Mutual Visibility by Luminous Robots
Without Collisions

G.A. Di Luna[a], P. Flocchini[b], S. Gan Chaudhuri[c], F. Poloni[d], N. Santoro[e],
G. Viglietta[b]

[a]*Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Università degli
Studi di Roma "La Sapienza", Italy*
[b]*School of Electrical Engineering and Computer Science, University of Ottawa, Canada*
[c]*Department of Information Technology, Jadavpur University, Kolkata, India*
[d]*Dipartimento di Informatica, Università di Pisa, Italy*
[e]*School of Computer Science, Carleton University, Ottawa, Canada*

Abstract

Consider a finite set of identical computational entities that can move freely in the Euclidean plane operating in Look-Compute-Move cycles. Let $p(t)$ denote the location of entity p at time t; entity p can see entity q at time t if at that time no other entity lies on the line segment $p(t)q(t)$. We consider the basic problem called Mutual Visibility: starting from arbitrary distinct locations, within finite time the entities must reach, without collisions, a configuration where they all see each other. This problem must be solved by each entity autonomously executing the same algorithm. We study this problem in the *luminous robots* model; in this generalization of the standard model of oblivious robots, each entity, called *robot*, has an externally visible persistent light that can assume colors from a fixed set of size c. The case where the number of colors is less than 2 (i.e., $c \leqslant 1$) corresponds to the classical model without lights: indeed, having lights of one possible color is equivalent to having no lights at all.

The extensive literature on computability in such a model, mostly for $c \leqslant 1$ and recently for $c > 1$, has never considered the problem of Mutual Visibility because it has always assumed that three collinear robots are mutually visible.

In this paper we remove this assumption, and investigate under what conditions luminous robots can solve Mutual Visibility without collisions, and at what cost, in terms of the number of colors used by the robots. We establish a spectrum of results, depending on the power of the adversary (i.e., the scheduler controlling the robots' actions), on the number c of colors, and on the a-priori knowledge the robots have about the system. Among such

results, we prove that Mutual Visibility can always be solved without collisions in SSynch with $c = 2$ colors and in ASynch with $c = 3$ colors. If an adversary can interrupt and stop a robot before it reaches its computed destination, Mutual Visibility is still solvable without collisions in SSynch with $c = 3$ colors, and, if the robots agree on the direction of one axis, also in ASynch. All the results are obtained constructively by means of novel protocols.

As a byproduct of our solutions, we provide the first obstructed-visibility solutions to two classical problems for oblivious robots: *collision-less convergence to a point* (also called *near-gathering*) and *circle formation*.

1. Introduction

1.1. Computational Framework

Consider a distributed system composed of a team of mobile computational entities, called *robots*, moving and operating in the Euclidean plane $\mathbb{R}^2$, initially each at a distinct point. Each robot can move freely in the plane, and operates in *Look-Compute-Move* cycles. During a cycle, a robot determines the position (in its own coordinate system) of the other robots (*Look*); it executes a protocol (which is deterministic and it is the same for all robots) to determine a destination point (*Compute*); and moves towards the computed destination (*Move*). After each cycle, a robot may be inactive for an arbitrary but finite amount of time. The robots are anonymous, without a central control, and oblivious (i.e., at the beginning of a cycle, a robot has no memory of any observation or computation performed in its previous cycles). What is computable by such entities has been the object of extensive research within distributed computing; e.g., see [2, 7, 8, 14, 21, 25, 27, 29, 34, 35, 37]; for a recent review see [23].

Vision and mobility provide the robots with *stigmergy*, enabling the robots to communicate and coordinate their actions by moving and sensing their relative positions; they are otherwise assumed to lack any means of explicit direct communication. This restriction could enable deployment in extremely harsh environments where communication is impossible or can be jammed. Nevertheless, in many other situations it is possible to assume the availability of some sort of direct communication. The theoretical interest is obviously for weak communication capabilities.

A model employing a weak explicit communication mechanism is that of *robots with lights*, or *luminous robots*, initially suggested by Peleg [33]. In this model, each robot is provided with a local externally-visible *light*, which

can assume colors from a fixed set. The robots explicitly communicate with each other using these lights. The lights are persistent (i.e., the color is not erased at the end of a cycle), but otherwise the robots are oblivious [12, 13, 21, 26, 33, 36]. Notice that a light with only one possible color is the same as no light; hence the luminous robots model generalizes the classical one.

Both in the classical model and in that with lights, depending on the assumptions on the activation schedule and the duration of the cycles, different settings are identified. In the synchronous setting, the robots operate in rounds, and all the robots that are activated in a round perform their cycle in perfect synchrony. In this case, the system is *fully synchronous* (or FSYNCH) if all robots are activated at all rounds, and it is *semi-synchronous* (or SSYNCH) otherwise. In the *asynchronous* setting (or ASYNCH), there is no common notion of time, and no assumption is made on the timing and duration of each computation and movement, other than that it is finite. (For the SSYNCH and ASYNCH models there are bland fairness assumptions that prevent robots from remaining inactive forever, which are discussed in Section 2.1).

The choice of when a robot is activated (in SSYNCH) and the duration of an activity within a cycle (in ASYNCH) is made under the control of an *adversary*, or *scheduler*. Similarly, the choices of the initial location of each robot and of its private coordinate system are made under adversarial conditions.

A crucial distinction is whether or not the adversary has also the power to stop a moving robot before it reaches its destination. If so, the moves are said to be *non-rigid*. The only constraint is that, if interrupted before reaching its destination, the robot moves at least a minimum distance $\delta > 0$ (otherwise, the adversary would be able to prevent robots from reaching any destination, in any amount of cycles). If the adversary does not have such a power, the moves are said to be *rigid*. The model with rigid moves is referred to as RIGID, and the other one is called NON-RIGID.

In the rest of the paper, with abuse of terminology, we will often refer to FSYNCH, SSYNCH, or ASYNCH robots or schedulers (as opposed to systems), and to RIGID or NON-RIGID robots or schedulers (as opposed to models).

1.2. Obstructed Visibility

The classical model and the more recent model of robots with lights share a common assumption: that three or more collinear robots are mutually visible. It can be easily argued against such an assumption, and for the

importance of investigating computability when visibility is *obstructed* by the presence of other robots: that is, if two robots r and s are located at $r(t)$ and $s(t)$ at time t, they can see each other if and only if no other robot lies on the segment $r(t)s(t)$ at that time.

Very little is known on computing with obstructed visibility. In fact, the few studies on obstructed visibility have been carried out in other models: the model of robots in the *one-dimensional* space $\mathbb{R}$ [9]; and the so-called *fat robots* model, where robots are not geometric points but occupy unit disks, and collisions are allowed and can be used as an explicit computational tool (e.g., [1, 5, 11]). In our model, collisions can create unbreakable symmetries: since robots are oblivious and anonymous and execute the same protocol, if $r(t) = s(t)$ (a collision), then the activation adversary can force $r(t') = s(t')$ for all $t' > t$ if the two robots do not have lights or their lights have the same color. Thus, unless this is the intended outcome, collision avoidance is always a requirement for all algorithms in the model considered here.

In this paper we focus on luminous robots in the presence of obstructed visibility, and investigate computing in such a setting. Clearly, obstructed visibility increases the difficulty of solving problems without the use of additional assumptions. For example, with unobstructed visibility, every active robot can determine the total number n of robots at each activity cycle. With obstructed visibility, unless a robot has a-priori knowledge of n and this knowledge is persistently stored, the robot might be unable to decide if it sees all the robots; hence it might be unable to determine the value n.

The main problem we investigate, called **Mutual Visibility**, is perhaps the most basic in a situation of obstructed visibility: starting from arbitrary distinct positions in the plane, within finite time the robots must reach a configuration in which they are in distinct locations, they can all see each other, and they no longer move. This problem is clearly at the basis of any subsequent task requiring complete visibility. Notice that this problem does not exist under unobstructed visibility, and has never been investigated before.

Among the configurations that achieve mutual visibility, a special class is that where all robots are in a strictly convex position; within that class, of particular interest are those where the robots are on the perimeter of a circle, possibly equally spaced. The problems of forming such configurations (respectively called **Convex Formation** and **Circle Formation**) have been extensively studied both directly (e.g., [15, 17, 18, 24]) and as part of the more general **Pattern Formation** problem (e.g., [25, 27, 34, 35, 37]). Unfortu-

nately, none of these investigations consider obstructed visibility, and those algorithms do not work in the setting considered here.

Note that a requirement of the Mutual Visibility problem is that robots stop moving after they have reached a configuration in which they all see each other. To this end, we will grant robots the ability to perform a special operation called *termination*, after which they can no longer be activated by the scheduler. The termination operation is especially useful in practice when the robots have to perform several tasks in succession. Of course, even if this operation is not directly available, it can still be simulated via the addition of an extra color, which can be used by a robot to indicate (to the other robots, as well as to itself) that it has terminated. Moreover, if the termination operation is removed from the algorithms presented in this paper (and some straightforward adjustments are made, which do not require extra colors), then a weaker form of the Mutual Visibility problem is solved, in which the robots get to permanently see each other, but they never stop moving. That is, all obstructions are permanently removed, but the termination condition is not met. In some cases, removing the termination operation will even allow us to successfully apply our algorithms to different problems, such as Circle Formation and Near Gathering, as discussed in Section 6.

1.3. Main Contributions

In this paper we investigate under what conditions luminous robots can solve Mutual Visibility and at what cost (i.e., with how many colors). We establish a spectrum of results, depending on the power of the adversary, on the number c of colors, and on the a-priori knowledge the robots have about the systems.

We first consider the case when the adversary can choose the activation schedule (in SSYNCH) and the duration of each robot's operations (in ASYNCH), but cannot interrupt the movements of the robots; that is, movements are rigid. In this case, we show the following.

Theorem 1.1. Mutual Visibility *is solvable without collisions by* RIGID *robots*

  (a) *with* no *colors in* SSYNCH, *if the robots know their number, n;*

  (b) *with* 2 *colors in* SSYNCH, *always;*

  (c) *with* 3 *colors in* ASYNCH, *always.*

We then consider the case when the adversary has also the power to interrupt the movements of the robots; that is, movements are non-rigid. The only restriction is that there exists a constant absolute length $\delta > 0$ such that, even if a robot's move is interrupted before it reaches the destination, it travels at least a length δ towards it (otherwise it many never be able to reach any destination). In the case of non-rigid movements, we prove the following.

Theorem 1.2. Mutual Visibility *is solvable without collisions by* Non-Rigid *robots*

(a) *with* no *colors in* SSynch*, if the robots know δ and their number, n;*

(b) *with* 2 *colors in* SSynch*, if the robots know δ;*

(c) *with* 3 *colors in* SSynch*, always;*

(d) *with* 3 *colors in* ASynch*, if the robots agree on the direction of one coordinate axis.*

All these results are established constructively. We present and analyze two protocols, Algorithm 1 (*Shrink*) and Algorithm 2 (*Contain*), whose goal is to allow the robots to position themselves at the vertices of a convex polygon, solving Convex Formation, and thus Mutual Visibility. These two algorithms are based on different strategies, and are tailored for different situations. Protocol *Shrink* uses two colors and requires rigid movements, while protocol *Contain* uses more colors but operates also with non-rigid movements. We prove their correctness for SSynch robots (Sections 3 and 4). We then show how, directly or with simple expansions and modifications of these two algorithms, all the claimed results follow (Sections 5 and 6). Finally, we propose some open problems (Section 7).

Let us point out that, to prove the correctness of *Shrink*, we solve a seemingly unrelated problem, Communicating Vessels, which is interesting in its own right.

As a byproduct of our solutions, we provide the first obstructed-visibility solution to a classical problem for oblivious robots: *collision-less convergence to a point* (Near-Gathering) (see [23, 32]), Indeed, if the robots continue to follow algorithm *Shrink* once they reach full visibility, the convex hull of their positions converges to a point, and the robots approach it without colliding, thus solving Near-Gathering (Section 6.3). This algorithm has an

6

interesting fault-tolerance property: if a single robot is faulty and becomes unable to move, the robots will still solve Near-Gathering, converging to the faulty robot's location (Section 6.6).

Additionally, both protocols can be modified so that the robots can position themselves on the perimeter of a circle, thus providing an obstructed-visibility solution to the classical problem of Circle Formation. The problem can be solved with 2 colors in RIGID SSYNCH, with 3 colors in NON-RIGID SSYNCH, and with 4 colors in RIGID ASYNCH, and NON-RIGID ASYNCH with agreement on one axis (Section 6.2).

2. Model and Definitions

2.1. Modeling Robots

We mostly follow the terminology and definitions of the standard model of oblivious mobile robots (e.g., see [23]).

By $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ we denote a set of oblivious mobile computational entities, called *robots*, operating in the Euclidean plane, and initially placed at distinct points. Each robot is provided with its own local coordinate system centered in itself, and its own notion of unit distance and handedness. We denote by $r(t) \in \mathbb{R}^2$ the position occupied by robot $r \in \mathcal{R}$ at time t; these positions are expressed here in a global coordinate system, which is used for description purposes, but is unknown to the robots. Two robots r and s are said to *collide* at time t if $r(t) = s(t)$. A robot r can *see* another robot s (equivalently, s is *visible* to r) at time t if and only if no other robot lies in the segment $r(t)s(t)$ at that time.

The robots are luminous: each robot r has a persistent state variable, called *light*, which may assume any value in a finite set $\mathcal{C}$ of *colors*. The color of r at time t can be seen by all robots that can see r at that time.

The robots are autonomous (i.e., without any external control), anonymous (i.e., without internal identifiers), indistinguishable (i.e., without external markings), without any direct means of communication, other than their lights. At any time, robots can be performing a variety of operations, but initially (i.e., at time $t = 0$) they are all still and idle.

When activated, a robot performs a *Look-Compute-Move* sequence of operations: it first obtains a snapshot of the positions, expressed in its local coordinate system, of all visible robots, along with their respective colors (*Look* phase); using the last obtained snapshot as an input, the robot executes a deterministic algorithm, which is the same for all robots, to compute a

7

destination point $x \in \mathbb{R}^2$ (expressed in its local coordinate system) and a color $c \in \mathcal{C}$, and it sets its light to c (*Compute* phase); finally, it moves towards x (*Move* phase). It then starts a new cycle, whenever the scheduler (which is an abstract entity controlling to some extent the behavior of the robots) decides to activate it again.

In the *Compute* phase, a robot may also decide to terminate its execution. When a robot has terminated, it remains still forever, and its light remains the same color that it was at the moment of termination.

The robots are *oblivious* in the sense that, when a robot transitions from one cycle to the next, all its local memory, except for the light, is reset. In other words, a robot has no memory of past computations and snapshots, except for the light.

With regards to the activation and timing of the robots, there are two basic settings: *semi-synchronous* (SSYNCH) and *asynchronous* (ASYNCH). In SSYNCH, the time is discrete; at each time instant $t \in \mathbb{N}$ (called a *round* or a *turn*) a subset of the robots is activated by the scheduler and performs a whole *Look-Compute-Move* cycle atomically and instantly. At any given round, any subset of robots may be activated, from the empty set to all of $\mathcal{R}$. In particular, if all robots are activated at every round, the setting is called *fully synchronous* (FSYNCH). There is a bland *fairness* constraint on the choices that the scheduler can make: every robot must be activated infinitely many times (unless it terminates). In ASYNCH, there is no common notion of time: each robot executes its cycles independently, the *Look* operation is instantaneous, but the *Compute* and *Move* operation can take an unpredictable (but finite) amount of time, unknown to the robot. In a *Move* phase there are no constraints on the speed of a robot, as long as it always moves directly towards its destination point at non-negative speed.

The scheduler that controls the activations (in SSYNCH) and the durations of the operations (in ASYNCH) can be thought of as an *adversary*, whose purpose is to prevent the robots from doing their task. Other than acting as a scheduler, the adversary also determines the initial position of the robots and their local coordinate systems; in particular, the coordinate system of a robot might not be preserved over time and might be modified by the adversary between one cycle and the next. In the simplest model, the robots do not necessarily agree on the orientation of the coordinate axes, on the unit distance, and on the clockwise direction (i.e., the handedness of the system). However, in Section 5.2, we will discuss the special model in which all the robots agree on the direction of one axis, and the adversary is unable

8

to change it.

