Gathering of Asynchronous Robots with Limited Visibility*

Paola Flocchini† Giuseppe Prencipe‡ Nicola Santoro§
Peter Widmayer¶

Abstract

In this paper we study the problem of *gathering* in the same location of the plane a collection of identical oblivious mobile robots. Previous investigations have focused mostly on the unlimited visibility setting, where each robot can always see all the other ones, regardless of their distance.

In the more difficult and realistic setting where the robots have *limited visibility*, the existing algorithmic results are only for convergence (towards a common point, without ever reaching it) and only for synchronous environments, where robots' movements are assumed to be performed instantaneously.

In contrast, we study this problem in a totally *asynchronous* setting, where robots' actions, computations, and movements require a finite but otherwise unpredictable amount of time. We present a protocol that allows anonymous oblivious robots with limited visibility to gather in the same location in *finite time*, provided they have orientation (i.e., agreement on a coordinate system).

Our result indicates that, with respect to gathering, orientation is at least as powerful as instantaneous movements.

Keywords: Mobile Robots, Distributed Computing, Asynchrony, Point Formation, RendezVous, Orientation, Cooperation and Control.

*A preliminary version of this paper has been presented at the 18^{th} Symposium on Theoretical Aspects of Computer Science [17].

†SITE, University of Ottawa, flocchin@site.uottawa.ca

‡Dipartimento di Informatica, Università di Pisa, prencipe@di.unipi.it

§School of Computer Science, Carleton University, santoro@scs.carleton.ca

¶Theoretische Informatik, ETH Zürich, widmayer@inf.ethz.ch

1 Introduction

In current robotics research, both from engineering and behavioral viewpoints, the trend has been to move away from the design and deployment of few problem-specific robots which are powerful enough to solve the problem at hand; the main reason is that such robots are clearly complex, difficult to program and to construct, and, thus, usually expensive.

The interest has shifted instead towards the design and use of a large number of generic robots which have minimal physical capabilities, exceedingly simple behavior but are easy (and, thus, relatively inexpensive) to design, construct and deploy (e.g., see [4, 5, 6, 7, 14, 15, 18, 19, 20, 21]). In particular, these robots are only capable of sensing their immediate surrounding, performing simple computations on the sensed data, and moving towards the computed destination; their behavior is a simple cycle of sensing, computing, moving and being inactive. In spite of their limitations, the robots should be able, together, of performing rather complex tasks.

A basic set of theoretical questions refer to determining *minimal robot capabilities*; that is, how "simple" the robots can be to perform the required task. In computational terms, this question is to identify the factors which influence solvability of a given problem (the task) by a set of mobile robots. The research is still focusing on basic tasks such as gathering, flocking, pattern formation, scattering, etc. [1, 4, 8, 9, 10, 12, 13, 16, 17, 22, 23, 24, 25].

In this paper we are interested in *gathering*: the basic task of having the robots meet in a single location (the choice of the location is not predetermined). Since the robots are modeled as points in the plane, the task of robots gathering is also called *point formation*. A related task is that of *convergence*, in which the robots must converge towards a single point without necessarily reaching it.

Both gathering and convergence have been extensively investigated experimentally and theoretically in the *unlimited visibility* setting, that is assuming that the robots are capable to sense ("see") the entire space (e.g., see [1, 8, 9, 10, 11, 12, 16, 18, 22, 23, 25]).

In general, and more realistically, robots can sense only a surrounding within a radius of bounded size (refer to the example depicted in Figure 1). This setting, called the *limited visibility* case, is understandably more difficult; for example, a robot might not even know the total number of robots nor where they are located if outside its radius of visibility.

Not surprisingly, only few algorithmic results are known [2, 3]. In particular, Ando et al. [2] presented a procedure which allows indistinguishable robots, placed on a plane without any common coordinate system, to *converge* towards the same point. Their procedure does not require the robots to remember observations nor computations performed in the previous steps. Their result implies that convergence can be achieved by robots with limited visibility which are indeed very simple: anonymous, oblivious and disoriented. This powerful result is however based on a very strong "atemporal" assumption: the robots must be capable in each cycle to perform all the computing and moving *instantaneously*; moreover, cycles are performed at specific instants of time (i.e., they are synchronized) although not all robots might be active at those times. This model is called *semi-synchronous*. The properies of this model have many crucial consequences; in particular, if movement is instantaneous, a robot will not be seen by the others while moving; if

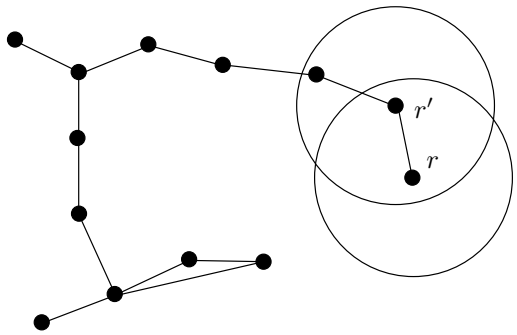


Figure 1: The limited visibility setting: a robot can only see robots that are within its radius of visibility.

computing is instantaneous, a robot always decides based on the correct current situation in its neighborhood. This assumption clearly limits the practical impact of the results.

