axis-aligned line through w_i such that on one side of L_H (the *bottom side*) it sees no robots with color `finalizing` and sees at least one robot with color `leader1` (A), `endpoint` ($w_{N'-1}$), or `final`, and on the other side of L_H (the *top side*) it sees no robots with color `leader1` (A), `endpoint` ($w_{N'-1}$), or `final` but can see robots with color `finalizing`. If w_i sees robots with both colors `leader1` (A) and `endpoint` ($w_{N'-1}$) on the bottom side, then it can extract the coordinates of A and $w_{N'-1}$ and its own coordinates (i, y_i) in that system. If $y_i = i^2 \mod N'$, then w_i changes it color to `final` and terminates. Otherwise ($y_i \neq i^2 \mod N'$ or w_i does not see both A and $w_{N'-1}$), w_i moves to the adjacent grid point perpendicular to L_H in the top side. We will show that after all the robots assume color `final` and terminate, COMPLETE VISIBILITY is achieved.

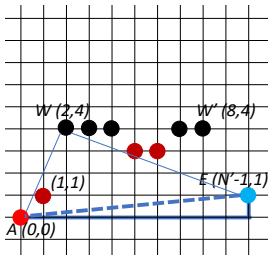

Fig. 7. Placement to reach COMPLETE VISIBILITY points

Fig. 7 illustrates how this is done for a robot w at coordinate $(2,4)$: Seeing both A and E (the robot colored `endpoint`), w can compute its coordinate $(2,4)$, with respect to $(0,0)$. Robot w' at coordinate $(8,4)$ does not terminate at the grid point since (i) if it sees both A and E, then the coordinate computed $(i', y_i') = (8, 8^2 \mod 11) = (8,9)$,

which does not match with $(8, 4)$, (ii) if it does not see both A and E, then its current position is not the COMPLETE VISIBILITY position.

Correctness and runtime of Stage 3. We need the following theorem.

Theorem 4 (Roth [12]). *Suppose M is a prime number. Consider a 2-dimensional $M \times M$ grid G of M^2 grid points with the bottom-left grid point denoted as $(0, 0)$ and the top-right grid point denoted as $(M - 1, M - 1)$. The set of M grid points $(i, i^2 \mod M)$, for $0 \leq i < M$, on G contains no three collinear grid points.*

What essentially Theorem 4 means is that, for any prime number M, if the robots can be put on the M grid points $(i, i^2 \mod M), 0 \leq i < M$, then COMPLETE VISIBILITY is solved. However, there are several challenges in our model: robots do not know N and robots do not have a compass or agreement on the global coordinate system. We show that despite these limitations, robots can move to the grid points satisfying Theorem 4, hence COMPLETE VISIBILITY is correctly solved after Stage 3. We prove the following lemma that helps in the correctness proof.

Lemma 16. *Consider an $M \times M$ grid G of M^2 grid points as in Theorem 4 with the bottom-left grid point denoted as $(0, 0)$ and the top-right grid point denoted as $(M - 1, M - 1)$. Let LP_j^A be the line connecting grid points $(0, j)$ and $(M - 1, j)$, for $0 \leq j \leq M - 1$, with LP_0^A being $\overleftrightarrow{AB}$. Suppose, for $0 \leq k \leq M - 1$, where $0 \leq k^2 \mod M \leq j - 1$, the robots on $LP_0^A, \ldots, LP_{j-1}^A$ are on the grid points $(k, k^2 \mod M)$ of G and the rest of the robots are on LP_j^A. Any robot w on LP_j^A can determine in Stage 3 whether it is on a grid point $(k, k^2 \mod M)$ of G.*