The adversary might or might not have the power to interrupt the movement of a robot before it reaches its destination in the *Move* operation. If it does, the system is said to be NON-RIGID. The only constraint on the adversary is that there exists a constant $\delta > 0$ such that, if interrupted before reaching its destination, a robot moves at least a distance δ. The value of δ is decided by the scheduler once and for all, and typically it is not known by the robots, which therefore cannot use it in their computations (we will discuss the scenario in which the robots know the value of δ in Section 6.4). Notice that, without this constraint, the adversary would be able to prevent a robot from reaching any given destination in a finite number of turns. If movements are not under the control of the adversary, and every robot reaches its destination at every turn, the system is said to be RIGID.

2.2. Mutual Visibility and Related Problems

The Mutual Visibility problem requires n robots to form a configuration in which they occupy n distinct locations, and no three of them are collinear. Subsequently, the robots have to terminate. A protocol P is a solution of Mutual Visibility if it allows the robots to solve Mutual Visibility starting from any initial configuration in which their positions are all distinct, and regardless of the decisions of the adversary (including the activation schedule, the local coordinate systems of the robots, and the value of δ).

Let us stress that, since robots are oblivious and anonymous and execute the same protocol, if $r(t) = s(t)$ (a collision), then the adversary can force $r(t') = s(t')$ for all $t' > t$ if the two robots do not have lights or their lights have the same color. Hence the two robots will never again occupy distinct locations, and will no longer be able to solve Mutual Visibility. Thus, collision avoidance of robots with the same color is a requirement for any solution protocol.

Among the configurations that solve the Mutual Visibility problem, a special class is that in which all robots are in a strictly convex position. Within this class, of particular interest are the configurations in which the robots lie on the perimeter of a circle. Among these, there are the notable configurations in which the robots occupy the vertices of a regular n-gon. The problems of forming such configurations are called Convex Formation, Circle Formation, and Uniform Circle Formation, respectively.

2.3. Geometric Notions and Observations

A finite set of points $S \subset \mathbb{R}^2$ is said to be *convex* if all the points of S lie on the perimeter of the convex hull of S. If a point p of a convex set S lies in the relative interior of an edge of the convex hull of S, then p is said to be a *degenerate* vertex of the convex hull. If none of the points of a convex set S is a degenerate vertex of the convex hull, then S is said to be a *strictly convex* set. On the other hand, we will say that a polygon is *degenerate* if its area is zero. These two notions of degeneracy are used in different contexts (one refers to vertices, the other refers to whole polygons), hence they can hardly be confused.

Let $\mathcal{H}(t)$ denote the convex hull of $\{r_1(t), r_2(t), \cdots, r_n(t)\}$ at time t. The robots lying on its boundary are called *external robots* at time t, while the ones lying in its interior are the *internal robots* at time t.

Observe that a robot may not know where the convex hull's vertices are located, because its view may be obstructed by other robots. However, it can easily determine whether it is an external or an internal robot. In fact, a robot r is external at time t if and only if there is a half-plane bounded by a straight line through $r(t)$ whose interior contains no robots at time t. In other words, r is external if and only if it lies on the boundary of the convex hull of the robots that it can currently see. Note also that the neighbors of an external robot on its visible convex hull are indeed its neighbors on the actual convex hull. If, in addition, r lies at a non-degenerate vertex of the (visible) convex hull, it is said to be a *vertex* robot.

Moreover, a robot is able to tell if $\mathcal{H}$ is a line segment, i.e., if all the robots are collinear. In particular, if a robot can see only one other robot, it understands that it is an *endpoint robot*. Conversely, non-endpoint robots can always see more than one other robot.

The points of $\mathbb{R}^2$ are treated like vectors, and as such they can be added, subtracted, multiplied by scalars, etc. The dot product between vectors a and b will be indicated by the expression $a \bullet b$.

3. Solving Mutual Visibility for RIGID SSYNCH Robots

In this section we consider the Mutual Visibility problem in the RIGID SSYNCH setting. We present and analyze a protocol, Algorithm 1 (*Shrink*), and we prove it solves Mutual Visibility in such a setting using only two colors.

3.1. Description of Algorithm 1

The main idea of Algorithm 1 is to make only the external robots move, so as to shrink the convex hull. When a former internal robot becomes external, it starts moving as well. Eventually, all the robots reach a strictly convex configuration, and at this point they all see each other and they can terminate.

If an active robot r_i, located at p, realizes that it is not a vertex robot, it does not move. Otherwise, it locates its clockwise and counterclockwise neighbors (in its own coordinate system) on the convex hull's boundary, say located at a and b, which are necessarily visible. Then, r_i attempts to move somewhere in the triangle $\triangle pab$, in such a way to shrink the convex hull, and possibly make one more robot become a vertex robot. To avoid collisions with other robots that may be moving at the same time, r_i's movements are restricted to a smaller triangle, shaded in gray in Figure 1. Moreover, to avoid becoming a non-vertex robot, r_i does not cross any line parallel to ab that passes through another robot, and it carefully positions itself on the closest of such lines, as shown in Figure 1(a). In particular, if no such line intersects the gray area, r_i makes a *default move*, and it moves halfway toward the midpoint of the segment ab, as indicated in Figure 1(b).

In order to recognize that the Mutual Visibility problem has been solved, and to correctly terminate, the robots carry visible lights of two possible colors: namely, $\mathcal{C} = \{$*Off*, *Vertex*$\}$. All robots' lights are initially set to *Off*. If an active robot realizes that it is a vertex of the convex hull, it sets its light to the other value, *Vertex*. Hence, when a robot sees only robots whose lights are set to *Vertex*, it knows it can see all the robots in the swarm, and hence it terminates.

The above rules are sufficient to solve the Mutual Visibility problem in most cases, but there are some exceptions. It is easy to see that there are configurations in which Mutual Visibility is never solved until an internal robot moves, regardless of the algorithm employed. For instance, suppose that the configuration is centrally symmetric, with one robot lying at the center. Let the local coordinate systems of any two symmetric robots be oriented symmetrically and have the same unit distance, and assume that the scheduler chooses to activate all robots at every turn. Then, every two symmetric robots have symmetric views, and therefore they move symmetrically. If the central robot—which is an internal robot—never moves, then the configuration remains centrally symmetric, and the central robot always obstructs all

Algorithm 1: Shrink. Solving the Mutual Visibility problem for Rigid SSynch robots with 2-colored lights

Input: $\mathcal{V}$: set of robots visible to me (myself included) whose positions are expressed in a coordinate system centered at my location.

1 $r^* \longleftarrow$ myself
2 $\mathcal{P} \longleftarrow \{r.position \mid r \in \mathcal{V}\}$
3 $\mathcal{H} \longleftarrow$ convex hull of $\mathcal{P}$
4 **if** $|\mathcal{V}| = 3$ **and** $\mathcal{H}$ *is a line segment* **then**
5 $\quad$ Move orthogonally to $\mathcal{H}$ by any positive amount

6 **else**
7 $\quad$ **if** $r^*.position$ *is a vertex of* $\mathcal{H}$ **then**
8 $\quad\quad$ $r^*.light \longleftarrow Vertex$
9 $\quad\quad$ **if** $\forall r \in \mathcal{V}$, $r.light = Vertex$ **then** Terminate
10 $\quad\quad$ **else if** $|\mathcal{V}| > 2$ **then**
11 $\quad\quad\quad$ $a \longleftarrow$ position of my ccw neighbor on the boundary of $\mathcal{H}$
12 $\quad\quad\quad$ $b \longleftarrow$ position of my cw neighbor on the boundary of $\mathcal{H}$
13 $\quad\quad\quad$ $u \longleftarrow a/2$
14 $\quad\quad\quad$ $\gamma \longleftarrow 1/2$
15 $\quad\quad\quad$ **foreach** $r \in \mathcal{V} \setminus \{r^*\}$ **do**
16 $\quad\quad\quad\quad$ Let α, β be such that $r.position = \alpha \cdot a + \beta \cdot b$
17 $\quad\quad\quad\quad$ **if** $\alpha + \beta < \gamma$ **then**
18 $\quad\quad\quad\quad\quad$ $u \longleftarrow r.position$
19 $\quad\quad\quad\quad\quad$ $\gamma \longleftarrow \alpha + \beta$
20 $\quad\quad\quad\quad$ **else if** $\alpha + \beta = \gamma$ **and** $r.position$ *is closer to* b *than* u **then** $u \longleftarrow r.position$
21 $\quad\quad\quad$ $v \longleftarrow \gamma \cdot b$
22 $\quad\quad\quad$ Move to $(u + v)/2$

23 $\quad$ **else if** $\forall r \in \mathcal{V} \setminus \{r^*\}$, $r.light = Vertex$ **and** $r^*.position$ *lies in the interior of* $\mathcal{H}$ **then** Move to the midpoint of any edge of $\mathcal{H}$

12
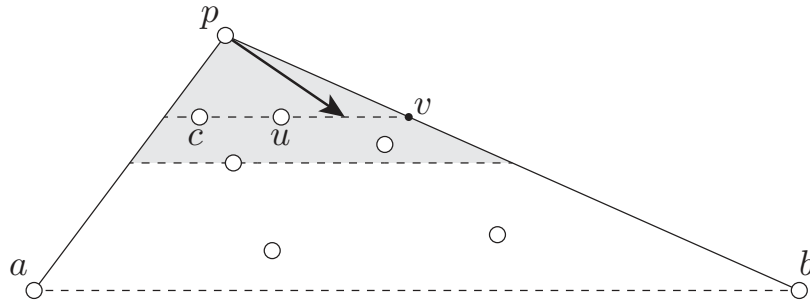
(a) Making c become a vertex robot, without moving past it
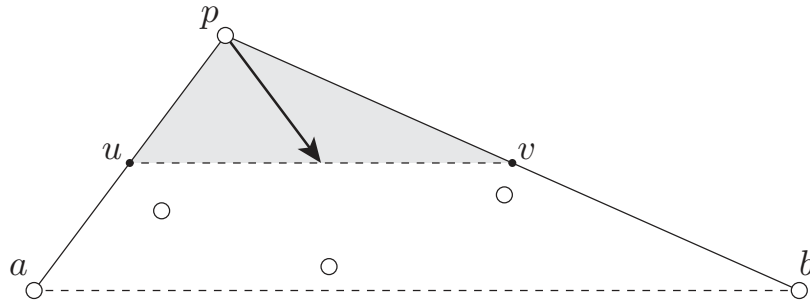
(b) Default move

Figure 1: Move of an external robot, in two different cases (robots' locations are indicated by small circles)

pairs of symmetric robots. Hence Mutual Visibility is never solved, no matter what algorithm is executed.

It turns out that our rules can be fixed in a simple way to resolve also this special case: whenever an internal robot sees only robots whose lights are set to *Vertex* (except its own light), it moves to the midpoint of any edge of the convex hull.

Finally, the configurations in which all the robots are collinear need special handling. In this case it is impossible to solve Mutual Visibility unless some robots leave the current convex hull. Suppose that a robot r realizes that all robots lie on a line, and that it is not an endpoint (i.e., r can see only two other robots, which are collinear with it). Then, r moves by any positive amount, orthogonally to the line formed by the other two visible robots. When this is done, the previous rules apply.

13

3.2. Correctness of Algorithm 1

3.2.1. Invariants

In the following we discuss some basic invariants, which will serve to prove the correctness of Algorithm 1.

Suppose that, for some $t \in \mathbb{N}$, $\mathcal{H}(t)$ is not a line segment: the situation is illustrated in Figure 2. If a vertex robot is activated, it is bound to remain in the corresponding gray triangle, called *movement region* of the robot. More precisely, the movement region consists of the interior of the gray triangle, plus the vertex where the robot currently is, plus the interior of the edge that is opposite to the robot. Hence all movement regions are disjoint. Moreover, if there is only one internal robot and it sees only robots whose light is set to *Vertex*, it moves to the midpoint of an edge of $\mathcal{H}(t)$, which does not lie in any movement region. It follows that, no matter which robots are activated at time t, they will not collide at time $t + 1$. Also, $\mathcal{H}(t + 1) \subseteq \mathcal{H}(t)$.
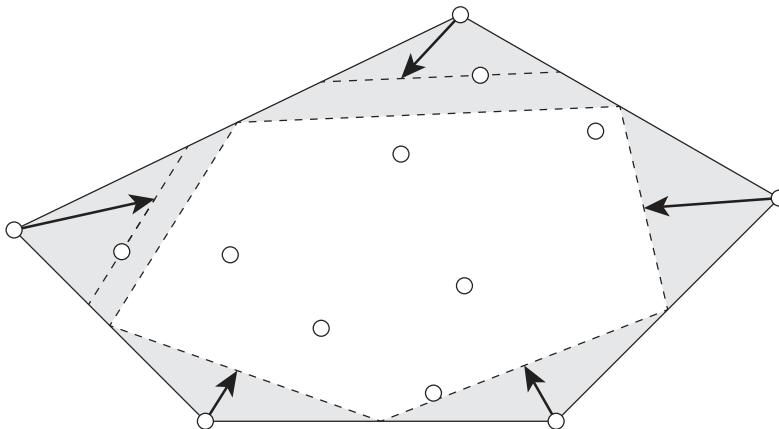

Figure 2: Combined motion of all vertex robots

Recall that a robot $r \in \mathcal{R}$ is a vertex robot if an only if it lies at the vertex of a reflex angle whose interior does not contain any robots. Now, referring to Figure 1, it is clear that a vertex robot will remain a vertex robot after a move. Additionally, if no new vertex robots are acquired between time t and $t + 1$, then the ordering of the vertex robots around the convex hull is preserved from time t to time $t + 1$. This easily follows from the fact that every robot remains in its own movement region (cf. Figure 2).

3.2.2. Convergence

We seek to prove that Algorithm 1 makes every robot eventually become a vertex robot. As it will be apparent in the proof of Theorem 3.1, the crux of the problem is the situation in which only default moves are made (cf. Figure 1(b)). We first prove that, if all robots perform only default moves, then they all converge to the same point (see Lemma 3.3 below).

Since we are assuming that only the vertex robots move, and that their movements depend only on the positions of other visible vertex robots, we may as well assume that all robots are vertex robots, and that their indices follow their order around the convex hull. Indeed, by the invariants observed in Section 3.2.1, all robots will remain vertex robots throughout the execution, and their ordering around the convex hull will remain the same. So, let r_{i-1}, r_i, r_{i+1} be three vertex robots, which appear on the boundary of $\mathcal{H}(t)$ consecutively in this order. Let r_i perform a default move at time t. Then, the new position of r_i is a convex combination of the current positions of these three robots, and precisely

$$r_i(t+1) = \frac{r_{i-1}(t)}{4} + \frac{r_i(t)}{2} + \frac{r_{i+1}(t)}{4}. \tag{1}$$

In general, as different sets of vertex robots are activated in several rounds, and nothing but default moves are made, the new location of each robot is always a convex combination of the *original* positions of all the robots, obtained by applying (1) to the set of active robots, at every round. In formulas,

$$r_i(t_0 + t) = \sum_{j=1}^{n} \alpha_{i,j,t} \cdot r_j(t_0),$$

with $\alpha_{i,j,t} \geqslant 0$ and $\sum_{j=1}^{n} \alpha_{i,j,t} = 1$, assuming that the robots start making only default moves at time t_0. Let $I = \{1, 2, \cdots, n\}$. We fix $j \in I$, and we let $w_{i,t} = \alpha_{i,j,t} - \alpha_{i-1,j,t}$, where indices are taken modulo n. We claim that

$$\lim_{t \to \infty} w_{1,t} = \lim_{t \to \infty} w_{2,t} = \cdots = \lim_{t \to \infty} w_{n,t} = 0. \tag{2}$$

If such a claim is true (for all $j \in I$), it implies that the robots get arbitrarily close to each other, as t grows. This, paired with the fact that $\mathcal{H}(t_0 + t + 1) \subseteq \mathcal{H}(t_0 + t)$ for every t, as observed in Section 3.2.1, allows us to conclude that the robots converge to the same limit point.

A proof of this statement can be obtained using the theory of convergence of asynchronous algorithms in the book [3]. Indeed, the update rule (1)

corresponds to performing time stepping on a Markov chain with circulant transition matrix

$$P = \begin{bmatrix} 1/2 & 1/4 & & & 1/4 \\ 1/4 & 1/2 & 1/4 & & \\ & 1/4 & 1/2 & \ddots & \\ & & \ddots & \ddots & 1/4 \\ 1/4 & & & 1/4 & 1/2 \end{bmatrix}.$$

It is proven in a statement on [3, page 435] that the time-stepping iteration converges even when performed asynchronously, under a model that generalizes our SSYNCH.