There are some immediate and natural questions arising from observing the nature of this result. In particular:

- Under what conditions anonymous oblivious robots with limited visibility can *gather* in finite time (and not just converge without ever reaching the point)?

- Under what conditions can they do so if movement is *not instantaneous*?

In other words, the quest is for a gathering algorithm for the most general *asynchronous* setting, when both computations and movements require a (finite but otherwise) unpredictable amount of time. Notice that, if movement is not instantaneous, a robot can be seen by the others while moving, and that position mistaken for a destination point of a move; if computing is not instantaneous, a robot might make a decision based on a view of its neighborhood that is totally outdated.

In this paper, we provide an effective answer to both questions. In fact, we prove that the availability of *orientation*[1] enables a set of anonymous oblivious robots with limited visibility to gather in finite time even if they are fully asynchronous. This result holds not only allowing each activity and inactivity of the robots to be totally unpredictable (but finite) in duration, but also making their movement towards a destination unpredictable (but not infinitesimally small) in length.

This shows that gathering can be performed in a finite number of moves by simpler robots with fewer restrictions on the problem than known before, provided they have a common orientation. From a computational point of view, our result implies that with respect to the gathering problem, orientation is at least as powerful as instantaneous action.

Let us remark that the proposed algorithm does *not* assume that the robots have the capability of *multiplicity detection* (i.e., the ability to determine in the sensing phase if more than one robot is in a given location).

[1] i.e., agreement on axes and directions (positive vs. negative) of a common coordinate system, but not necessarily on the origin nor on the unit distance.

The paper is organized as follows. In Section 2 the terminology and notations used in the paper and some useful geometric lemmas are introduced. The gathering algorithm is described in Section 3, and in Section 4 its correctness is proven. Finally, in Section 5, some conclusions are drawn and ideas for future work are presented.

2 Terminology and Definitions

2.1 The Robots

We consider a system of autonomous mobile robots. Each robot is capable of sensing its immediate surrounding, performing computations on the sensed data, and moving towards the computed destination; its behavior is an (endless) cycle of being inactive, sensing, computing, and moving.

The robots are viewed as points, and modeled as units with computational capabilities, which are able to freely move in the plane. Each robot has its own local coordinate system, defined by two linearly independent vectors in the plane (as for the usual Cartesian coordinate system), one called the direction of the "x-axis", the other of the "y-axis". The vectors are the same for all robots; we say the robots "agree on the axes and their direction", and we call this their "orientation". The unit length along each axis is chosen independently by each robot; we say the robots "do not agree on the unit distance". Furthermore, the origin of the coordinate system need not be the same for all robots; we say the robots "do not agree on the location of the origin". They are equipped with sensors that let each robot observe the positions of the others. The robots are *oblivious*, meaning that they do not remember any previous observation nor computations performed in the previous steps.

The robots are *anonymous*, meaning that they are a priori indistinguishable by their appearances, and they do not have any kind of identifiers that can be used during the computation. Moreover, there are no explicit direct means of communication.

The robots are *fully asynchronous*: there is no common notion of time, and the amount of time spent in observation[2], in computation, in movement, and in inactivity is finite but otherwise unpredictable.

The robots have *limited visibility*; that is, each robot can observe only what is at most at a fixed distance V from it.

Summarizing, the robots are asynchronous, oblivious, anonymous, and with limited visibility; they do however have orientation.

2.2 The Cycle of Activity

The robots execute the same deterministic algorithm, which takes as input the observed positions of the robots within the visibility radius, and returns a destination point towards which the executing robot moves.

A robot is initially in a *waiting* state (*Wait*); asynchronously and independently from the other robots, it *observes* the environment in its area of visibility (*Look*); it *calculates*

[2]i.e., activating the sensors and receiving their data.

its destination point based only on the observed locations of the robots (*Compute*); it then *moves* towards that point (*Move*); after the move it goes back to a waiting state.

The sequence: *Wait - Look - Compute - Move* will be called a *cycle of activity* (or briefly *cycle*) of a robot.

The operations performed by the robots in each state will be now described in more details.

1. **Wait** The robot is idle. A robot cannot stay idle forever (see Assumption A2 below).

2. **Look** The robot observes the world by activating its sensors which will return a snapshot of the positions of all other robots with respect to its local coordinate system. (Since robots are viewed as points, their positions in the plane is just the set of their coordinates; furthermore, they do not obstruct visibility among other points).

   Notice that a robot might not in general be able to detect *multiplicity*, i.e. whether there is more than one robot on any of the observed points, including the position where the observing robot is.

3. **Compute** The robot performs a *local computation* according to its deterministic, oblivious algorithm. The result of the computation is a destination point (which could be the current location).