Proof. Since all the robots on $LP_0^A = (\overrightarrow{AB}), \ldots, LP_{i-1}^A$ are on the grid points $(k, k^2 \mod M), 0 \leq k \leq M-1$, and no robot is on any other grid point of $LP_0^A, \ldots, LP_{j-1}^A$, from Theorem 4, they must see each other (i.e., no three are collinear among robots already placed on their final positions). If a robot w on LP_j^A is on grid point $(k, k^2 \mod M)$ such that $j = k^2 \mod M$, again from Theorem 4, it must see all robots on $LP_0^A, \ldots, LP_{j-1}^A$. We have that in Theorem 4, one robot is on LP_0^A at $(0, 0)$ (which is robot A colored `leader1`) and two robots are on LP_1^A, one at $(1, 1)$ and one at $(M - 1, 1)$. The robot at $(M - 1, 1)$ is colored `endpoint`. Let that robot be denoted as E. When w sees both A and E, it can compute M and the coordinates of A and E and its offset k from A. With k and M, w can compute whether $(k, k^2 \mod M)$ is its current grid point. □

Theorem 5 (Bertrand's postulate). *For any integer $N > 3$, there exists at least one prime number N' with $N < N' < 2N - 2$. For any integer $N > 1$, $N < N' < 2N$.*

Lemma 17. *At the end of Stage 3, all robots are positioned on grid points solving* COMPLETE VISIBILITY. *Stage 3 finishes in $\mathcal{O}(N)$ epochs avoiding collisions.*

Proof. Note that in the beginning of Stage 3, all robots are positioned on an axis-aligned line $\overrightarrow{AB}$, starting from A in the direction from A to B. Let the position of A be $(0, 0)$. At least $N - 2$ other robots are on consecutive grid points. Let those points be $(1, 0), \ldots, (N - 2, 0)$. For N' the first prime number greater than or equal to N,

$(N'-1,0)$ also has a robot on it. Consider A's position be the origin of the global coordinate system with $\overleftrightarrow{AB}$ as one axis.

Let HP_γ be one half-grid bordered by $\overleftrightarrow{AB}$. If each robot on $(i,0)$, $0 \leq i < N'$, can move toward HP_γ perpendicularly to $\overrightarrow{AB}$ and position itself on the grid point at distance $i^2 \mod N'$, Theorem 4 is satisfied and COMPLETE VISIBILITY is solved.

The move of robot w_1 to $(1,0)$ provides a direction for robots $w_2, \ldots$ to move, so that they can all move to the same half-grid HP_γ. Since $1 \mod N' = 1$ for any $N' > 1$, terminating w_1 at $(1,1)$ satisfies $(i, i^2 \mod N')$. All robots, except A, then move to LP_1^A (which has w_1 on it) and no robot moves to LP_2^A until all robots of $\overrightarrow{AB}$ reach LP_1^A. A remains on $\overrightarrow{AB}$, providing $(0,0)$. Robot $w_{N'-1}$ terminating on LP_1^A satisfies $(N'-1, (N'-1)^2 \mod N') = (N'-1, 1)$, since for any number $M' > 1$, $(M'-1)^2 \mod M' = 1$. After all robots move to LP_2^A, Lemma 16 is satisfied for each robot. If a robot w is at a coordinate $(i, i^2 \mod N')$, it terminates assuming color `final`. Otherwise, it continues moving to LP_3^A and beyond until it can terminate satisfying Lemma 16. This way all robots settle at the grid points of Theorem 4, achieving COMPLETE VISIBILITY. Collision avoidance is immediate since each robot moves on a line with no other robot.

Regarding runtime, in one epoch, w_1 moves to LP_1^A. In the second epoch, w_1 terminates. In the same epoch, all the robots on $\overrightarrow{AB}$, except A, move to LP_1^A. In the third epoch, $w_1, w_{N'-1}$ terminate on LP_1^A and the rest move to LP_2^A. Since there are $N \leq N'$ robots in $\mathcal{Q}$ and $(i^2 \mod N') \leq N'-1$, the robots reach at most $LP_{N'-1}^A$. Therefore, robots reach $LP_{N'-1}^A$ at epoch N'. The robots that reach $LP_{N'-1}^A$ terminate at epoch $N'+1$. Therefore, Stage 3 finishes in $N'+1$ epochs. Using Theorem 5, $N \leq N' < 2N-2$. Therefore, Stage 3 finishes in $\mathcal{O}(N)$ epochs. □

Overall correctness of the algorithm. We showed so far that if each stage (and sub-stage) achieves the configurations as required to start the next stage (and sub-stage), COMPLETE VISIBILITY is guaranteed in our algorithm. We prove here that it is indeed the case: The robots can correctly distinguish each stage (and sub-stage) during the execution and they work as intended. Specifically, we prove two lemmas below.