Nevertheless, we give here an alternative self-contained proof. First we reformulate the problem in the following terms.

Communicating Vessels. Suppose that n vessels containing water are arranged in a circle, and there is a pipe between each pair of adjacent vessels, regulated by a valve. At every second, some of the valves are opened and others are closed, in such a way that each of the n valves stays open for infinitely many seconds, in total. If a valve between two adjacent vessels stays open between seconds t and $t+1$, then $1/4$ of the surplus of water, measured at second t, flows from the fuller vessel to the emptier one. Our claim is that the amount of water converges to the same limit in all vessels, no matter how the valves are opened and closed. We call this problem Communicating Vessels.

In this formulation, the amount of water in the i-th vessel at time $t \in \mathbb{N}$ would be our previous $w_{i,t}$. However, here we somewhat abstract from the Mutual Visibility problem, and we consider a slightly more general initial configuration, in which the $w_{i,0}$'s are arbitrary real numbers.

This problem is a special case of a diffusion model on a simple circular graph. To solve it, we shall introduce a quadratic energy functional $\|w_t\|^2$, and prove that it is decreasing. The use of such an energy functional in this class of problems is well known in the literature (see for instance [10]), but the fact that the iteration is performed semi-synchronously on each node separately is less standard, so we need to do a little more work.

We set $v_{i,t} = 1$ if the valve between the i-th and the $(i+1)$-th vessel is open between time t and $t+1$ (indices are taken modulo n), and $v_{i,t} = 0$ otherwise.

It is easy to verify that activating robot r_i at time t in our previous discussion corresponds to setting $v_{i,t} = 1$ in the Communicating Vessels formulation.

Let us denote by w_t the vector whose i-th entry is $w_{i,t}$, and let $q_{i,t} = w_{i+1,t} - w_{i,t}$. We first prove an inequality on the Euclidean norms of the vectors w_t. Note that the inequality holds regardless of what assumptions are made on the opening pattern of the valves.

Lemma 3.1. *For every $t \in \mathbb{N}$,*

$$\|w_t\|^2 - \|w_{t+1}\|^2 \geqslant \frac{1}{4} \sum_{i=1}^{n} v_{i,t} \cdot q_{i,t}^2. \tag{3}$$

Proof. For brevity, let $a = w_{i-1,t}$, $b = w_{i,t}$, $c = w_{i+1,t}$; hence, $q_{i-1,t} = b - a$ and $q_{i,t} = c - b$.

Suppose first that $v_{i-1,t} = v_{i,t} = 1$, i.e., both valves connecting the i-th vessel with its neighbors are open. Then, $w_{i,t+1} = (a + 2b + c)/4$. We have

$$\frac{w_{i-1,t}^2}{4} + \frac{w_{i,t}^2}{2} + \frac{w_{i+1,t}^2}{4} - w_{i,t+1}^2 \geqslant \frac{q_{i-1,t}^2}{8} + \frac{q_{i,t}^2}{8}, \tag{4}$$

which can be obtained by dropping the term $(a - c)^2/16$ from the algebraic identity

$$\frac{a^2}{4} + \frac{b^2}{2} + \frac{c^2}{4} - \frac{(a + 2b + c)^2}{16} = \frac{(a - b)^2}{8} + \frac{(b - c)^2}{8} + \frac{(a - c)^2}{16}.$$

Now, suppose instead that $v_{i-1,t} = 1$ and $v_{i,t} = 0$. Then we have $w_{i,t+1} = (a + 3b)/4$, and

$$\frac{w_{i-1,t}^2}{4} + \frac{3w_{i,t}^2}{4} - w_{i,t+1}^2 = \frac{3q_{i-1,t}^2}{16} \geqslant \frac{q_{i-1,t}^2}{8}, \tag{5}$$

where the first equality comes from the identity

$$\frac{a^2}{4} + \frac{3b^2}{4} - \frac{(a + 3b)^2}{16} = \frac{3(a - b)^2}{16}.$$

If $v_{i-1,t} = 0$ and $v_{i,t} = 1$, an analogous argument gives

$$\frac{3w_{i,t}^2}{4} + \frac{w_{i+1,t}^2}{4} - w_{i,t+1}^2 \geqslant \frac{q_{i,t}^2}{8}. \tag{6}$$

17

Finally, if $v_{i-1,t} = v_{i,t} = 0$, $w_{i,t+1} = w_{i,t}$, and trivially

$$w_{i,t}^2 - w_{i,t+1}^2 = 0. \tag{7}$$

We sum for each $i \in I$ the relevant inequality among (4), (5), (6), (7), depending on the value of $v_{i-1,t}$ and $v_{i,t}$. Each of the terms $q_{i,t}^2/8$ appears twice if and only if $v_{i,t} = 1$, and the coefficients of the terms in $w_{i,t}^2$ sum to 1 for every i, hence we get (3). $\qquad \square$

From the previous lemma, it immediately follows that the sequence $(\|w_t\|)_{t \geqslant 0}$ is non-increasing. Since it is also bounded below by 0, it converges to a limit, which we call ℓ. Let $M_t = \max_{i \in I}\{w_{i,t}\}$ and $m_t = \min_{i \in I}\{w_{i,t}\}$. Observe that each entry of w_{t+1} is a convex combination of entries of w_t, hence $(M_t)_{t \geqslant 0}$ is non-increasing and $(m_t)_{t \geqslant 0}$ is non-decreasing. Therefore they both converge, and we let $M = \lim_{t \to \infty} M_t$ and $m = \lim_{t \to \infty} m_t$.

Corollary 3.1.

$$m \leqslant \frac{\ell}{\sqrt{n}} \leqslant M.$$

Proof. For every $t \in \mathbb{N}$, we have

$$nM_t^2 \geqslant \sum_{i=1}^n w_{i,t}^2 = \|w_t\|^2 \geqslant \ell^2,$$

which proves the second inequality. As for the first inequality, for every $\varepsilon > 0$ and large-enough t, we have $nm_t^2 \leqslant \|w_t\|^2 \leqslant \ell^2 + \varepsilon$. $\qquad \square$

For the next lemma, we let $V_i = \{t \in \mathbb{N} \mid v_{i,t} = 1\}$.

Lemma 3.2. *Suppose that $|V_i| = \infty$ for at least $n-1$ distinct values of $i \in I$. Then,*

$$M = m = \frac{\ell}{\sqrt{n}}.$$

Proof. Due to Corollary 3.1, it is enough to prove that $M - m = 0$. By contradiction, assume $M - m > 0$, and let $\delta = (M - m)/(n + 1) > 0$. We have

$$\lim_{t \to \infty} \left(\|w_t\|^2 - \|w_{t+1}\|^2\right) = \ell^2 - \ell^2 = 0,$$

18

hence there exists $T \in \mathbb{N}$ such that $\|w_t\|^2 - \|w_{t+1}\|^2 < \delta^2/4$ for every $t \geqslant T$. By Lemma 3.1,

$$\frac{q_{i,t}^2}{4} \leqslant \|w_t\|^2 - \|w_{t+1}\|^2 < \frac{\delta^2}{4}$$

for every $t \geqslant T$ and every i such that $v_{i,t} = 1$. This implies $|q_{i,t}| < \delta$, that is, a necessary condition for the valve between the i-th and the $(i+1)$-th vessel to be open at time $t \geqslant T$ is that $|w_{i+1,t} - w_{i,t}| < \delta$. Consider now the $n+1$ open intervals

$$(m, m + \delta), (m + \delta, m + 2\delta), \cdots, (m + n\delta, M),$$

each of width δ. Since $M_T \geqslant M$ and $m_T \leqslant m$, there are $w_{i,T}$'s above and below all these intervals. Moreover, by the pigeonhole principle, at least one of the intervals contains no $w_{i,T}$'s, for any $i \in I$. In other words, we can find a partition $I_1 \cup I_2 = I$, with I_1 and I_2 both non-empty, and a threshold value λ such that $w_{i,T} \leqslant \lambda$ for every $i \in I_1$, and $w_{i,T} \geqslant \lambda + \delta$ for every $i \in I_2$. Hence, at time T, only valves between entries of w_t whose indices belong to the same I_k can be open. It is now easy to prove by induction on $t \geqslant T$ the following facts:

- $\max_{i \in I_1}\{w_{i,t}\} \leqslant \lambda$,

- $\min_{i \in I_2}\{w_{i,t}\} \geqslant \lambda + \delta$,

- $v_{i,t} = 0$ whenever i and $i + 1$ belong to two different classes of the partition.

Since I_1 and I_2 are non-empty, there must be at least two distinct indices $i' \in I_1$ and $i'' \in I_2$ such that $i' + 1 \in I_2$ and $i'' + 1 \in I_1$ (where indices are taken modulo n). It follows that the i'-th and i''-th valve are never open for $t \geqslant T$, and this contradicts the hypothesis that $|V_i| < \infty$ for at most one choice of $i \in I$. $\qquad\square$

This solves the **Communicating Vessels** problem.

Corollary 3.2. *Under the hypotheses of Lemma 3.2, for every $i \in I$,*

$$\lim_{t \to \infty} w_{i,t} = \frac{\ell}{\sqrt{n}} = \frac{\sum_{j=1}^{n} w_{j,0}}{n}.$$

19

Proof. By Lemma 3.2, since $m_t \leqslant w_{i,t} \leqslant M_t$, all the limits coincide. Moreover, the sum of the $w_{i,t}$'s does not depend on t; hence their average, taken at any time, must be equal to the joint limit. □

Let us return to the Mutual Visibility problem, to prove our final lemma.

Lemma 3.3. *If, at every round, each robot makes a default move (cf. Figure 1(b)) or stays still, all external robots have their lights set to* Vertex, *and no new robots become vertex robots or terminate, then all robots' locations converge to the same limit point.*

Proof. As discussed at the beginning of Section 3.2.2, this is implied by (2). Recall that $w_{i,0} = \alpha_{i,j,0} - \alpha_{i-1,j,0}$, and hence $\sum_{i=1}^{n} w_{i,0} = 0$. Then, (2) follows immediately from Corollary 3.2. □

We are now ready to prove our main theorem.

Theorem 3.1. *Algorithm 1 solves* Mutual Visibility *for* RIGID SSYNCH *robots with 2-colored lights.*

Proof. If the initial convex hull is a line segment, it becomes a non-degenerate polygon as soon as one or more of the non-vertex robots are activated. It is also easy to observe (cf. Figure 2) that, from this configuration, the convex hull may never become a line segment. So the invariants discussed in Section 3.2.1 apply, possibly after a few initial rounds: no two robots will ever collide, and a vertex robot will never become a non-vertex robot.

Assume by contradiction that the execution never terminates. Note that a robot terminates if and only if all robots terminate. Indeed, if there are any non-vertex robots (whose lights are still set to *Off*), then each vertex robot can see at least one of them. Hence we are assuming that all robots execute the algorithm forever.

At some point, the set of vertex robots reaches a maximum $\mathcal{M} \subseteq \mathcal{R}$, and as soon as all of these robots have been activated, they permanently set their lights to *Vertex*. Let $T \in \mathbb{N}$ be a time at which all the robots in $\mathcal{M}$ have their lights set to *Vertex*. Suppose that there are external robots that are not vertex robots after time T, and let r be one such robot that is adjacent to a vertex robot r'. Then, after r' is activated and moves, r becomes a vertex robot as well, contradicting the maximality of $\mathcal{M}$. Hence the external robots are exactly the robots in $\mathcal{M}$, and no other robot may become external after time T.

If there is only one internal robot at time $t \geqslant T$, it becomes external as soon as it is activated, due to line 23 of the algorithm, which is impossible, as argued in the previous paragraph. Therefore there are at least two internal robots at every time $t \geqslant T$. On the other hand, if a vertex robot makes a non-default move at any time $t \geqslant T$, a new robot becomes external at time $t + 1$. Indeed, referring to Figure 1(a), the line uv passes through $p(t + 1)$ and $c(t+1)$, and no robot lies above this line at time $t+1$. Hence c becomes a new external robot, which again is impossible.

As a consequence, only default moves are made after time T. Moreover, no robot becomes external or becomes a vertex robot after time T, and no robot ever terminates. Therefore Lemma 3.3 applies, and the robots converge to the same limit point. But since there are at least two internal robots, this means that at least one of them has to move, implying that it becomes a vertex robot at some point (by the above assumptions, only vertex robots can move), a contradiction.

Hence the execution terminates, meaning that at some point one of the robots sees only vertex robots. This implies that there are no non-vertex robots, hence the configuration is strictly convex, all robots can see each other, and they all terminate without moving as soon as they are activated, thus solving the Mutual Visibility problem. □

4. Solving Mutual Visibility for Non-Rigid SSynch Robots

Here we give a protocol, Algorithm 2 (*Contain*), for the Mutual Visibility problem that works for Non-Rigid robots and the SSynch scheduler. Recall that, in the Non-Rigid model, the robots make unreliable moves, that is, the scheduler can stop them before they reach their destination point, but not before they have moved by at least a constant $\delta > 0$. Since these robots are weaker than the ones considered in Section 3, they will require lights of three possible colors, as opposed to two.

Our goal is also to design an algorithm that can be applied to robots in the Rigid ASynch model, as well as the Non-Rigid SSynch one. This model will be discussed in Section 5.1. In order to do this, we introduce a couple of extra technical subtleties into Algorithm 2, which are irrelevant here, but will turn out to be necessary in Section 5.1.

4.1. Description of Algorithm 2

Algorithm 2 consists of three phases, to be executed in succession: a *segment breaking* phase, an *interior depletion* phase, and a *vertex adjustments* phase. The first phase deals with the special configuration in which the robots are all collinear, and makes them not collinear. If the robots are not initially collinear, this phase is skipped. In the second phase, the internal robots move toward the boundary of the convex hull, thus forming a convex configuration, perhaps with some degenerate vertices. In the third phase the robots (which are now all external) make small movements to finally reach a strictly convex configuration. Three colors are used by the robots: $\mathcal{C} = \{\textit{Off}, \textit{External}, \textit{Adjusting}\}$. Initially, all robots' lights are set to *Off*.

For added clarity, in the algorithm the line numbers of instructions belonging logically to different phases are typeset in different colors, according to the following table.

Segment breaking	red
Interior depletion	green
Vertex adjustments	blue

Recall that we denote by $\mathcal{H}(t)$ the convex hull of the positions of all the robots at time $t \in \mathbb{N}$. In this section we also denote by $\mathcal{H}'(t)$ the convex hull of the positions of the internal robots at time $t \in \mathbb{N}$. Note that the "global" notions of $\mathcal{H}$ and $\mathcal{H}'$ may differ from the ones computed by the robots executing Algorithm 2, because a robot may be unable to see the positions of all the other robots in the swarm. In the following discussion, when referring to $\mathcal{H}$ and $\mathcal{H}'$, we will typically mean the "global" ones, unless we explicitly state otherwise.

We first describe the interior depletion phase, starting from a non-collinear initial configuration. To begin with, all the robots' lights are set to *Off*. As soon as an external robot is activated, it sets its own light to *External* (lines 34, 35) and does not move as long as it can still see robots whose light is *Off* (lines 26, 27). Note that a robot r that occupies a vertex of $\mathcal{H}'$ eventually becomes aware of it, by looking at the convex hull of the visible robots whose lights are *Off*. These may not all be internal robots, because perhaps not all external robots have been activated yet, but eventually r gets to see a good-enough approximation of a "neighborhood" of $\mathcal{H}'$, and it realizes it occupies one of its vertices.

Algorithm 2: Contain. Solving the Mutual Visibility problem for Non-Rigid SSynch robots and Rigid ASynch robots with 3-colored lights

---

Input: $\mathcal{V}$: set of robots visible to me (myself included) whose positions are expressed in a coordinate system centered at my location.

1 1 1 $r^* \longleftarrow$ myself
2 2 2 $\mathcal{P} \longleftarrow \{r.position \mid r \in \mathcal{V}\}$
3 3 3 $\mathcal{H} \longleftarrow$ convex hull of $\mathcal{P}$
4 4 4 $\partial\mathcal{H} \longleftarrow$ boundary of $\mathcal{H}$
5 **if** $|\mathcal{V}| = 1$ **then** Terminate
6 **else if** $|\mathcal{V}| = 2$ **then**
7   **if** $r^*.light = Adjusting$ **then**
8 $r^*.light \longleftarrow External$
9   Terminate
10   **else**
11 $r^*.light \longleftarrow Adjusting$
12   Move orthogonally to $\mathcal{H}$ by the length of $\mathcal{H}$

13 **else if** $\mathcal{H}$ *is a line segment* **then**
14   **if** $\forall r \in \mathcal{V} \setminus \{r^*\}$, $r.light = External$ **then**
15 $r^*.light \longleftarrow Adjusting$
16   Move orthogonally to $\mathcal{H}$ by any positive amount

---

Algorithm 2: Contain (continued).