4. **Move** The robot moves towards the computed destination. If the destination is the current location, the robot is said to perform a *null movement*; otherwise, it is said to execute a *real movement*. The movement is *uninterrupted*, but the speed is not necessarily uniform. Provided the distance travelled is neither infinite nor infinitesimally small (see Assumption A1 below), the move can end anywhere before the destination[3].

In the rest of the paper we shall partition the robots into sets depending on their state at a given time.

$\mathbb{W}(t)$ and $\mathbb{L}(t)$ are the sets of all the robots that are respectively in state *Wait* and *Look* at time t.

$\mathbb{C}(t)$ is the set of all the robots that at time t are in state *Compute*; the subset $\mathbb{C}_\emptyset$ contains those robots whose computation's result is to execute a *null movement*.

$\mathbb{M}(t)$ is the set of all the robots that at time t are executing a movement; the subset $\mathbb{M}_\emptyset(t)$ contains the robots executing a *null movement* (they stay still).

Regarding the *Look* state, without loss of generality, in the following we will assume that it consists only of the time instant where the snapshot is actually done. In fact, any time the robot needs to activate its sensors (before the snapshot is taken) and to process the information retrieved with the snapshot (that will be used in the next *Compute* state, after the snapshot is taken), can be charged to the *Wait* and to the *Compute* state, respectively.

[3] e.g. because of limits to the robot's motorial capability.

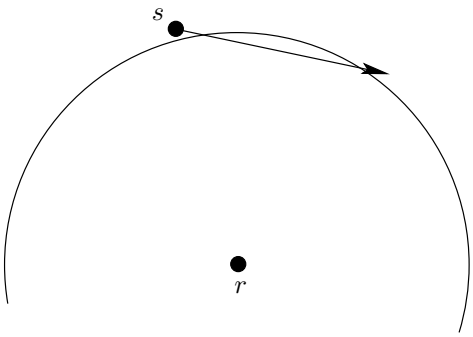


Figure 2: When s starts moving (the left end of the arrow), r and s do not see each other. While s is moving, r *Look*s and sees s; however, s is still unaware of r. After s passes the area of visibility of r, it is still unaware of r.

In the model, there are only two limiting assumptions about time and space. The first refers to space; namely, the distance traveled by a robot during a cycle of activity.

Assumption A1 (Distance) *The distance traveled by a robot r in a Move is not infinite. Furthermore, it is not infinitesimally small: there exists a constant $\delta_r > 0$, such that, if the destination point is closer than δ_r, r will reach it; otherwise, r will move towards it by at least δ_r.*

Note that, without this assumption, it would be impossible for any algorithm to terminate in finite time. As no other assumptions on space are made, the distance traveled by a robot in a cycle is unpredictable. In the following, we shall use $\delta = \min_r \delta_r$.

The second assumption in the model refers to the length of a cycle of activity.

Assumption A2 (Cycle of Activity) *The amount of time required by a robot r to complete a cycle of activity is not infinite. Furthermore, it is not infinitesimally small: there exists a constant $\varepsilon_r > 0$ such that the cycle will require at least ε_r time.*

The purpose of this assumption is to ensure that gathering does not become impossible because some robot takes an infinite time to complete one of its cycles. As no other assumption on time exists, the resulting system is truly *asynchronous* and the duration of each activity (or inactivity) is unpredictable. As a result, robots can be seen while moving, and computations can be made based on obsolete observations. For example (see Figure 2), robot s in transit towards its destination, is seen by r; however, s is not aware of r's existence and, if it starts the next cycle before r starts moving, s will continue to be unaware of r.

2.3 Visibility and Geometric Properties

Given a robot r, let $First(r,t) = \min\{t' \geq t | r \in \mathbb{L}(t')\}$ be the first time, from time t on, at which r performs a *Look* operation; and let $Last(r,t) = \max\{t' \leq t | r \in \mathbb{L}(t')\}$ be the last time, from the start up to time t, at which r has performed a *Look* operation.

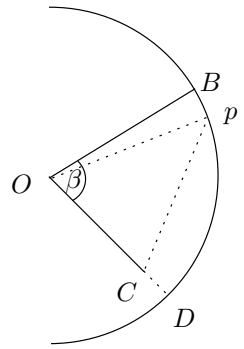


Figure 3: Lemma 2.3.

The robots are (viewed as) points in the plane. Let $r(t)$ denote the position of robot r at time t; when no ambiguity arises, we shall omit the temporal indication. The surrounding circle $Circle(r,t)$ of r at time t is the circle of radius V centered in $r(t)$. The *circle of visibility* of r at time t, $\mathcal{C}_t(r) = Circle(r, Last(r,t))$ is the surrounding circle of r in its most recent *Look*.

The result of an observation by a robot (in the *Look* state) will be the positions[4] of the robots in its circle of visibility at that time. Since a robot might not be able to detect multiplicity, it follows that it might not be aware of the real number of robots that populate its area of visibility at the time of the *Look*; in fact, a position can be occupied by more than one robot, but the observing robot will see all these robots as just one.