Lemma 18. *Stages 1–3 execute sequentially one after another in the algorithm.*

Proof. Consider first any $C_{non,init}$. Stage 1 forms a FCC with four robots A, B, C, D on it. A is colored `leader1`, B `leader2`, C `defeated1`, and D `defeated2`. B (C) is adjacent to A (D). In Stage 2, the robots in the interior of the FCC that are the first to move (or change color without moving) must see both C, D colored `defeated1` and `defeated2`, respectively. This happens only after Stage 1.2 (and hence Stage 1) is finished with D colored `defeated2`. For $C_{col,init}$, when A, B, C, D pick color, all others are colored `collinear-moving` (different than they would have been colored at end of Stage 1 for $C_{non,init}$) and hence they can continue until the desired line configuration is achieved. For any C_{init} (i.e., collinear and non-collinear), Stage 3 is started only after all robots are in a collinear configuration with specific colors. In Stage 3, w_1, adjacent to A, moves first. For w_1 to move, it should not see C or D. D is the last to move to $\overrightarrow{AB}$ in Stage 2 and as soon as D moves to $\overline{AB}$, Stage 2 is finished. Therefore, Stage 3 executes only after Stage 2 finishes. □

Lemma 19. *Stages 1.1, 1.2, 2.1–2.7 correctly finish one after another in the algorithm.*

Proof. Stage 1.1 finishes after four robots, $r_1, \ldots, r_4$, are on a FCC and colored `rectangle-point`. Stage 1.2 cannot start until at least three rectangle points of the FCC have color `rectangle-point`. If Stage 1.2 starts after all $r_1, \ldots, r_4$ colored `rectangle-point`, we are done. Otherwise, Stage 1.2 must be started by r_i that sees both of its neighboring rectangle points colored `rectangle-point`. In this case, r_{i-1} and r_{i+1} detect that one of their neighbors is still not colored `rectangle-point` and do not take part in Stage 1.2 procedure started by r_i. Lemma 18 establishes the sequentiality between Stages 1.2 and 2.1. Stage 2.2 starts only after all robots in the interior of the FCC are colored `multi-robot`, since the robots closest to $\overrightarrow{AB}$ start this process. For Stage 2.3, the robots on LP^A_β needs to be colored `multi-robot-2`. If robots on any line L^A_i start Stage 2.3 before Stage 2.2 finishes, the robots on L^A_i later wait for others to finish Stage 2.3, since they are independent of each other. Similarly, Stage 2.4 is synchronized from L^A_i close to B towards A and hence it is synchronized. For Stage 2.5, the only robots in the interior of the FCC should be on LP^A_β with color `multi-robot-3`. For Stage 2.6, there should be no robot in the interior of the SER, i.e., all robots on $\overrightarrow{AB}$ except C and D. For Stage 3, C and D should also be on $\overrightarrow{AB}$. This happens only after D moves to $\overrightarrow{AB}$ in Stage 2.7 (Lemma 18). □

Proof of Theorem 1. For any $\mathbf{C}_{init}$ with diameter $D = \mathcal{O}(N)$, we have Theorem 1 with runtime $\mathcal{O}(N)$ epochs combining the results of Lemmas 4–15 and 17 –19. For any initial configuration C_{init} with diameter $D = \Omega(N)$, running the techniques for Stages 1–3 only increases the runtime from $\mathcal{O}(N)$ epochs to $\mathcal{O}(D)$ epochs. In other words, $D = \Omega(N)$ does not have impact on the correctness of the algorithm. The total number of colors used throughout the algorithm depend on whether leader election is used or not. With leader election, the number of colors is 50, combining 38 colors for Stage 1, 9 colors for Stage 2, and 3 colors Stage 3; note that 2 colors for Stage 1 and 2 of collinear configurations do not need to add here since for collinear configurations, Stage 1 for non-collinear configurations is not executed. Without leader election, we need 4 colors to differentiate A, B, C, D (4 leaders) and 1 color for initial configuration. After that Stage 2 needs 9 colors and Stage 3 needs 3 colors, which totals 17. Overall correctness is then given by Lemmas 18 and 19. Theorem 1 follows. ■