17   **else if** $r^*.position \in \partial\mathcal{H}$ **then**

18 $a \longleftarrow$ my ccw-neighboring robot on $\partial\mathcal{H}$

19 $b \longleftarrow$ my cw-neighboring robot on $\partial\mathcal{H}$

20   **if** $r^*.light = Adjusting$ **then**

21   **if** $\forall r \in \mathcal{V}$, $r.light \neq Off$

22   **or** $\exists r \in \mathcal{V}$, $r.light = External$ **then**

23 $r^*.light \longleftarrow External$

24   **if** $a.light \neq Off$ **and** $b.light \neq Off$

25   **and** $(\mathcal{H} \setminus \partial\mathcal{H}) \cap \mathcal{P} = \varnothing$ **then** Terminate

26   **else if** $r^*.position$ is a non-degenerate vertex of $\mathcal{H}$

27   **and** $\forall r \in \mathcal{V}$, $r.light = External$ **then**

28 $r^*.light \longleftarrow Adjusting$

29   Move to $(a.position + b.position)/4$

30   **else**

31 $\mathcal{W} \longleftarrow \{r \in \mathcal{V} \mid r.light = Adjusting\}$

32   **if** $(|\mathcal{V}| = 3$ **and** *the internal angle of* $\mathcal{H}$ *at* $r^*.position$ *is acute*$)$

33   **or** $(|\mathcal{W}| > 1$ **and** $r^*.position$ *is a non-degenerate vertex of* $\mathcal{H})$

34   **or** $\mathcal{W} = \varnothing$ **then**

35 $r^*.light \longleftarrow External$

Algorithm 2: Contain (continued).

36 **else if** $\forall r \in \mathcal{V}$, $r.light \neq Adjusting$ **then**

37 $\quad$ $\mathcal{P}' \longleftarrow \{r.position \mid r \in \mathcal{V} \land r.light = Off\}$

38 $\quad$ $\mathcal{H}' \longleftarrow$ convex hull of $\mathcal{P}'$

39 $\quad$ $\partial\mathcal{H}' \longleftarrow$ boundary of $\mathcal{H}'$

40 $\quad$ **if** $|\mathcal{P}'| = 1$ **then**

41 $\quad\quad$ Move to a closest midpoint of a connected component of $\partial\mathcal{H} \setminus \mathcal{P}$

42 $\quad$ **else if** $|\mathcal{P}'| = 2$ **then**

43 $\quad\quad$ $\ell \longleftarrow$ line containing $\mathcal{H}'$

44 $\quad\quad$ $\mathcal{A} \longleftarrow$ right angle with axis of symmetry ℓ such that
$\quad\quad\quad\quad$ $\mathcal{A} \cap \mathcal{H}' = \{r^*.position\}$

45 $\quad\quad$ Move to any point of $(\mathcal{A} \cap \partial\mathcal{H}) \setminus \mathcal{P}$

46 $\quad$ **else if** $r^*.position$ *is a non-degenerate vertex of* $\mathcal{H}'$ **then**

47 $\quad\quad$ $\mathcal{A} \longleftarrow$ internal angle of $\mathcal{H}'$ whose vertex is $r^*.position$

48 $\quad\quad$ $\alpha \longleftarrow$ measure of $\mathcal{A}$

49 $\quad\quad$ $\ell \longleftarrow$ axis of symmetry of $\mathcal{A}$

50 $\quad\quad$ **if** $\alpha \leqslant \pi/2$ **then** $\alpha' \longleftarrow \alpha$

51 $\quad\quad$ **else** $\alpha' \longleftarrow \pi - \alpha$

52 $\quad\quad$ $\mathcal{A}' \longleftarrow$ angle of measure α' with axis of symmetry ℓ such that
$\quad\quad\quad\quad$ $\mathcal{A}' \cap \mathcal{H}' = \{r^*.position\}$

53 $\quad\quad$ $\mathcal{E} \longleftarrow \{p \in \partial\mathcal{H} \mid \exists a, b \in \mathcal{P} \setminus \mathcal{A}, p \in ab\}$

54 $\quad\quad$ **if** $(\mathcal{A}' \cap \mathcal{E}) \setminus \mathcal{P} \neq \varnothing$ **then** Move to any point of $(\mathcal{A}' \cap \mathcal{E}) \setminus \mathcal{P}$

So, when a robot understands that it lies on a vertex of $\mathcal{H}'$, it moves toward the boundary of $\mathcal{H}$, part of which is also identifiable by r. We distinguish three cases.

1. If r realizes it is the only internal robot, it moves toward the midpoint of an edge of the convex hull (line 41). To avoid bouncing back and forth at different turns, it always chooses the closest of such midpoints.
2. If r realizes that $\mathcal{H}'$ is a line segment and it occupies one endpoint of it, it moves like in Figure 3(a). That is, it moves to the boundary of $\mathcal{H}$, while remaining within a right angle oriented away from $\mathcal{H}'$ (lines 43–45).
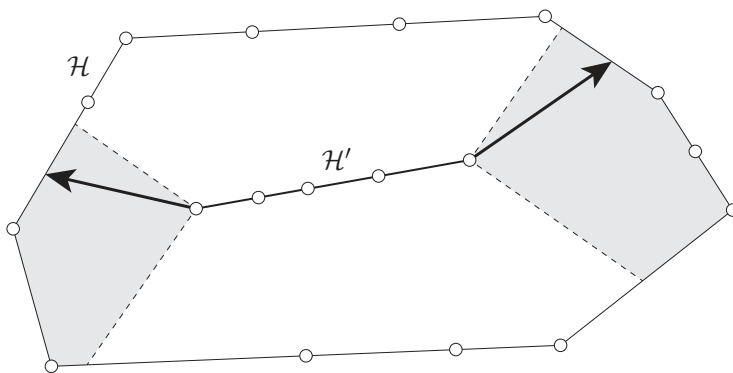
(a) Case with collinear internal robots
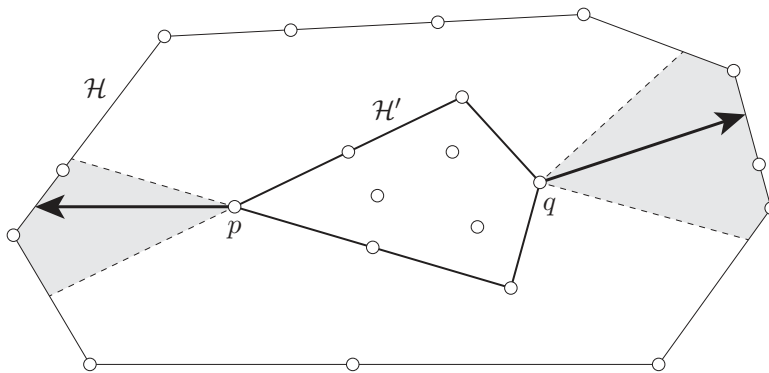
(b) General case

Figure 3: Interior depletion phase

26

3. Finally, if r "believes" that $\mathcal{H}'$ is a non-degenerate polygon and that it lies on one of its vertices, it moves as in Figure 3(b) (lines 47–54). Remember that r may believe so even if $\mathcal{H}'$ is actually degenerate, because some external robots may still be *Off*. However, r gets an approximation of $\mathcal{H}'$, which we call $\mathcal{H}'_r$, and it knows it lies on a vertex of $\mathcal{H}'_r$, implying that it also lies on a vertex of the "real" $\mathcal{H}'$. Now, if the internal angle of $\mathcal{H}'_r$ at $r(t)$ is acute, r moves as the robot in p in Figure 3(b): it moves to the boundary of $\mathcal{H}$ while remaining between the extensions of its two incident edges of $\mathcal{H}'_r$. Otherwise, if the angle is not acute, r moves as the robot in q in Figure 3(b): it moves to the boundary of $\mathcal{H}$ while staying between the two perpendiculars to its incident edges of $\mathcal{H}'_r$. Moreover, r actually performs the move only if it is sure that its destination point lies on the boundary of the "real" $\mathcal{H}$. For this reason, it has to check if the destination point computed as described above lies on a completely-visible edge of the observed convex hull whose endpoints are both set to *External* (line 53). For instance, in Figure 4, the robot in p cannot move to the gray area even if the robots in a and b are set to *External*, because the robot in q prevents the one in p from seeing the whole edge ab. On the other hand, the robot in q can move to its own gray area, provided that a and b are set to *External*. Indeed, the robot in q can see all of ab, and it is therefore sure that it is an edge of the "real" convex hull.
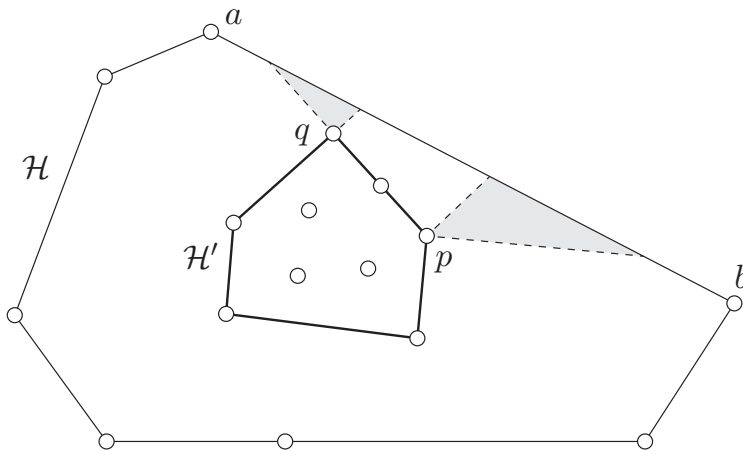

Figure 4: The robot in p cannot move even if a and b are set to *External*, because q may be hiding some other external robots, and ab may not be an edge of the convex hull

Now to the vertex adjustments phase. When a robot lies at a vertex of $\mathcal{H}$ and it sees only robots whose light is set to *External*, it makes the "default move" of Figure 1(b), where a and b are the locations of its two neighbors on $\mathcal{H}$ (line 29). Moreover, while doing so it also sets its light to the third value, *Adjusting*, as a "self-reminder" (line 28). So, when it is activated again, it knows it has already adjusted its position, and it terminates, after reverting its light to *External* (lines 20–25). This way we make sure that each vertex robot adjusts its position exactly once, and we ensure termination. When the adjustment is done, the robots at a and b are guaranteed to occupy vertices of $\mathcal{H}$, instead of lying in the middle of an edge. So, each external robot becomes a vertex robot at some point, then it adjusts its position while remaining a vertex, possibly making its adjacent robots become vertices as well, and it terminates. When all robots have terminated, the configuration is strictly convex, and therefore Mutual Visibility is solved.

Finally, the segment breaking phase deals with the special case in which all robots are initially collinear. Let robots r and s be the two endpoints of $\mathcal{H}$: as soon as one of them is activated (possibly both), it sets its light to *Adjusting*, moves orthogonally to $\mathcal{H}$, and then waits (lines 11, 12). Meanwhile, the other robots do not do anything until some conditions are met (lines 31–34). If only r moves, s realizes it (line 32) and sets its own light to *External* (and vice versa). If both r and s move together, some other robot realizes that it is a non-degenerate vertex of the convex hull and that it can see both r and s set to *Adjusting* (line 33): in this case, it sets itself to *External*. When r or s sees some robots set to *External*, it finally sets itself to *External*, as well (lines 22, 23). Additionally, it may terminate, provided that neither of its neighboring robots on the convex hull's boundary has still its light set to *Off* (line 24) and that it recognizes no robots as internal (line 25). This is to force r and s to make at least one default move in the unfortunate case that a third external robot is found between them after their initial move, or gets there during the interior depletion phase (refer to the complete discussion in Section 4.2.3). After this is done, the execution transitions seamlessly into one of the general cases.

If $n \leqslant 3$ this is not sufficient. Suppose first that $n = 3$. Then, r and s may move in such a way that the configuration remains centrally symmetric, with the middle robot q obstructing r and s. However, after moving once, r and s become *External* and terminate (lines 8, 9). Meanwhile q waits until it sees both r and s set to *External*, and finally it moves orthogonally to $\mathcal{H}$ (lines 13–16), thus solving Mutual Visibility also in this special case.

If $n = 2$, each robot moves once (lines 11, 12), and then it detects a situation in which it can safely terminate (lines 6–9).

4.2. Correctness of Algorithm 2

4.2.1. Interior Depletion Phase

We first prove that no collisions occur during the interior depletion phase, and then that the phase itself eventually terminates, with all the robots becoming external. In this section we will assume that the robots are not initially collinear. The collinear case will be discussed in Section 4.2.3, and it will be shown that is seamlessly transitions into one of the other cases.

It is easy to observe that, during the interior depletion phase, all external robots keep seeing (internal) robots whose lights are set to *Off*, and therefore none of them moves. On the other hand, no internal robot moves outside of the convex hull.

Observation 4.1. *If there are internal robots at time t, no external robot moves, and $\mathcal{H}(t) = \mathcal{H}(t + 1)$.*

Lemma 4.1. *If r and s are two internal robots at time t, then*

$$(r(t + 1) - r(t)) \bullet (s(t) - r(t)) \leqslant 0.$$

Proof. If r is not activated at time t, or it is activated but it does not move, then the left-hand side is zero, and therefore the inequality holds. Suppose now that r moves by a positive amount, so $r(t + 1) - r(t)$ is not the null vector. Let ℓ be the line through $r(t)$ that is orthogonal to the segment $r(t)r(t + 1)$. By construction, r moves in such a way that $r(t + 1)$ lies in the open half-plane bounded by ℓ that does not contain $\mathcal{H}'(t)$ (note that this holds *a fortiori* also if some external robots have not set their lights to *External* yet, and therefore the $\mathcal{H}'$ computed by r is larger than the real one). Since $s(t) \in \mathcal{H}'(t)$, $s(t)$ lies on ℓ or in the half-plane bounded by ℓ that does not contain $r(t + 1)$. This is equivalent to saying that the dot product between $r(t + 1) - r(t)$ and $s(t) - r(t)$ is not positive. $\square$

Lemma 4.2. *As long as there are internal robots, no collisions occur.*

Proof. If there are internal robots, every external robot sees robots whose light is set to *Off*, and hence it does not move. By construction, the internal robots avoid moving on top of external robots, and therefore there can be no collision involving external robots.

Suppose by contradiction that two robots r and s that are internal at time t collide for the first time at $t+1$, and therefore $r(t+1) = s(t+1) = p$. By Lemma 4.1 applied to r and s, we have

$$(p - r(t)) \bullet (s(t) - r(t)) \leqslant 0. \tag{8}$$

Applying Lemma 4.1 again with r and s inverted, we also have

$$(p - s(t)) \bullet (r(t) - s(t)) \leqslant 0. \tag{9}$$

Adding 8 and 9 together and doing some algebraic manipulations, we obtain

$$(p - r(t)) \bullet (s(t) - r(t)) + (p - s(t)) \bullet (r(t) - s(t)) \leqslant 0,$$

$$(s(t) - r(t)) \bullet ((p - r(t)) - (p - s(t))) \leqslant 0,$$

$$(s(t) - r(t)) \bullet (s(t) - r(t)) \leqslant 0.$$

The latter is equivalent to $\|s(t) - r(t)\| \leqslant 0$, implying that $r(t) = s(t)$. This contradicts the fact that r and s collide for the first time at $t+1$. $\square$

We still have to prove that the interior depletion phase terminates, that is, eventually all robots become external. Due to Observation 4.1, when a robot becomes external, it stops moving and remains external, at least as long as there are other internal robots. Thus, if by contradiction this phase does not terminate, the set of internal robots reaches a non-empty minimum, and from that time on no new robot becomes external. After possibly some more turns, say at time $T \in \mathbb{N}$, all external robots have been activated and have set their lights to *External*, and hence no robot changes its light any more.