We now introduce some notations and geometrical lemmas which will be needed later. Let A and B be two points; with $\overline{AB}$ we will indicate the segment starting in A and terminating in B. When no ambiguity arises, we will also use the notation $\overline{AB}$ to denote the length of such a segment. Let A and B be two non-opposite points on a circle; with $arc(AB)$ we indicate the smaller of the two arcs on the circle between A and B; moreover, $seg(AB)$ indicates the region limited by the line segment $\overline{AB}$ and $arc(AB)$. Finally, given three distinct points A, B, and C, we denote $\triangle(A,B,C)$ the triangle having them as corners, and by $A\widehat{B}C$ the corresponding angle in B.

Capital Greek letters will represent vertical axes; capital calligraphic letters (e.g. $\mathcal{L}$, $\mathcal{R}$) indicate regions. Given a region $\mathcal{X}$, we denote by $|\mathcal{X}(t)|$ the number of robots in that region at time t; when no ambiguity arises, we will omit the temporal indication.

The following are elementary useful geometric properties.

Lemma 2.1. *Every internal chord of a triangle has length less or equal to the longest side of the triangle.*

Lemma 2.2. *Let Q be a convex quadrilateral. If all the sides and the two internal diagonals have length less or equal to V then every internal chord of Q has length less or equal to V.*

Another useful property is the following (see Figure 3).

[4] in its local coordinate system.

Lemma 2.3. *Let $\overline{OB}$ be the radius of a circle centered in O and let $B\widehat{O}D = \beta$, with $0 \leq \beta \leq 90°$, for D on the circle. Then $\overline{pC} \leq \overline{BC}$, $\forall p \in arc(BD)$ and $\forall C \in \overline{OD}$.*

Proof. The perpendicular bisector of the segment $\overline{Bp}$ passes through O, and all of $\overline{OD}$ lies on p's side of the bisector, and the lemma follows. □

3 The Algorithm

Let *Left* and *Right* be the leftmost and rightmost vertical axis, respectively, where some robot initially lie, and let us call *Universe* ($\mathfrak{U}$) the portion of the plane delimited by *Left* and *Right*.

The idea of the algorithm is to make the robots move towards *Right*, in such a way that, after a finite number of steps, they will reach it and gather at the bottommost position occupied by a robot at that time.

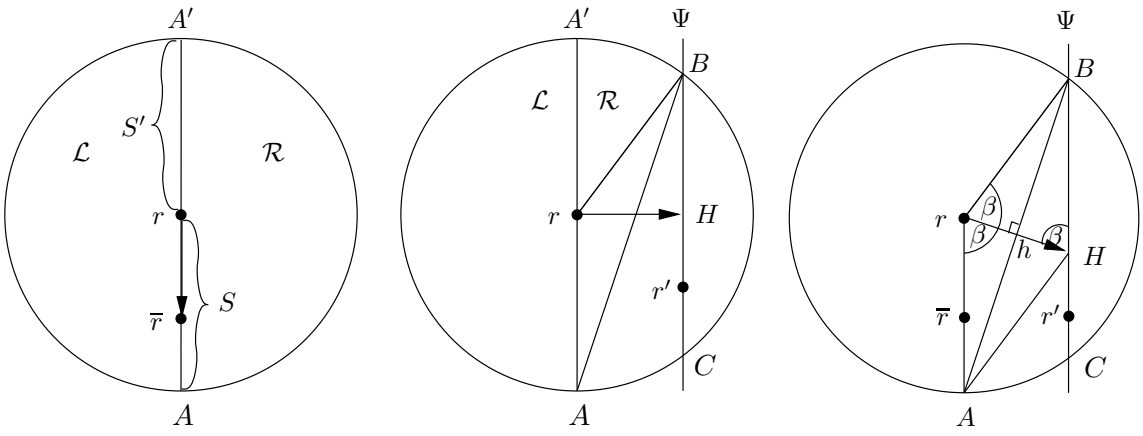
Let r perform a *Look* operation at time t; as a result, it has available its circle of visibility $\mathcal{C}_t(r)$ with the positions of all the robots in it at time t. The algorithm describes the computation that r will now do with this input. Different destination points will be computed depending on the positions of the robots in its circle of visibility; once the computation is completed, r starts moving towards its destination (but it may stop before the destination is reached). Informally,

- If r sees robots to its left or above on its vertical axis, it does not move.
- If r sees robots only below on its vertical axis, it moves down towards the nearest robot.
- If r sees robots only to its right, it moves horizontally towards the vertical axis of the nearest robot.
- If r sees robots both below on its axis and on its right, it computes a destination point and performs a diagonal move to the right and down, as explained later.