4 Concluding Remarks

We have presented an $\mathcal{O}(\max\{D, N\})$-time algorithm (deterministic without leader election and randomized with leader election) for COMPLETE VISIBILITY for $N \geq 1$ robots with lights in the $\mathcal{ASYNC}$ setting on an infinite grid, a natural discretization of the 2-dimensional Euclidean plane, where D is the diameter of the initial configuration. The number of colors in our algorithm is as follows: 17 when leader election is not needed and 50 when leader election is needed. We also proved a time lower bound of $\Omega(N)$ irrespective of whether unlimited number of colors are available and leader, which shows that our algorithm is optimal for $D = \mathcal{O}(N)$. The best previously known 11-color deterministic algorithm for this problem on an infinite grid only provides correctness and finite time termination guarantees. Our analysis shows that the previous

algorithm has $\mathcal{O}(\max\{DN, N^2\})$ runtime. This problem is fundamental and has been receiving much attention recently.

Several questions remain open from this work. We focused only on minimizing time but not the number of colors used. A better analysis of our algorithm on the number of colors would provide smaller number of colors than current 50 when leader election is needed and 17 when leader election is not needed. Therefore, one interesting open question is can the number of colors be minimized to 2 (optimal) or close to 2 for COMPLETE VISIBILITY on an infinite grid. Another direction is to see whether robot faults (crash and/or byzantine) can be handled. Both the algorithm we presented here and the previous algorithm assume that robots do not suffer from faults.

References

1. Adhikary, R., Bose, K., Kundu, M.K., Sau, B.: Mutual visibility by asynchronous robots on infinite grid. In: ALGOSENSORS. pp. 83–101 (2018)
2. Cord-Landwehr, A., Degener, B., Fischer, M., Hüllmann, M., Kempkes, B., Klaas, A., Kling, P., Kurras, S., Märtens, M., Meyer auf der Heide, F., Raupach, C., Swierkot, K., Warner, D., Weddemann, C., Wonisch, D.: A new approach for analyzing convergence algorithms for mobile robots. In: ICALP. pp. 650–661 (2011)
3. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theor. Comput. Sci. 609, 171–184 (2016)
4. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Inf. Comput. 254, 392–418 (2017)
5. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Santoro, N., Viglietta, G.: Robots with lights: Overcoming obstructed visibility without colliding. In: SSS. pp. 150–164 (2014)
6. Flammenkamp, A.: Progress in the no-three-in-line-problem. J. Comb. Theory, Ser. A 60(2), 305–311 (1992)
7. Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by oblivious mobile robots. Synthesis Lectures on Distributed Computing Theory 3(2), 1–185 (2012)
8. Guy, R.K., Kelly, P.A.: The no-three-in-line problem. Canadian Mathematical Bulletin 11(4), 527531 (1968)
9. Hall, R.R., Jackson, T.H., Sudbery, A., Wild, K.: Some advances in the no-three-in-line problem. J. Comb. Theory, Ser. A 18(3), 336–341 (1975)
10. 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)
11. Peleg, D.: Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In: IWDC. pp. 1–12 (2005)
12. Roth, K.F.: On a problem of heilbronn. Journal of the London Mathematical Society s1-26(3), 198–204 (1951)
13. Sharma, G., Busch, C., Mukhopadhyay, S.: Mutual visibility with an optimal number of colors. In: ALGOSENSORS. pp. 196–210 (2015)
14. Sharma, G., Dutta, A., Kim, J.H.: Optimal online coverage path planning with energy constraints. In: AAMAS. pp. 1189–1197 (2019)
15. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for asynchronous robots with lights. In: SSS. pp. 265–281 (2017)
16. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Complete visibility for robots with lights in O(1) time. In: SSS. pp. 327–345 (2016)
17. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: O(log n)-time complete visibility for asynchronous robots with lights. In: IPDPS. pp. 513–522 (2017)
18. Vaidyanathan, R., Busch, C., Trahan, J.L., Sharma, G., Rai, S.: Logarithmic-time complete visibility for robots with lights. In: IPDPS. pp. 375–384 (2015)