In the following lemmas, we will show that these assumptions on T yield a contradiction. We will prove that, if $\mathcal{H}'(T)$ is a non-degenerate polygon, then either its area or its diameter will grow unboundedly. Therefore, at some point in time, $\mathcal{H}'$ will not be a subset of $\mathcal{H}$ any more. (The analysis when $\mathcal{H}'(T)$ is a degenerate polygon is easy, and it will be carried out in the proof of Lemma 4.7.)

Recall that, due to line 50 of Algorithm 2, when a robot computes its destination, it remains within the extensions of its incident edges of $\mathcal{H}'(t)$. Hence, referring to Figure 5, it is easy to observe the following.

Observation 4.2. *Let robots r and s lie at adjacent vertices of $\mathcal{H}'(t)$ at time $t \geqslant T$, and let the area of $\mathcal{H}'(t)$ be positive. Then, $r(t+1)$ and $s(t+1)$ lie on the same side of the line through $r(t)$ and $s(t)$ (or possibly on the line itself).*

Lemma 4.3. *If $t \geqslant T$ and the area of $\mathcal{H}'(t)$ is positive, then $\mathcal{H}'(t) \subseteq \mathcal{H}'(t+1)$.*

Proof. Let $\mathcal{R}' \subset \mathcal{R}$ be the set of robots that lie at vertices of $\mathcal{H}'(t)$, at time $t \geqslant T$. Let $\mathcal{P}$ be the polygon (illustrated in Figure 5 as a thick dashed polygon) whose vertices are the locations at time $t+1$ of the robots of $\mathcal{R}'$, taken in the same order as they appear around the boundary of $\mathcal{H}'$. Note that, since the robots are NON-RIGID and not all of them are necessarily activated at time t, $\mathcal{P}$ is not necessarily a convex polygon.

Because the property stated in Observation 4.2 holds for all the edges of $\mathcal{H}'(t)$ and $\mathcal{P}$, we have that $\mathcal{H}'(t) \subseteq \mathcal{P}$. But, by definition of T, none of the robots of $\mathcal{R}'$ ever becomes external, and hence $\mathcal{H}'(t+1)$ is the convex hull of $\mathcal{P}$. We conclude that $\mathcal{H}'(t) \subseteq \mathcal{P} \subseteq \mathcal{H}'(t+1)$. □

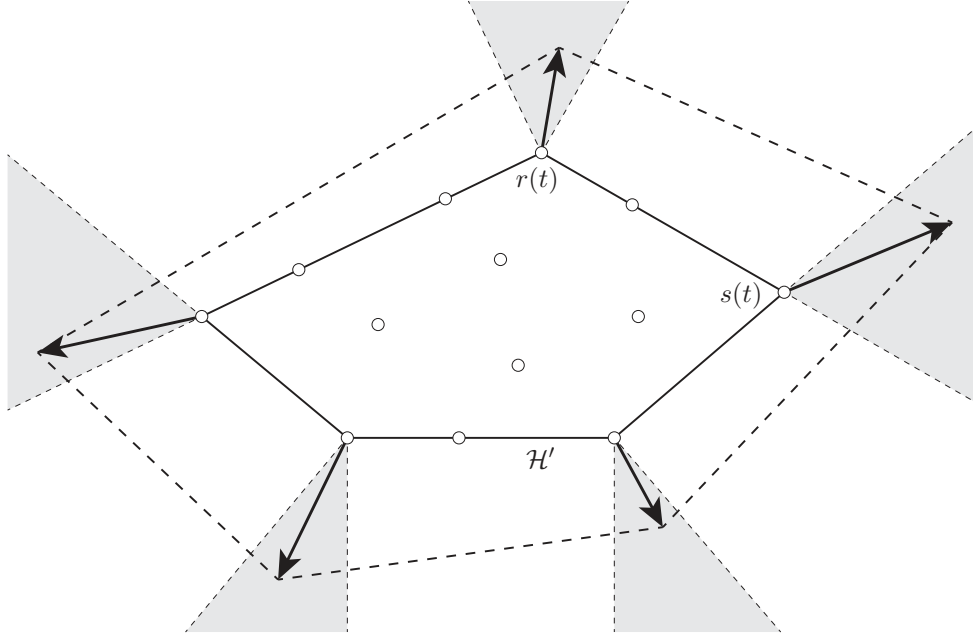Figure 5: Combined motion of internal robots (external robots are not shown)

Corollary 4.1. *For $t \geqslant T$, the area of $\mathcal{H}'(t)$ and the diameter of $\mathcal{H}'(t)$ do not decrease as t increases.*

31

Proof. By Lemma 4.3, if $T \leqslant t_1 \leqslant t_2$, then $\mathcal{H}'(t_1) \subseteq \mathcal{H}'(t_2)$. Hence the area of $\mathcal{H}'(t_1)$ cannot be greater than the area of $\mathcal{H}'(t_2)$, and the diameter of $\mathcal{H}'(t_1)$ cannot be greater than the diameter of $\mathcal{H}'(t_2)$. □

Corollary 4.2. *If r is an internal robot at time $t \geqslant T$, and $r(t)$ is not a non-degenerate vertex of $\mathcal{H}'(t)$, then r is internal at any time $t' \geqslant t$, and $r(t')$ is not a non-degenerate vertex of $\mathcal{H}'(t')$.*

Proof. By Lemma 4.3, if $t' \geqslant t$, then $\mathcal{H}'(t) \subseteq \mathcal{H}'(t')$. Moreover, according to line 46 of Algorithm 2, an internal robot that does not lie at a degenerate vertex of $\mathcal{H}'$ does not move. It follows that, after time t, robot r will not move, hence it will remain internal, and it will never lie at a non-degenerate vertex of $\mathcal{H}'$. □

Recall that, due to line 53 of Algorithm 2, a robot on the perimeter of $\mathcal{H}'$ is able to move only if it completely sees an entire edge of $\mathcal{H}$. Next we prove that, after time T, there exists at least one robot that is able to move.

Lemma 4.4. *At any time after T, there is a robot that, if activated, makes a non-null movement.*

Proof. Suppose for a contradiction that no robot is able to move, and let p_0 be a non-degenerate vertex of $\mathcal{H}'$. If $\mathcal{A}$ is the internal angle of $\mathcal{H}'$ at p_0, we let p_0' be the point at which the bisector of $\mathbb{R}^2 \setminus \mathcal{A}$ intersects the perimeter of $\mathcal{H}$, as Figure 6 shows.
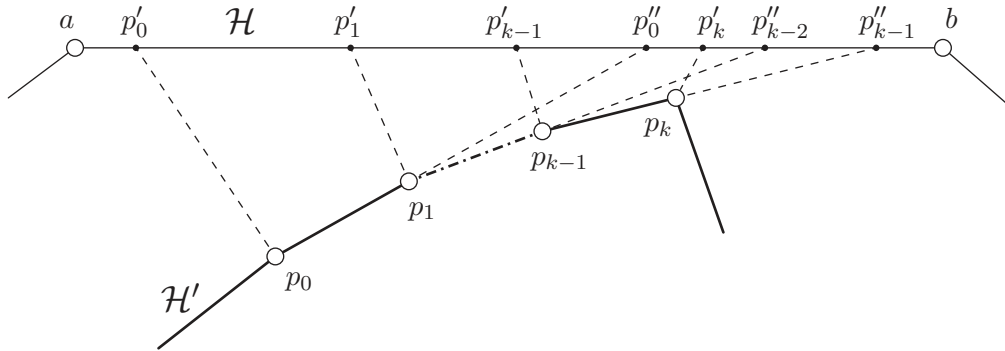

Figure 6: The robots in $p_0, p_1, \cdots, p_{k-1}$ are unable to move; the one in p_k is able to move

Let ab be an edge of $\mathcal{H}$ such that $p_0' \in ab$ (note that p_0' may coincide with either a or b). Since $\mathcal{H}$ and $\mathcal{H}'$ are convex, either the segment ap_0' is

completely visible to p_0, or the segment bp_0' is. Without loss of generality, let ap_0' be completely visible to p_0, i.e., $ap_0' \subset \mathbb{R}^2 \setminus \mathcal{A}$. By definition of T, both robots in a and b have their lights let to *External*. Also, note that p_0' lies within $\mathcal{A}'$, as computed by the robot in p_0 executing line 52 of Algorithm 2. Hence, by line 53, the robot in p_0 can move, provided that $b \in \mathbb{R}^2 \setminus \mathcal{A}$. But by assumption no robot can move, and therefore there must be one robot occupying a non-degenerate vertex of $\mathcal{H}'$ neighboring p_0, say p_1, such that the ray from p_0 through p_1 intersects the segment ab, say in p_0''.

Observe that p_1 is strictly closer to the line ab than p_0. By the convexity of $\mathcal{H}'$, the bisector of the explementary of the internal angle at p_1 intersects the segment ab, say in p_1'. In fact, p_1' lies between p_0' and p_0'' (refer to Figure 6). Also, p_1 can see all the points in the segment ap_1'. As we argued for p_0 in the previous paragraph, there must be another vertex p_2 of $\mathcal{H}'$, closer to the line ab, that prevents p_1 from seeing the entire segment ab.

Proceeding in this fashion, we obtain a sequence p_0, p_1, p_2, p_3, $\cdots$ of vertices of $\mathcal{H}'$, which get closer and closer to the line ab. Since these vertices must be all distinct, and there are only finitely many robots in the swarm, there must be one last element of the sequence, p_k. It follows that p_k can see all of ab, and the corresponding angle bisector intersects ab as well, say in p_k'. This implies that the robot in p_k can actually move to a neighborhood of p_k', contradicting our assumption. $\qquad\square$

As a consequence of Corollary 4.2, no new robot becomes a non-degenerate vertex of $\mathcal{H}'$ after time T, but some robots may indeed cease to be non-degenerate vertices of $\mathcal{H}'$, and stop moving forever. Hence, at some time $T' \geqslant T$, the set of robots that lie at non-degenerate vertices of $\mathcal{H}'$ reaches a minimum $\mathcal{M} \subset \mathcal{R}$. By Lemma 4.3, the area of $\mathcal{H}'$ is positive at every time $t \geqslant T'$, and hence $|\mathcal{M}| \geqslant 3$.

In the next lemma we prove two fundamental properties of $\mathcal{H}'$ that hold after time T'.

Lemma 4.5. *After time T', the following statements hold.*

    a. *The cyclic order of the robots of $\mathcal{M}$ around $\mathcal{H}'$ is preserved.*

    b. *The length of the shortest edge of $\mathcal{H}'(t)$ does not decrease as t increases.*

Proof. Similarly to Section 3, we call the gray regions in Figures 3 and 5 the *movement regions* of the respective robots.

Let $t \geqslant T'$, and consider the polygon $\mathcal{P}$ as defined in the proof of Lemma 4.3. By our assumptions on $\mathcal{M}$, $\mathcal{P}$ is a convex polygon, or else some robot would cease to occupy a vertex of $\mathcal{H}'$. Hence $\mathcal{P} = \mathcal{H}'(t+1)$ and, by Lemma 4.3, $\mathcal{H}'(t) \subseteq \mathcal{H}'(t+1)$. Let $r, s \in \mathcal{M}$ occupy two adjacent edges of $\mathcal{H}'$ at time t, and let ℓ be the axis of the segment $r(t)s(t)$. It follows from Algorithm 2 that ℓ separates the movement regions of r and s at time t (cf. Figure 5). This, paired with the fact that the movement region of a robot of $\mathcal{M}$ at time t does not intersect the interior of $\mathcal{H}'$, implies (a).

Now, to prove (b), it is sufficient to note that, with the previous paragraph's notation, the distance between the movement regions of r and s at time t is precisely the distance between $r(t)$ and $s(t)$ (see Figure 5). Therefore, the segment $r(t+1)s(t+1)$ is not shorter than $r(t)s(t)$, implying that the length of the shortest edge of $\mathcal{H}'$ cannot decrease. $\qquad \square$

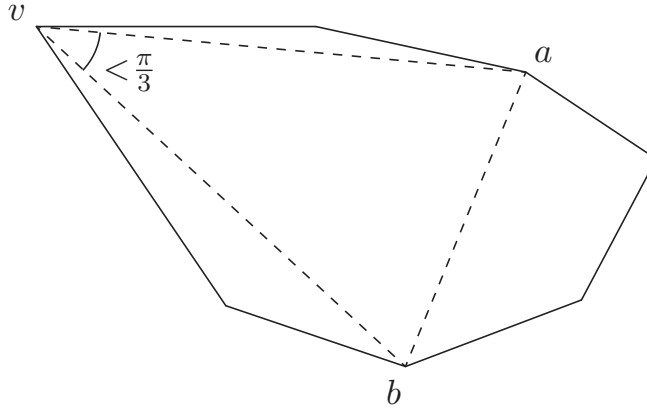We need one last geometric observation.

Figure 7: The longest edge of triangle $\triangle vab$ is not ab

Lemma 4.6. *If the internal angle at vertex v of a convex polygon has measure less than $\pi/3$, then the diameter of the polygon is the distance between v and another vertex.*

Proof. The diameter of a polygon is the longest distance between two of its vertices. Suppose for a contradiction that vertices a and b have the maximum distance, with $a \neq v \neq b$, as in Figure 7. Then, since the polygon is convex, the angle $\angle avb$ is containted in the internal angle at v, and therefore its

measure is less than $\pi/3$. Since the sum of the internal angles of a triangle is π, it follows that either $\angle bav > \pi/3$ or $\angle vba > \pi/3$. By the law of sines, in the first case vb is longer than ab, and in the second case va is longer than ab. In both cases, ab is not the longest segment joining two vertices of the polygon, which is a contradiction. $\qquad\square$

We are finally ready to prove the termination, and therefore the correctness, of the interior depletion phase.

Lemma 4.7. *If Algorithm 2 is executed from a non-collinear configuration, after finitely many turns all robots become external, no robot's light is set to* Adjusting, *and no two robots collide.*

Proof. The non-collision part has already been proven in Lemma 4.2, so we need to prove that eventually all robots become external.

By assumption $\mathcal{H}$ has positive area, and we have to show that all robots become external in finitely many turns. If there is just one internal robot, it keeps moving somewhere within $\mathcal{H}$, until it either becomes external, or all external robots have been activated. When all external robots have their lights set to *External*, if there is still a single internal robot, it keeps moving toward the same edge of $\mathcal{H}$, until it finally reaches it.

If there there are exactly two internal robots, they move as shown in Figure 3(a). It is easy to see that, each time at least one of the two internal robots moves (by at least $\delta > 0$), their distance increases by more than $\delta/\sqrt{2}$. Therefore, after finitely many turns, at least one of the two internal robots becomes external, and at most one internal robot remains.

Suppose now that there are at least three internal robots, but $\mathcal{H}'$ has null area, that is, all the internal robots are collinear. Then, according to Algorithm 2, only the two endpoint robots of $\mathcal{H}'$ are allowed to move, as Figure 3(a) shows. If they move in such a way that the internal robots keep remaining collinear, eventually one of them reaches the boundary of $\mathcal{H}$, and there is one less internal robot. Otherwise, they reach a situation in which $\mathcal{H}'$ has strictly positive area.

Therefore we may assume that $\mathcal{H}'$ has positive area, and we suppose for a contradiction that some internal robots never become external. By the previous lemmas and observations, we know that at some time T' the situation becomes "stable". Specifically, $\mathcal{H}$ never changes, the set $\mathcal{M}$ of robots that occupy non-degenerate vertices of $\mathcal{H}'$ keeps remaining the same, and these robots' positions preserve their order around $\mathcal{H}'$, by Lemma 4.5.a.

Also, the area and diameter of $\mathcal{H}'$ do not decrease, by Corollary 4.1. Let $a(t)$ be the length of the shortest edge of $\mathcal{H}'(t)$. By Lemma 4.5.b, we know that $a(t)$ is a weakly increasing function of $t \geqslant T'$.