Let us now describe the algorithm in details (refer to Figure 4). Let $\overline{AA'}$ be the vertical diameter of $\mathcal{C}_t(r)$ with A' the top and A the bottom end point; let $\mathcal{R}_r$ and $\mathcal{L}_r$ denote the two topologically open regions[5] inside $\mathcal{C}_t(r)$ and to the right and to the left of r, respectively (when no ambiguity arises, we will use $\mathcal{R}$ and $\mathcal{L}$, respectively); and let $S = \overline{rA}$ and $S' = \overline{rA'}$, where both S' and S are topologically open on the r side (i.e., r belongs neither to S' nor to S). Let Ψ be the vertical axis of the robot in $\mathcal{R}_r$, if any, nearest to r with respect to its projection on the horizontal axis; and let B and C be the upper and the lower intersection, respectively, between $\mathcal{C}_t(r)$ and Ψ.

The algorithm for gathering in a point, described for robot r, works as follows.

[5] with respect to $\overline{AA'}$. The circle boundary belongs to $\mathcal{R}_r$ and $\mathcal{L}_r$.



a. b. c.

Figure 4: Notation used in the Gathering Algorithm: (a) vertical move, (b) horizontal move, and (c) diagonal move.

Gathering Algorithm

$Extreme := (|\mathcal{L}| = 0 \wedge |S'| = 0)$;
If $\neg Extreme$ **Then**
  `Do_nothing();`
Else
  Case $(|\mathcal{R}|, |S|)$
    •$(0, 0)$:
      `Do_nothing();`
    •$(0, \neq 0)$: *Vertical Move*
      $\overline{r}$:= nearest visible robot on S;
      `Move(`$\overline{r}$`).`
    •$(\neq 0, 0)$: *Horizontal Move*
      Ψ := `Nearest();`
      H := `Horizontal_Destination(`Ψ`);`
      `Move(`H`).`
    •$(\neq 0, \neq 0)$: *Diagonal Move*
      Ψ := `Nearest();`
      `Diagonal_Movement(`Ψ`).`

where the functions `Nearest()`, `Horizontal_Destination(`Ψ`)`, and `Move(p)` are as follows.

`Nearest()` returns the vertical axis Ψ of the robot in $\mathcal{R}$ nearest to r with respect to its projection on the horizontal axis (robot r' in Figure 4.b). `Horizontal_Destination(`Ψ`)` returns the intersection between Ψ and the horizontal line passing through r (see Figure 4.b). `Move(`p`)` terminates the local computation of the calling robot and moves it towards p.

In the last case of the Algorithm, r sees some robots below it and some to its right (robots $\overline{r}$ and r' in Figure 4.c); to avoid losing some robots, r moves diagonally, according

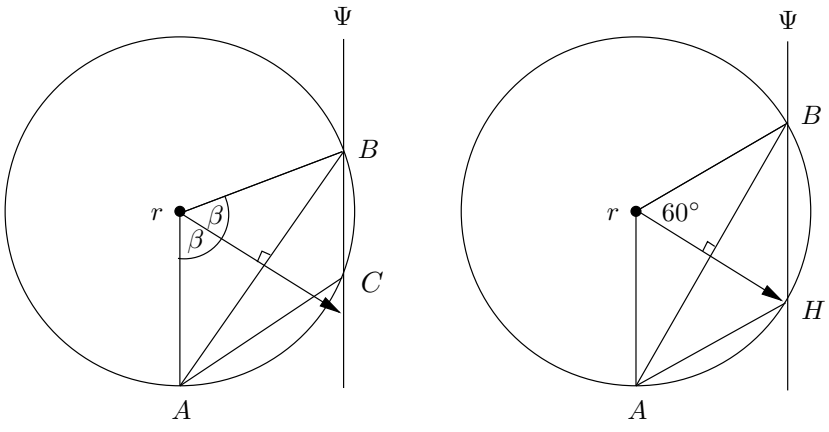


Figure 5: Routine `Rotate()`: in (a), $\beta < 60°$; in (b) the scenario after `Rotate()` has been invoked.

to the following routine.

```
Diagonal_Movement(Ψ)
  B := upper intersection between C_t(r) and Ψ;
  C := lower intersection between C_t(r) and Ψ;
  A := point on S at distance V from r;
  2β = ArB;
  If β < 60° Then
    (B, Ψ) := Rotate(r, B);
  H := Diagonal_Destination(V, Ψ, A, B);
  Move(H).
```

where `Rotate`(r, B) and `Diagonal_Destination`(V, Ψ, A, B) are as follows (see Figure 4.c).

`Rotate`(r, B) rotates the segment $\overline{rB}$ in such a way that $\beta = 60°$ and returns the new position of B and Ψ. This choice of angle ensures that the destination point is not outside the circle, that is $\overline{rH} \leq V$ (see Figure 5).

`Diagonal_Destination`(V, Ψ, A, B) computes the destination of r in the following way: the direction of r's movement is given by the perpendicular to the segment $\overline{AB}$; the destination of r is the point H on the intersection of the direction of its movement and of the axis Ψ.

4 Correctness

In this section, we prove the correctness of the algorithm by first showing that the robots which are initially visible will stay visible until the end of the computation, and then proving that at the end of the computation, all robots will gather in one point.

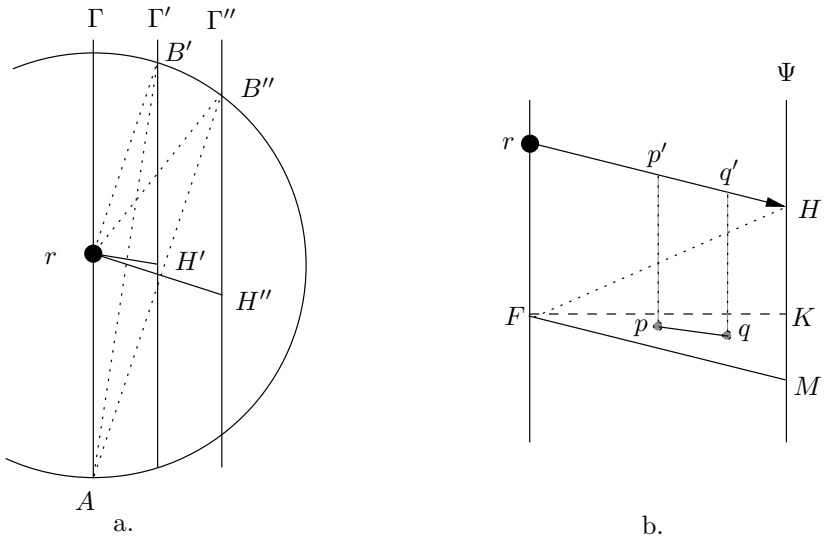


Figure 6: (a) Lemma 4.3; (b) Lemma 4.4.

For simplicity, in the following we use "to the right" etc. to denote the projection onto a single coordinate axis; i.e., the other coordinates are ignored.

4.1 Preliminary Observations

We first make some preliminary observations on the workings of the Algorithm. Let $r \in \mathbb{L}(t)$ be on vertical axis Γ at time t. Let β be the smallest of the two angles between the vertical axis of r and the direction of its diagonal movement (e.g., $A\widehat{r}H$ in Figure 4.c). By construction, $\beta \geq 60^\circ$ (this is guaranteed by routine `Rotate()`). Furthermore, $\triangle(A, r, B)$ is isosceles; hence, we get:

Lemma 4.1. *The length of line segment $\overline{rH}$ is always smaller or equal to V.*

Lemma 4.2. $\overline{BH} = \overline{AH} = V$, *and,* $\forall\ p \in \overline{rA}$ *and* $q \in \overline{rH}$, $\overline{pq} \leq V$. *Moreover,* $\diamond(A, r, B, H)$ *is a parallelogram.*

Let Γ' and Γ'' be two vertical axes to the right of Γ. We will denote by $\overline{\Gamma\Gamma'}$ and $\overline{\Gamma\Gamma''}$ the distances between the corresponding axes. Then we have:

Lemma 4.3. *Let $\beta_{\Gamma'}$ and $\beta_{\Gamma''}$ be the angles computed by routines* `Diagonal_Movement(`Γ'`)` *and* `Diagonal_ Movement(`Γ''`)`*, respectively. We have that: if $\beta_{\Gamma'} > \beta_{\Gamma''}$ then $\overline{\Gamma\Gamma'} < \overline{\Gamma\Gamma''}$; if $\overline{\Gamma\Gamma'} < \overline{\Gamma\Gamma''}$ then $\beta_{\Gamma'} \geq \beta_{\Gamma''}$.*

Proof. Let A be the point on Γ below r at distance V (refer to Figure 6.a). If $\beta_{\Gamma'} > \beta_{\Gamma''}$, clearly $\overline{\Gamma\Gamma'} < \overline{\Gamma\Gamma''}$. On the other hand, if $\overline{\Gamma\Gamma'} < \overline{\Gamma\Gamma''}$, then the upper intersection B' between $\mathcal{C}_t(r)$ and Γ' is higher than the upper intersection B'' between $\mathcal{C}_t(r)$ and Γ'' (and B' is to the left of B''). Thus, if $A\widehat{r}B'$ and $A\widehat{r}B''$ are both smaller than or equal to 120° the routine `Diagonal_Movement(`Γ'`)` returns 60° for both $\beta_{\Gamma'}$ and $\beta_{\Gamma''}$ (thus, $\beta_{\Gamma'} = \beta_{\Gamma''}$), otherwise clearly $A\widehat{r}B' > A\widehat{r}B''$ and, thus, $\beta_{\Gamma'} > \beta_{\Gamma''}$. □

Let $F \neq r(t)$ be a point on Γ below r at distance at most V from r, H be the destination point of the planned diagonal move of r, and Ψ the vertical axis where H is. Moreover, let K be the horizontal projection of F on Ψ, and M be the point on Ψ below H at distance $\overline{rF}$ from H (the situation is depicted in Figure 6.b).