Suppose that, at some time $T'' \geqslant T'$, some robot $r \in \mathcal{M}$ that is able to move is activated. By lines 53, 54 of Algorithm 2, the destination point of r lies on the boundary of $\mathcal{H}$. Hence, the adversary must stop r before it reaches its destination, or it would become external. But this cannot happen before r has moved by at least δ, implying that $\|r(T''+1) - r(T'')\| \geqslant \delta$. We distinguish two cases, based on the measure α of the internal angle of $\mathcal{H}'(T'')$ corresponding to vertex $r(T'')$. We will prove that, if $\alpha < \pi/3$, then the square of the diameter of $\mathcal{H}'(T'')$ increases by at least a constant; if $\alpha \geqslant \pi/3$, then the area of $\mathcal{H}'(T'')$ increases by at least a constant.
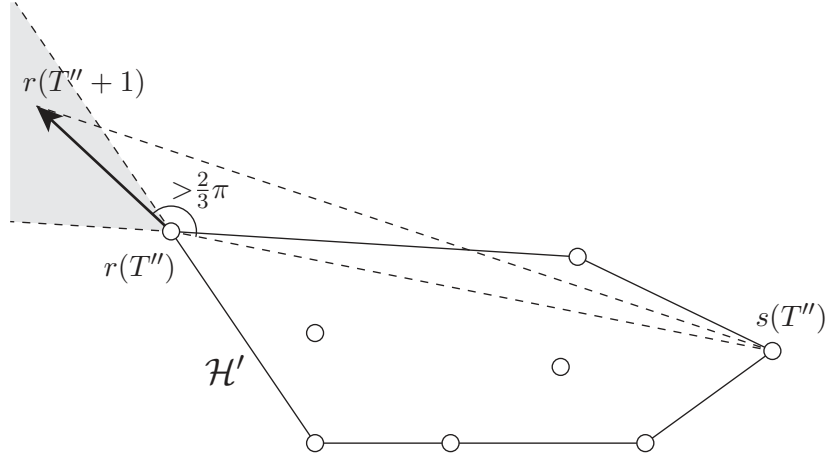
Figure 8: The square of the diameter of $\mathcal{H}'$ increases by at least δ^2

Suppose that $\alpha < \pi/3$. Let $\mathcal{D}(t)$ the diameter of $\mathcal{H}'(t)$, and let $s \in \mathcal{M}$ be such that $\mathcal{D}(T'') = \|r(T'') - s(T'')\|$. Due to line 50 of Algorithm 2, and referring to Figure 8, it is easy to prove that

$$\frac{2}{3}\pi < \angle s(T'')r(T'')r(T''+1) \leqslant \pi.$$

Hence

$$\cos(\angle s(T'')r(T'')r(T''+1)) < -\frac{1}{2}.$$

Because $\mathcal{H}'(T'') \subseteq \mathcal{H}'(T''+1)$, it follows that $s(T'') \in \mathcal{H}'(T''+1)$, and $\mathcal{D}(T''+1) \geqslant \|r(T''+1) - s(T'')\|$. Therefore, by the law of cosines applied to triangle $\triangle s(T'')r(T'')r(T''+1)$,

$$\mathcal{D}^2(T''+1) \geqslant \|r(T''+1) - s(T'')\|^2 > \mathcal{D}^2(T'') + \delta^2 + \mathcal{D}(T'') \cdot \delta > \mathcal{D}^2(T'') + \delta^2.$$

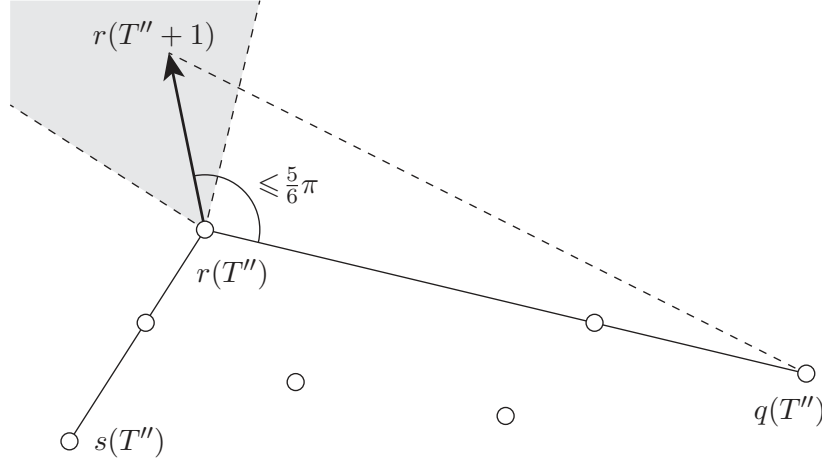Hence, in this case, the square of the diameter of $\mathcal{H}'$ increases by at least δ^2.

Figure 9: The area of $\mathcal{H}'$ increases by at least $a(T') \cdot \delta/8$

Let $\alpha \geqslant \pi/3$, and let $q, s \in \mathcal{M}$ occupy the two vertices of $\mathcal{H}'(T'')$ adjacent to vertex $r(T'')$, in such a way that

$$\angle q(T'')r(T'')r(T''+1) \leqslant \angle r(T''+1)r(T'')s(T''),$$

as Figure 9 shows. It follows that

$$\frac{\pi}{2} \leqslant \angle q(T'')r(T'')r(T''+1) \leqslant \frac{2\pi - \alpha}{2} \leqslant \frac{5}{6}\pi.$$

Then, recalling that $\mathcal{H}'(T'') \subseteq \mathcal{H}'(T''+1)$, we have that the area of $\mathcal{H}'$ increases at least by the area of the triangle $\triangle q(T'')r(T'')r(T''+1)$, which in turn is at least

$$\frac{1}{2} \cdot \|q(T'') - r(T'')\| \cdot \|r(T''+1) - r(T'')\| \cdot \sin(\angle q(T'')r(T'')r(T''+1))$$

$$\geqslant \frac{1}{2} \cdot a(T'') \cdot \delta \cdot \sin(5\pi/6) \geqslant a(T') \cdot \delta/8.$$

37

Concluding, every time a robot of $\mathcal{M}$ moves, either the square of the diameter of $\mathcal{H}'$ increases by at least δ^2, or the area of $\mathcal{H}'$ increases by at least $a(T') \cdot \delta/8$. But Lemma 4.4 states that there is always at least one robot of $\mathcal{M}$ that is able to move, and therefore the robots will move infinitely many times, due to the fairness of the SSYNCH scheduler. From Corollary 4.1, and from the fact δ^2 and $a(T') \cdot \delta/8$ are positive constants, it follows that either the diameter or the area of $\mathcal{H}'$ grows unboundedly. This contradicting the fact that $\mathcal{H}'(t) \subseteq \mathcal{H}(t)$, and that $\mathcal{H}(t)$ is independent of t. □

4.2.2. Vertex Adjustments Phase

Due to Lemma 4.7, all robots become external at some point, and it remains to show that they finally reach a *strictly* convex configuration and correctly terminate. This turns out to be a significantly easier task.

Lemma 4.8. *If Algorithm 2 is executed from a configuration in which all robots are external and no robot's light is set to* Adjusting, *then after finitely many turns all robots have terminated, no two of them have collided, and the configuration is strictly convex.*

Proof. In the vertex adjustments phase, each robot eventually sets its own light to *External* (if it has not already done so in the interior depletion phase). Meanwhile, as soon as a vertex robot r sees only robots whose lights are set to *External*, it sets its own light to *Adjusting* and makes a default move, as in Figure 1(b). Recall that robots are NON-RIGID, hence a vertex robot may be stopped before reaching its destination, but not before having moved by at least $\delta > 0$. When r is activated again, it sees itself in the *Adjusting* state and, if it sees a robot set to *Off*, it necessarily also sees a robot set to *External*. Indeed, after the default move, r can see all the robots in the swarm. Note that, if a robot is set to *Adjusting*, neither of its two neighbors on the perimeter of the convex hull can be *Off*. Hence, if r sees a robot set to *Off*, and since it sees itself set to *Adjusting*, it must also see a robot set to *External*. Therefore r sets its light back to *External* (lines 21–23), thus allowing other robots to move. r also terminates because, as already noted, its two neighbors cannot be *Off* (line 24) and, since r sees every robot in the swarm and the configuration is convex, r does not see any internal robots (line 25).

As observed in Section 3, where a similar procedure was used to reduce the size of the convex hull while increasing the set of vertex robots, when a robot occupying a vertex of the convex hull moves, it becomes a vertex of the

new convex hull. On the other hand, if a robot r lies in the interior of an edge of the convex hull and one of its two neighbors s lies on a vertex, then, as soon as s moves, r becomes a vertex of the new convex hull. Hence, eventually all robots become vertices of the convex hull. After that, whenever a robot is activated, it permanently sets its light to *External* and terminates. It follows that eventually all robots terminate in a strictly convex configuration.

Moreover, by the observations already made in Section 3.2.1, and referring to Figure 2, it is clear that no collisions can occur in this phase. □

4.2.3. Segment Breaking Phase

From Lemmas 4.7 and 4.8, the correctness of Algorithm 2 immediately follows, provided that the robots are not initially collinear. This case is considered in the following.

Theorem 4.1. *Algorithm 2 solves* Mutual Visibility *for* NON-RIGID SSYNCH *robots with 3-colored lights.*

Proof. Due to Lemmas 4.7 and 4.8, we only have to show that the case in which the robots are initially collinear correctly evolves into one of the other cases. If $n \leqslant 3$, this is easy to verify through a case analysis, following the algorithm's description of Section 4.1.

So, let $n \geqslant 4$, and let robots r and s initially occupy the vertices of the line segment $\mathcal{H}(0)$. Nothing happens until r or s is activated; then, at least one of them becomes *Adjusting* and moves orthogonally to $\mathcal{H}$. After r or s has moved, some robots eventually become *External* (line 35). Indeed, if both r and s move, then all other robots can see both of them. In particular, there is at least one such robot occupying a non-degenerate vertex of the convex hull, which sees two *Adjusting* robots, and therefore becomes *External* (line 33). If only r moves (or vice versa), then s sees only three robots (including itself), and its corresponding convex hull angle is acute, hence it becomes *External* (line 32). In the latter case, only s is allowed to become *External*, while the other robots wait, because their corresponding angles are not acute. When a robot has become *External*, any robot that was *Adjusting* can see it, and hence becomes *External* as well. If, in addition, neither of its two neighbors is *Off*, it also terminates (note that an *Adjusting* robot must be located on the convex hull's boundary).

At this point the execution proceeds normally, except that there may be one of two vertex robots that have already terminated, and we have to show that this does not prevent the others from forming a strictly convex

configuration. Obviously the interior depletion phase causes no trouble and it is carried out correctly, but the vertex adjustments phase might "get stuck". We will prove that this is not the case. Clearly, if at most one robot has terminated, all robots except perhaps one are able to move in the vertex adjustments phase, and therefore they all become non-degenerate vertices. If r and s initially move in the same direction, they become neighboring vertices, and all the other robots become consecutive external robots. It is easy to see that in this case the external robots are still able to make adjusting movements in cascade, and become non-degenerate vertices.
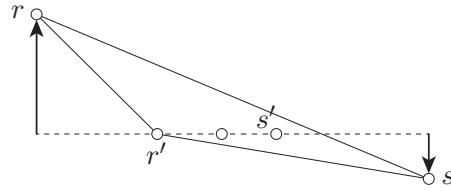
(a) s' becomes internal
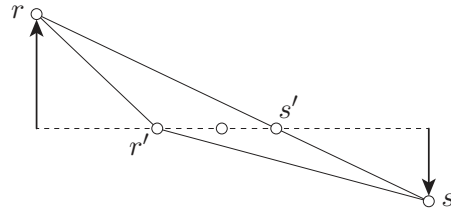
(b) s' becomes a degenerate vertex of the convex hull
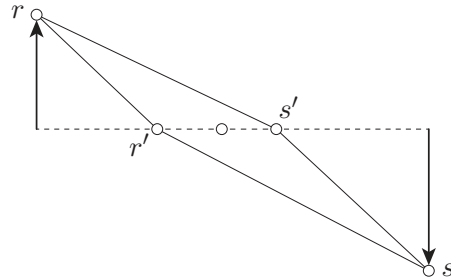
(c) s' becomes a non-degenerate vertex of the convex hull

Figure 10: Possible evolutions of a collinear configuration in which the endpoints move in opposite directions

Finally, suppose that r and s initially move in opposite directions. Let r' be the robot closest to r, and s' the robot closest to s. Since $n \geqslant 4$, $r' \neq s'$.

40

As already noted, at least one between r' and s' becomes a non-degenerate vertex of the convex hull, say r'. On the other hand, depending on how much r and s move, s' may become internal, or a degenerate or non-degenerate vertex of the convex hull. If s' becomes internal, as in Figure 10(a), then both r and s see an internal robot: indeed, after the move, r and s can see all robots. Then, r and s become *External* but do not terminate (line 25), and everything works as intended in the later phases. If s' becomes a degenerate vertex of the convex hull, as in Figure 10(b), then it is an *Off* neighbor of both r and s, which once again become *External* but do not terminate (line 24). Again, the execution transitions seamlessly into another phase. If s' becomes a non-degenerate vertex of the convex hull, as well as r', then all the robots between them become internal. Note that, in this case, r and s may terminate (indeed, they may not be able to see each other, and hence they may not realize that there are internal robots), but they do not lie at adjacent vertices of $\mathcal{H}$, due to the presence r' and s'. After the interior depletion phase, r' and s' are able to adjust, thus enabling all other external robots to become non-degenerate vertices, in cascade. $\qquad\square$

5. Solving Mutual Visibility for ASynch Robots

In this section we briefly touch on the ASynch model. In the Rigid case, we show that Algorithm 2 solves the Mutual Visibility problem. In the Non-Rigid case, we show how to solve Mutual Visibility assuming that the robots agree on the direction of one coordinate axis.

5.1. Rigid ASynch *Robots*

Algorithm 2 turns out to solve the Mutual Visibility problem for Rigid ASynch robots, as well. For the interior depletion phase, the collision avoidance proof gets slightly more complex, but termination is easier to prove. On the other hand, the vertex adjustments phase and the segment breaking phase work almost in the same way.

First we state an equivalent of Lemma 4.1. The only difference is that, instead of a generic time $t \in \mathbb{N}$, now we have to consider a specific time $t \in \mathbb{R}$ at which a robot r performs a *Look*. Also, instead of considering the position of r at time $t + 1$, we consider the destination point computed after such a *Look*. After these changes, the proof of Lemma 4.1 works in the ASynch case as well, and therefore we have the following.

Lemma 5.1. *Let* RIGID ASYNCH *robots execute Algorithm 2, and let r and s be two internal robots at time $t \in \mathbb{R}$. If r executes a Look phase at time t, and the next destination point of r is p, then*

$$(p - r(t)) \bullet (s(t) - r(t)) \leqslant 0.$$

$\square$

The previous lemma can be used to prove that no collisions occur during the interior depletion phase.

Lemma 5.2. *If* RIGID ASYNCH *robots execute Algorithm 2 from a non-collinear configuration, no collisions occur as long as there are internal robots.*

Proof. Suppose for a contradiction that the internal robot r performs a *Look* at time t, then robot s performs a *Look* at time $t' \geqslant t$, and they collide at time $t'' \geqslant t'$, in $r(t'') = s(t'')$. We may further assume that this is the first collision between the two robots, and therefore $r(t) \neq s(t)$. Because the model is RIGID and each internal robot's destination point is on the convex hull, it follows that each internal robot makes exactly one move and then becomes external. Therefore, $r(t)$, $r(t')$, $r(t'')$, and the destination point of r are all collinear, and the same holds for s. Additionally, we have $s(t) = s(t')$ (see Figure 11).
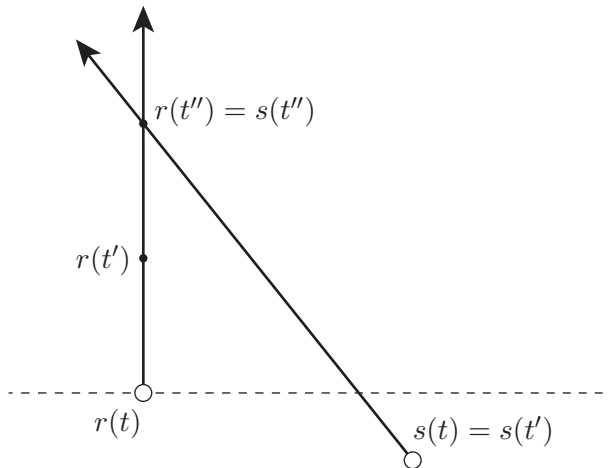

Figure 11: Two colliding internal robots

By Lemma 5.1 applied to s at time t', and because $s(t'')$ lies between $s(t')$ and the destination point of s, we have

$$(s(t'') - s(t')) \bullet (r(t') - s(t')) \leqslant 0.$$

On the other hand, $\|s(t'') - s(t')\| \geqslant 0$, implying that

$$(s(t'') - s(t')) \bullet (s(t') - s(t'')) \leqslant 0.$$

By adding the two inequalities together, we obtain

$$(s(t'') - s(t')) \bullet (r(t') - s(t'')) \leqslant 0.$$

Recall that $s(t'') = r(t'')$ and that $r(t')$ lies between $r(t)$ and $r(t'')$, and therefore the last inequality implies

$$(s(t'') - s(t')) \bullet (r(t) - r(t')) \leqslant 0,$$

hence

$$(s(t'') - s(t')) \bullet (r(t) - s(t') + s(t') - r(t')) \leqslant 0,$$

and

$$(s(t'') - s(t')) \bullet (r(t) - s(t')) \leqslant (s(t'') - s(t')) \bullet (r(t') - s(t')).$$

But we already know that the right-hand side is not positive, hence so is the left-hand side:

$$(s(t'') - s(t')) \bullet (r(t) - s(t')) \leqslant 0.$$

Now, by Lemma 5.1 applied to r at time t, and recalling that $r(t'')$ lies between $r(t)$ and the destination point of r, we have

$$(r(t'') - r(t)) \bullet (s(t) - r(t)) \leqslant 0.$$

If we add together the last two inequalities and we recall that $s(t') = s(t)$, we get

$$(r(t) - s(t)) \bullet (r(t) - r(t'') + s(t'') - s(t')) \leqslant 0.$$

Because $r(t'') = s(t'')$ and $s(t') = s(t)$, we finally obtain

$$(r(t) - s(t)) \bullet (r(t) - s(t)) \leqslant 0,$$

which is equivalent to $\|r(t) - s(t)\| \leqslant 0$, implying that $r(t) = s(t)$, a contradiction. $\qquad \square$

We can now prove that Algorithm 2 works also with RIGID ASYNCH robots.