Lemma 4.4. *Let $\overline{pq}$ be a segment in $\triangle(F, M, K)$, with q to the right of p, and q' the vertical projection of q over $\overline{rH}$. Then, $\forall\ l\ \in \overline{pq},\ \ l' \in \overline{q'H}$ we have that $\overline{ll'} \leq V$.*

Proof. By Lemma 2.1 on $\triangle(F, H, M)$, $\overline{pH} \leq V$ (recall that $\overline{FM} = \overline{rH} \leq V$ by Lemma 4.1). Furthermore, by Lemma 2.1 on $\triangle(p', p, H)$, $\overline{pq'} \leq V$, with p' the vertical projection of p over $\overline{rH}$. Since, by construction, $\overline{qq'} \leq V$, the lemma follows by applying Lemma 2.2 on $\diamond(q', H, q, p)$. □

Two other preliminary observations that follow directly from the Algorithm are:

Lemma 4.5. *Let robots r and s be within distance V at time t, and let $r \in \mathbb{L}(t)$. If s is to the right of r at time t, then r can not pass the vertical axis of s in one cycle.*

Lemma 4.6. *Let robots r and s be within distance V at time t, and let $r \in \mathbb{L}(t)$. If they are on the same axis Γ with s below r at time t, then r can not move on Γ below $s(t)$ in one cycle.*

4.2 Preserved Connectivity

The initial *distance graph* $D(0) = (N, E(0))$ of the robots is the graph whose node set N is the set of the input robots and, $\forall r, s \in N$, $(r, s) \in E(0)$ iff r and s are initially at distance no more than the visibility radius V. First of all notice that $D(0)$ must be connected for the gathering problem to be solvable:

Lemma 4.7. *If the distance graph $D(0)$ is disconnected, the gathering problem is unsolvable.*

Proof. By contradiction, assume that there exists a deterministic algorithm $\mathcal{A}$ that solves the problem even if $D(0)$ is disconnected. Consider the universe composed of only two robots, r and s, initially at distance more than V from each other. Consider the execution of the algorithm under a fully synchronous scheduler (i.e., the robots perform each phase of their cycle simultaneously and move at the same speed). Under such a scheduler, both r and s see the world in the same way (namely no other robot is in their circle of visibility), they will always compute the same move in the same direction, maintaining always the same distance; hence they will not gather. □

Thus, in the following we will always assume that $D(0)$ is connected. As the robots execute the algorithm, their distance conditions may vary. We will denote by $D(t) = (N, E(t))$ the *distance graph* at time $t \geq 0$; i.e., $\forall r, s \in N$, $(r, s) \in E(t)$ iff $0 \leq |r(t) - s(t)| \leq V$. We will now prove that the connectivity of the distance graph is preserved during the entire execution of the algorithm. We do so by first introducing the notion of *mutual visibility*.

Informally, two robots are mutually visible if they include each other in their computations. Formally, two robots r and s are *mutually visible* at time t iff both conditions hold:

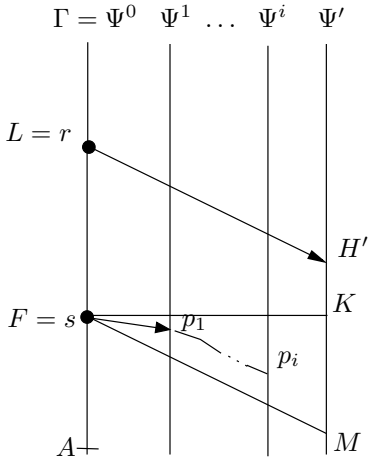


Figure 7: Case 2 of Lemma 4.9.

1. $0 < \overline{r(t)\ s(t)} \leq V$,
2. $r, s \in \mathbb{L}(t) \cup \mathbb{C}_\emptyset(t) \cup \mathbb{M}_\emptyset(t) \cup \mathbb{W}(t)$.

Since initially all robots are waiting, then by definition of mutual visibility we have

Lemma 4.8. *All the pairs of robots that are initially within distance V from each other are initially mutually visible.*

Note that the definition of mutual visibility does not include robots that are on the same point; i.e., $r(t) = s(t)$. This is because we do *not* assume that the robots have the capability of *multiplicity detection* (i.e., the ability to determine in the look phase if more than one robot is in a given location). In fact, as we will see, our gathering algorithm works without requiring such an additional assumption. We do therefore say that two robots r and s are *inert on the same point* at time t iff both conditions hold:

1. $r(t) = s(t)$,
2. $r, s \in \mathbb{L}(t) \cup \mathbb{C}_\emptyset(t) \cup \mathbb{M}_\emptyset(t) \cup \mathbb{W}(t)$.

We will now show that, if two robots are or become mutually visible, they will stay mutually visible until they become inert on the same point. We first prove that this property holds when two mutually visible robots lie on the same vertical axis; then we prove that it holds for two mutually visible robots lying on different vertical axes. In the next lemma we will refer to the notation introduced in Lemma 4.3 and 4.4.