Theorem 5.1. *Algorithm 2 solves* Mutual Visibility *for* RIGID ASYNCH *robots with 3-colored lights.*

Proof. In the interior depletion phase there can be no collisions, due to Lemma 5.2. Also, whenever an internal robot moves, its destination lies on the boundary of the "real" $\mathcal{H}$ (cf. line 53 of Algorithm 2). Since movements are rigid, such a robot becomes external in a single move. Suppose for a contradiction that the interior depletion phase does not terminate. Then, at some point, the set of external robots reaches a maximum, all the external robots are set to *External*, and no robot is moving (indeed, moving internal robots are bound to become external). Hence, Lemma 4.4 can be applied. As a consequence, there is some internal robot that is able to move, and which will therefore reach the convex hull's perimeter at the end of its next *Move* phase, thus becoming external. This contradicts our assumptions. Therefore, in finite time all robots become external, and the interior depletion phase terminates.

When all robots are external, none of them moves unless it sees only robots set to *External* (line 27). This means that, in the vertex adjustments phase, a robot waits until it is sure that no robot is in the middle of a move (note that this holds also for robots that it cannot see, because as soon as one of them moves it becomes visible to all other robots). Indeed, a robot sets itself to *Adjusting* right before starting to move and sets itself back to *External* when it is done moving. Hence the robots synchronize themselves, and we may pretend them to be SSYNCH, as opposed to ASYNCH. Then, the proof proceeds exactly as in Lemma 4.8.

In the case in which the robots are initially collinear, the proof follows the lines of Theorem 4.1, with a few differences. Indeed, despite being ASYNCH, the robots manage to wait for each other and synchronize their actions. Suppose that one endpoint robot r becomes *Adjusting* and starts moving to its destination. Then, every robot is bound to wait for the other endpoint robot, s, to take action. So, s could either become *Adjusting* as well and start moving (if it performed its *Look* before r started moving), or it could notice r and become *External*. If r and s are both *Adjusting* and moving asynchronously, some other robots eventually become *External*, but do not move yet. In particular, referring to Figure 12, at least robots r' and s' can become external in this phase. Notice that, if r and s move asynchronously in opposite directions

44
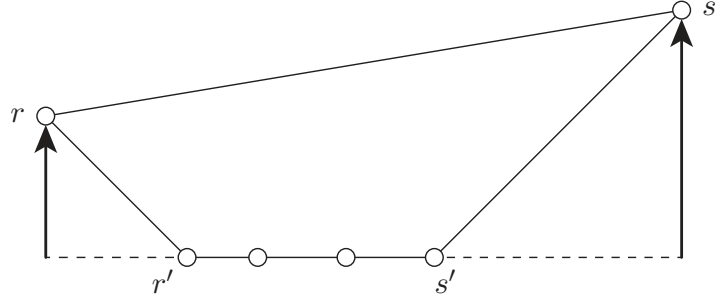
(a) r and s move in the same direction
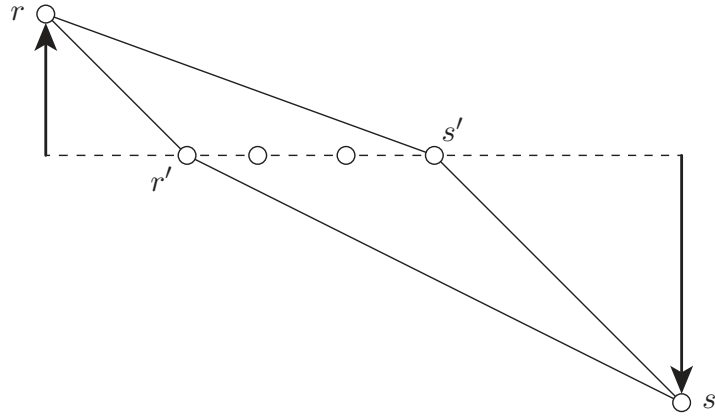
(b) r and s move in opposite directions

Figure 12: Evolutions of a collinear configuration in which both endpoints move

(Figure 12(b)), r' and s' may switch between being internal and being external several times. However, as soon as they set their light to *External*, they do not set it back to *Off* even if they become internal again. But r moves exactly by $r(0)r'(0)$, and s moves exactly by $s(0)s'(0)$ (line 12), because the model is RIGID. This movement length is chosen in such a way that both r' and s' eventually become vertex robots, as Figure 12(b) suggests. Therefore the colors of r' and s' are eventually consistent, despite asynchrony. So, every robot waits for both r and s to see some *External* robots and thus become *External* themselves. Only then do other robots start moving (lines 26–29). As a consequence, we may once again pretend that the robots in this phase are SSYNCH, and the proof is completed as in Theorem 4.1. □

5.2. Non-Rigid ASynch *Robots with Agreement on One Axis*

Unfortunately, for Non-Rigid ASynch robots, our correctness proof of the interior depletion phase of Algorithm 2 fails. Indeed, to prove collision avoidance, we used in a crucial way the fact that, if two internal robots are moving at the same time, then at most one of them saw the other robot in the middle of a movement. This is true under the Non-Rigid SSynch model (obviously) and under the Rigid ASynch model (because each internal robot becomes external after only one move), but not under the Non-Rigid ASynch model. In this model, an internal robot r may perform different moves in different directions before becoming external. For instance, if r's first movement is stopped by the adversary and, in the meantime, new robots have become *External* or new robots have become visible, r may decide to move in a significantly different direction the second time. This, paired with the ability of the ASynch scheduler to hold a moving robot for an indefinitely long time and then release it and let it complete its move, does cause collisions in some (quite pathological) cases. On the other hand, however, the vertex adjustments phase of Algorithm 2 works under all models; therefore we only need to replace the interior depletion phase and the segment breaking phase.

With the additional assumption that all robots agree on one axis, there is an easy way to fix the interior depletion phase, which is illustrated in Figure 13. Say that the robots agree on the y axis, i.e., they agree on the "North" direction, but they may disagree on "East" and "West". Then, if an internal robot sees that every robot that lies to the North is set to *External* (i.e., if its own y coordinate is maximum among the internal robots), it is eligible to move. If there is a row of several internal robots that are all eligible to move (as in Figure 13), then only the two endpoints are allowed to move, and the others wait. The left endpoint moves to the upper-left quadrant, and the right endpoint moves to the upper-right quadrant, and their destination points are on the convex hull, but not on locations already occupied by external robots. To guarantee termination, we make each robot move straight to the North toward the boundary of the convex hull of the visible robots, unless there are external robots in the way. In this special case, we make the robot move slightly sideways.

Also the protocol for the segment breaking phase needs some modifications: indeed, referring to Figure 12(b), in which r and s move in opposite directions, it is no longer true that r' and s' will eventually be external robots when r and s stop (recall that robots are Non-Rigid now). Unfortunately, r' and s' may become temporarily external while r and s move, and thus they
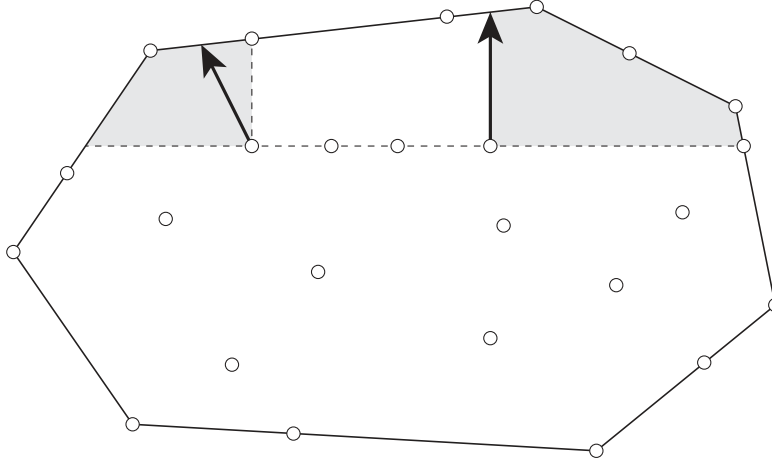
Figure 13: Interior depletion with agreement on one axis

may (permanently) set themselves to *External*, which could lead to inconsistent behaviors. Once again, we can fix the protocol if the robots agree on the y axis: now, in the segment breaking phase, an endpoint robot moves according to Algorithm 2 only if it has the maximum y coordinate. This makes only one endpoint move in most cases, which eliminates the aforementioned issue. Moreover, if both endpoints have the same y coordinate, they will both move North, thus forming a configuration like the one in Figure 12(a), which causes no trouble.

Theorem 5.2. *The* Mutual Visibility *problem is solvable by* Non-Rigid ASynch *robots carrying 3-colored lights, provided that they agree on one axis.*

Proof. We show that the above algorithm is correct. In the interior depletion phase, there can be no collisions, and each internal robot eventually reaches the convex hull. Indeed, suppose that initially there is a unique internal robot r with largest y coordinate. As soon as enough external robots have set themselves to *External*, r starts moving North, and no other robot moves. Eventually r becomes external without colliding with any robot (note that, even if r does not initially see the boundary of the convex hull, it will eventually see it after finitely many moves).

If several internal robots have the largest y coordinate, as in Figure 13, the argument is similar. At most two robots can move at the same time, and they cannot collide because the difference of their x coordinates cannot

decrease. After enough cycles, either they have reached the convex hull, or one of them has been "left behind" and is no longer eligible to move. Either way, at least one internal robot eventually becomes external.

Once an internal robot has become external, the same argument repeats for all other internal robots. Note that these "sub-phases" do not interfere with each other, because a new robot becomes eligible to move only after the previous eligible robots have stopped on the convex hull.

The moment the last internal robot becomes external, no robot is moving, and therefore the whole swarm correctly transitions to the vertex adjustments phase, which works exactly as described in Theorem 5.1 and Lemma 4.8.

If the robots are initially collinear, they correctly transition to a non-collinear configuration, as in Theorem 5.1. Indeed, note that the two end-point robots cannot move in opposite directions (as in Figure 12(b)), and hence it does not matter if they are RIGID or NON-RIGID, since in this case it is not harmful if they move by smaller amounts than those indicated by Algorithm 2 (cf. Figure 12(a)). The same clearly holds if $n = 3$ and the middle robot executes line 16. □

6. Related Problems and Alternative Models

Here we discuss some applications of the previous Mutual Visibility algorithms to other problems, and we also discuss different robot models.

6.1. Forming a Convex Configuration

As already observed, Algorithm 1 also solves the Convex Formation problem, where the robots have to terminate in a strictly convex position. Moreover, no robot ever crosses the perimeter of the initial convex hull unless, of course, all the robots are initially collinear. This works for RIGID SSYNCH robots carrying 2-colored lights.

For NON-RIGID SSYNCH robots carrying 3-colored lights, Algorithm 2 also solves the Convex Formation problem, but it has an additional property: during the interior depletion phase, the convex hull of the robots remains unaltered (unless all robots are collinear), and in the vertex adjustments phase it shrinks a little, due to the small movements of the vertices. We can actually make these movements as small as we want, by changing line 29 of Algorithm 2 into

$$\text{Move to } (a.position + b.position) \cdot \frac{\varepsilon}{\|a.position + b.position\|},$$

where ε is an arbitrarily-chosen positive constant. Similarly, in lines 12 and 16 we can make the robot move orthogonally to $\mathcal{H}$ by ε or less. As a result, we can guarantee that the robots will terminate in a (strictly convex) configuration whose vertices are contained in an ε-wide band around the initial convex hull's perimeter.

Similar observations hold for the algorithms and models discussed in Section 5.

6.2. Forming a Circle

As a followup to Algorithms 1 and 2, the robots can even solve the Circle Formation problem, in which they have to become concircular and then terminate. Moreover, if they are Rigid SSynch (respectively, Non-Rigid SSynch), they can do so with the same 2-colored (respectively, 3-colored) lights that they used to solve Mutual Visibility.

First, it is necessary to slightly modify the termination condition of the algorithms: in Algorithm 1, when a robot sees only robots set to *Vertex*, it does not terminate, but it starts executing a *circle formation* phase. Similarly, in Algorithm 2, we remove lines 23–25, thus preventing vertex robots from reverting their color to *External* and terminating after they have adjusted their position. Instead, they wait until they only see robots set to *Adjusting*. Accordingly, in lines 27 and 36 we remove the conditions that prevent robots from moving if they see other robots set to *Adjusting*. Since we are assuming that robots are SSynch, it is straightforward to see that the correctness proof of Section 4 goes through even after these modifications to the protocol, and that eventually all robots are set to *Adjusting*. At this point, the circle formation phase starts.

Since all robots are set to *Adjusting*, each robot knows that all of them occupy non-degenerate vertices of the convex hull, and that there are no other robots in the swarm. Hence the phase starts in a strictly convex configuration, and all the robots see each other. In particular, the *Smallest Enclosing Circle* (SEC) computed by each active robot is the same. In the circle formation phase, the robots move toward the perimeter of the SEC in a precise order, as illustrated in Figure 14. If a robot lies in p, which is not on the SEC, and one of its neighbors lies in s, which is on the SEC, then the robot in p moves along the extension of the edge of the convex hull that is incident to p and not to s. If both neighbors of the robot lie on the SEC (as with the robot in q in Figure 14), it chooses one of its two incident edges, and moves along the extension of that edge.
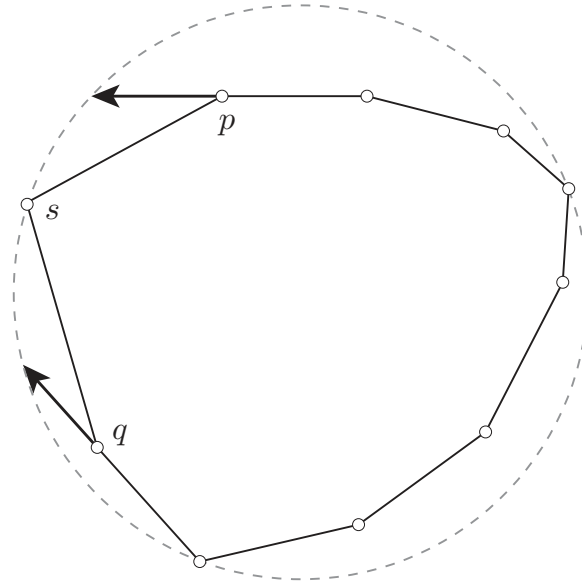
Figure 14: Forming a circle

It is clear that the combined motion of the robots does not cause collisions or obstructions, and that the SEC is always preserved. Indeed, any robot that is already on the SEC remains still, and those that are inside the SEC are allowed to move only within the SEC itself. Moreover, the direction in which each robot moves is preserved until one of them reaches the SEC. Hence, even if robots are NON-RIGID, after finitely many turns at least one of them reaches the SEC, and therefore eventually they all reach the SEC. At this point, they correctly terminate.

The same circle formation phase can also be used in conjunction with the algorithms discussed in Section 5 for ASYNCH robots. The only difference is that, instead of modifying the ASYNCH algorithms like we did with the SSYNCH ones, we simply add an extra state, called *Done*, to synchronize robots and make them transition correctly from the vertex adjustments phase into the circle formation phase. That is, instead of terminating, a robot sets itself to *Done*, and then waits until all other robots are set to *Done*, as well. Only then does it proceed to executing the circle formation phase described above. Of course, before the circle formation phase starts, if a robot sees another robot set to *Done*, it treats its like an *External* robot. This works

with both RIGID and NON-RIGID ASYNCH robots carrying 4-colored lights.

6.3. Converging to a Point Without Colliding

A simple modification of Algorithm 1 solves the Near-Gathering problem, which requires all the robots to converge to a point without colliding: it is sufficient to remove lines 8, 9, and 23, that is, all the operations involving colors, and the termination condition. Indeed, if there is only one internal robot, either it will become external, or the other robots will converge to its location. On the other hand, if all robots become external, the convex hull will keep shrinking until its vertices converge to a point. This works for RIGID SSYNCH robots, even without the use of colored lights.

However, if the robots carry 2-colored lights, they can also terminate when they get close enough to one another. This is done by simply modifying the termination condition of line 9:

if $\forall r, s \in \mathcal{V}$, $r.light = Vertex$ **and** $\|r.position - s.position\| < \varepsilon$ **then** Terminate

where ε is any given positive constant.

6.4. NON-RIGID SSYNCH Robots with Knowledge of δ

Suppose that the robots are NON-RIGID SSYNCH, and as such they can be stopped by the scheduler at each turn before they reach their destination point, but not before they have moved by at least δ. Recall that in this case they have an algorithm for Mutual Visibility that uses 3-colored lights, described in Section 4. However, if the robots know the exact value of δ and they can use it in their computations, they can solve Mutual Visibility even with 2-colored lights, by executing a slightly modified version of Algorithm 1.

If all the robots are initially collinear, Algorithm 1 makes them reach a non-collinear configuration, even if they are NON-RIGID. Subsequently, the invariants discussed in Section 3.2.1 keep holding, and in particular the convex hull of the robots never grows, and vertex robots never become non-vertex robots. We have to show that a version of Lemma 3.1 can be obtained for this model, as well. Referring to Figure 1, we can make the robot in p move toward $(a + b)/2$ by a smaller amount, never passing internal robots, and never colliding with them, unless they are closer than δ. If an internal robot r is closer than δ and it stands between p and $(a + b)/2$, the robot in p moves close enough to r, on the line parallel to ab, and it sets its light to the correct value (note that it knows before moving whether it will end up

being a vertex robot or not). This "lateral move" cannot be stopped by the scheduler, and it is guaranteed to create a new external robot, and eventually increase by one the number of vertex robots.

On the other hand, if only "non-lateral moves" are made, the analysis in Section 3.2.2 can be generalized, because Equation 1 takes the form

$$r_i(t+1) = \frac{\mu}{2} \cdot r_{i-1}(t) + \mu \cdot r_i(t) + \frac{\mu}{2} \cdot r_{i+1}(t),$$

where $\mu \in [\mu_0, 1/2]$, and μ_0 is a constant. Indeed, if the convex hull of the robots never grows, and its initial diameter is d, then each moving robot is guaranteed to move by at least a fraction of $\mu_0 = \delta/d$ of its computed movement vector. Therefore, all the lemmas in Section 3.2.2 can be reproved by merely adjusting some coefficients in the formulas.

It remains to prove that, if only one internal robot is left, it eventually reaches the boundary of the convex hull without colliding with other robots. But since δ is known, we can make this robot stay still until it either becomes external (due to other robots' movements), or the diameter of the convex hull becomes smaller than δ. As soon as it is guaranteed to make a reliable move, it can reach the midpoint of an edge of the convex hull, and therefore become external.

When all robots are external, they eventually reach a strictly convex configuration and they correctly terminate, as detailed in the proof of Theorem 3.1.

6.5. Trading Lights with the Knowledge of n

Suppose that the robots do not carry visible lights and have no internal memory, but they share the knowledge of the total number of robots in the swarm, n. If the robots are RIGID SSYNCH, it is possible to slightly modify Algorithm 1 to solve Mutual Visibility in this model, as well.

Note that the information given by other robots' visible lights is used only when a robot has to terminate (line 9), or when it is the only internal robot and it has to move to the perimeter of the convex hull (line 23). However, both these situations can be recognized locally by counting the vertices of the convex hull of the visible robots: if it has n non-degenerate vertices, Mutual Visibility has been solved, and the executing robot can terminate. If the convex hull has $n-1$ vertices and the executing robot is internal, it moves to the boundary, as in line 23 of Algorithm 1.

The same techniques can be used to modify the algorithm of Section 6.4, so that NON-RIGID SSYNCH robots with knowledge of δ and knowledge of n can solve Mutual Visibility without the use of colored lights.

We are also able to optimize Algorithm 2 for robots with knowledge of n: namely, we can achieve termination detection even if the robots do not use the *Adjusting* color, as follows. When all robots are external and a vertex robot makes a default move, it does not change its color, but remains *External*. Then, when a vertex robot sees n robots, it terminates. Note that making a default move allows a robot to see all other robots at its next activation, and therefore each external robot makes at most one default move before terminating. Moreover, when all robots are collinear, we apply this simple protocol: if a robot is an endpoint of the convex hull, it moves orthogonally to it (without changing color); otherwise it stays still. This way, as soon as an endpoint is activated, the configuration becomes non-collinear. The only exception to this rule is the case $n = 3$, in which the central robot has to move orthogonally to the segment, while the other two robots stay still.

This technique allows NON-RIGID SSYNCH robots with knowledge of n to solve Mutual Visibility with 2 colors as opposed to 3.

6.6. Fault Tolerance

Observe that Lemma 3.2 requires only $n-1$ valves to be opened infinitely often, as opposed to n. This implies that, if RIGID SSYNCH robots execute the modification of Algorithm 1 described in Section 6.3, they converge to a point even if one robot is unable to move. Therefore, in the presence of one faulty robot, Near-Gathering is still solvable, even without the use of colored lights. (On the other hand, if two robots are faulty, Near-Gathering is clearly unsolvable, because the two faulty robots cannot approach each other.) Additionally, if n is known, Mutual Visibility and Convex Formation are solved in the presence of a faulty robot, provided that it is located on the boundary of the convex hull.

6.7. Sequential Scheduler

Suppose that the scheduler is *sequential*, i.e., it is SSYNCH and it activates exactly one robot at each turn. In this very special model there is a simple algorithm to solve Mutual Visibility with no termination detection, even with no colored lights and no knowledge of n, and even if the robots are NON-RIGID and two of them are faulty. (If three robots are faulty, Mutual Visibility is clearly unsolvable.) When a robot is activated, it just moves by a small

amount, without crossing or landing on any line that passes through two robots that it can currently see (including itself), as illustrated in Figure 15. Clearly, when a robot moves as described, it becomes visible to all other robots. Hence, when all robots (except possibly two of them) have moved at least once, they can all see each other.
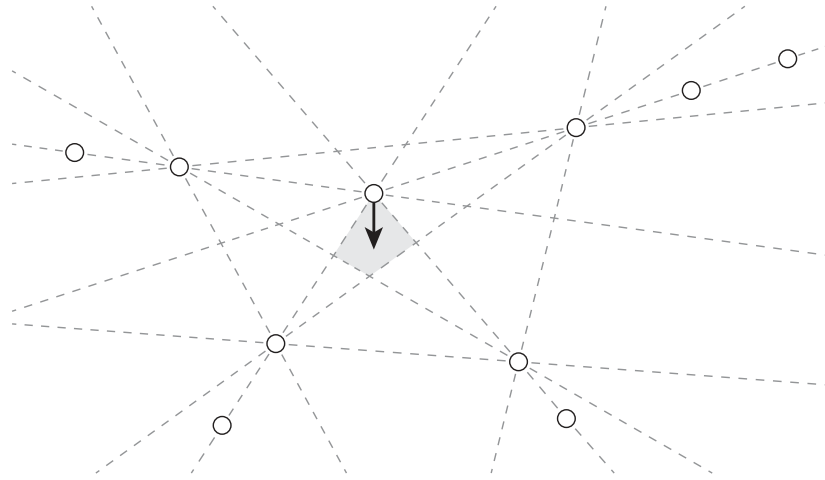

Figure 15: Solving Mutual Visibility under a sequential scheduler

This protocol solves Mutual Visibility with no termination detection, in the sense that, after finitely many turns, the robots will keep seeing each other. However they will never stop moving because they will never know if their task is terminated or not. Indeed, termination detection is not achievable under this set of assumptions and, to be able to obtain it, some other assumptions are needed; for example, 2-colored lights or the knowledge of n. With 2-colored lights, a robot can change its own color the first time it moves, and terminate at the next activation. With knowledge of n, a robot simply terminates when it sees n robots.

7. Concluding Remarks

In this paper we initiated the investigation of the computational capabilities of a swarm of anonymous mobile robots with obstructed visibility: in this model, which has never been considered in the literature, two robots cannot see each other if a third robot lies between them. We focused on the basic problem of Mutual Visibility, in which the robots, starting from an

arbitrary configuration, have to reach a configuration in which they all see each other, and then terminate the execution. This task is clearly impossible if the robots are completely oblivious, unable to communicate, and do not have any additional information. Indeed, in this scenario a robot can never distinguish between an initial configuration in which it cannot see some other robots, and a configuration in which all robots are visible and it is safe to terminate (recall that the termination operation cannot be undone).[1] Therefore we employed the extended model of luminous robots, in which each robot is carrying a visible light that it can set to different colors. The goal is then to minimize the number of colors required by the robots to solve the Mutual Visibility problem under different settings and restrictions. Namely, we considered SSynch and ASynch robots, and Rigid and Non-Rigid movements. We also discussed how to reduce the number of used colors if some information is given to the robots, such as the size of the swarm, n, or a minimum distance δ that a robot is guaranteed to cover in each movement. Our main results are summarized in Theorems 1.1 and 1.2. We then touched on more complex problems, and proposed solutions that use our Mutual Visibility protocols as a preprocessing step. Notably, we gave the first algorithms for the Near Gathering problem (with fault tolerance) and the Circle Formation problem that work under the obstructed visibility model.

We proposed two main algorithms, and several modifications and adaptations to various models. Algorithm 1 (*Shrink*) uses 2 colors and makes the convex hull of the robots shrink, ideally converging to a point. Algorithm 2 (*Contain*) uses 3 colors, and keeps the initial convex hull basically unaltered. It is therefore suited for practical situations in which the robots have to surround a large-enough area, as well as solving Mutual Visibility. Also, both algorithms keep the robots within the initial convex hull (unless they are initially collinear), which is useful in practice, for instance in the presence of hazardous areas around the swarm.

Some interesting research problems remain unsolved. We would like to reduce the number of colors used by our algorithms in the various models, or to prove our algorithms optimal. Our solutions to Mutual Visibility in some models use only 2 colors (or no lights at all if n is known), which is

---

[1]It is worth noting that, if robots are only required to remain still forever after they have all become mutually visible (as opposed to terminating their execution), then this argument is no longer valid. With such a notion of *weak termination*, there could exist an algorithm for Mutual Visibility that uses no colored lights and no extra information.

clearly optimal. For other models, such as NON-RIGID SSYNCH and RIGID ASYNCH, we used 3 colors, and our question is whether this can be improved. We conjecture that Algorithm 1, which uses only 2 colors and has been designed and proven correct for RIGID SSYNCH robots, can be applied with no changes also to NON-RIGID SSYNCH robots (we could prove that 2 colors are sufficient in this model under the assumption that the robots know δ). In the NON-RIGID ASYNCH setting we were only able to solve Mutual Visibility (with 3 colors) assuming that the robots agree on the direction of one coordinate axis. We ask if this assumption can be dropped, perhaps if more colors are used. Another question is whether Mutual Visibility can be solved deterministically without using colored lights or extra information, and without termination detection. That is, we allow the robots to move forever, but we require them to remain mutually visible from a certain time on. We proposed a simple solution that works under the sequential scheduler, and we ask if this can be generalized to SSYNCH or even ASYNCH schedulers.

We emphasize that obstructed visibility represents a broad line of research in the field of computation by mobile robots, and this paper explored just a few directions. Several classical problems are worth studying under this model, such as the general Pattern Formation problem, Flocking, Scattering, with or without bounded visibility, etc.

References.

[1] C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 250–259, 2013.

[2] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006.

[3] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice Hall, 1989.

[4] Z. Bouzid, S. Dolev, M. Potop-Butucaru, and S. Tixeuil. Robocast: Asynchronous communication in robot networks. In *Proceedings of the 14th Inter-*

national Conference on Principles of Distributed Systems (OPODIS), LNCS 6490, pages 16–31, 2010.

[5] K. Bolla, T. Kovacs, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *Proceedings of the International Symposium on Swarm and Evolutionary Computing*, pages 30–38, 2012.

[6] I. Chatzigiannakis, M. Markou, and S. Nikoletseas. Distributed circle formation for anonymous oblivious robots. In *Proceedings of the 3rd International Workshop on Experimental and Efficient Algorithms (WEA)*, pages 159–174, 2004.

[7] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing for mobile robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012.

[8] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithms in asynchronous robots systems. *SIAM Journal on Computing*, 34:1516–1528, 2005.

[9] R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399(1,2):71–82, 2008.

[10] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[11] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6,7):48–499, 2009.

[12] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In *Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 506–515, 2012.

[13] S. Das, P. Flocchini, G. Prencipe, and N. Santoro. Synchronized dancing of oblivious chameleons. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN)*, pages 113–124, 2014.

[14] S. Das, P. Flocchini, N. Santoro, and M. Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28(2):131–145, 2015.

[15] S. Datta, A. Dutta, S. Gan Chaudhuri, and K. Mukhopadhyaya. Circle formation by asynchronous fat robots. In *Proceedings of the 9th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pages 195–207, 2013.

[16] X. Défago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proceedings of the 2nd ACM International Workshop on Principles of Mobile Computing (POMC)*, pages 97–104, 2002.

[17] X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1,3):97–112, 2008.

[18] Y. Dieudonné, O. Labbani-Igbida, and F. Petit. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4):16:1–16:20, 2008.

[19] Y. Dieudonné and F. Petit. Swing words to make circle formation quiescent. 14th *Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 166–179, 2007.

[20] Y. Dieudonné and F. Petit. Squaring the circle with weak mobile robots. In *Proceedings of* 19th *Int. Symposium on Algorithms and Computation (ISAAC)*, 354–365, 2008.

[21] A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In *Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 70–87, 2007.

[22] P. Flocchini, G. Prencipe, and N. Santoro. Self-deployment algorithms for mobile sensors on a ring. *Theoretical Computer Science*, 402(1):67–80, 2008.

[23] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2012.

[24] P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta. Distributed computing by mobile robots: Solving the uniform circle formation problem. In *Proceedings of the 18th International Conference on Principles of Distributed Systems (OPODIS)*, pages 217–232, 2014.

[25] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science*, 407(1–3):412–447, 2008.

[26] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In *Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 189–200, 2013.

[27] N. Fujinaga, Y. Yamauchi, S. Kijima, and M. Yamashita. Asynchronous pattern formation by anonymous oblivious mobile robots. 26th *Int. Symposium on Distributed Computing (DISC)*, 312–325, 2012.

[28] V. Gervasi and G. Prencipe. Coordination without communication: The case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324–344, 2004.

[29] T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Defago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012.

[30] B. Katreniak. Biangular circle formation by asynchronous mobile robots. In *Proceedings of the 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 185–199, 2005.

[31] T. Miyamae, S. Ichikawa, and F. Hara. Emergent approach to circle formation by multiple autonomous modular robots. *J. Robotics and Mechatr.*, 21(1):3–11, 2009.

[32] L. Pagli, G. Prencipe, and G. Viglietta. Getting close without touching. In *the International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 315–326, 2012.

[33] D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *Proceedings of the 7th International Workshop on Distributed Computing (IWDC)*, pages 1–12, 2005.

[34] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems* 13(3):127–139, 1996.

[35] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[36] G. Viglietta. Rendezvous of two robots with visible bits. In *Proceedings of the Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 291–306, 2013.

[37] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26–28):2433–2453, 2010.