Lemma 4.9. *Let r and s be mutually visible at time t. Moreover, let them lie, at time t, on the same vertical axis Γ, with s being below r. Then there is a time $\overline{t} > t$ when r and s are mutually visible or are inert on the same point. Moreover, at all times between t and $\overline{t}$, r and s are at distance at most V.*

Proof. Observe that the lemma trivially holds if $\mathcal{L}$ is not empty at time $First(r,t)$. In fact, according to the Algorithm, r will perform a null move and perform a new *Look* at time $t' > t$; since r is above s from time t to t', according to the Algorithm, s can not move during this time. Thus, the lemma holds with $\overline{t} = t'$. We will thus assume that $\mathcal{L}$ is empty at time $First(r,t)$.

Let us first consider the case when $\mathcal{R}$ is empty at time $First(r,t)$. Then, according to the Algorithm, r can only perform a Vertical Move towards s, shortening their distance. Let $t' > First(r,t)$ be the time when r completes its move; thus $r \in \mathbb{W}(t')$. By Lemma 4.6, r will be above s during its move; hence, according to the Algorithm, s can not move before r completes its move. In other words, $s \in \mathbb{W}(\cdot) \cup \mathbb{L}(\cdot) \cup \mathbb{C}_{\emptyset}(\cdot) \cup \mathbb{M}_{\emptyset}(\cdot)$ from time t until time t'; and $\forall l \in \overline{r(t)\ s(t')}$, $\overline{l\ s(t')} \leq V$. Thus, the lemma holds with $\overline{t} = t'$.

Let us now consider the more interesting case when $\mathcal{R}$ is not empty at time $First(r,t)$. Then the move computed by r as a consequence of its *Look* at time $First(r,t)$ is a Diagonal Move. Let L be the position occupied by r on Γ, H be the destination point computed by r, H' be the point where r stops in its movement towards the computed destination, and Ψ' be the vertical axis where H' resides. Let t' and t'' be the time when r starts its move and when it stops, respectively. Clearly, s can not move before time $\max\{t', First(s,t')\}$.

Claim 4.1. *As long as s does not move, the distance between r and s is at most V during time t' to t''.*

Proof. Since $\overline{AH} = V$ (see Figure 4.c) and $\overline{LH'} \leq \overline{LH} \leq V$ (Lemma 4.1), the Claim follows from Lemma 2.1 on $\triangle(L, A, H)$. □

We shall now consider two possible situations:

Case 1: s does not look while r is moving or, whenever it looks between $First(s,t')$ and t'', it decides not to move (because it sees some robots above or to its left).

In this case, at time t'' when r stops at H', $s \in \mathbb{L}(t'') \cup \mathbb{C}_{\emptyset}(t'') \cup \mathbb{M}_{\emptyset}(t'') \cup \mathbb{W}(t'')$; clearly $r \in \mathbb{W}(t'')$. By Claim 4.1, $\overline{H'\ s(t)} = \overline{r(t'')\ s(t'')} \leq V$. That is, they are mutually visible at time t'' and, by Claim 4.1, they are at distance at most V from each other from t to t''; hence the lemma follows with $\overline{t} = t''$.

Case 2: s decides to move as a consequence of some of its *Look* operations performed between time t' and t''.

Let $t_0, t_1, t_2, \ldots, t_k$, $k \geq 0$, be the times when s performs a *Look* operation whose result is a real movement, $t' < t_0 < t_1 < t_2 < \ldots < t_k \leq t''$; let d_{i+1} be the destination point computed at time t_i, p_{i+1} be the point where s stops in that move, and Ψ^{i+1} be the axis where p_{i+1} resides, with $0 \leq i < k$. Let $\Psi^0 = \Gamma$, and $F = p_0$ be the position of s on Γ at time t (refer to Figure 7). Before proceeding, we establish the following Claim.

Claim 4.2. *Let $0 \leq i < k$. Then*

a. *Ψ^i is to the left of Ψ^{i+1}, and Ψ^{i+1} is to the left of $r(t_i)$;*

b. *The destination point d_{i+1} that s computes as a result of the Look operation performed at time t_i is inside $\triangle(F, K, M)$;*

c. $\overline{p_{i+1}\ r(t_i)} \leq V$.

Proof. We proceed by induction on i.

Basis Since s by definition does not move between time t and t_0, then by Claim 4.1, when s looks at time t_0 it will see r to its right. This implies that, according to the Algorithm, s will perform either a Horizontal or a Diagonal Move. Therefore, d_1 is to the right of Ψ^0; furthermore, by Lemma 4.5, d_1 can not be to the right of $r(t_0)$. Hence, since the movement of r is continuous, s can not reach $r(t_0)$, and claim **a** follows.

Furthermore, $\overline{\Gamma\Psi^1} < \overline{\Gamma\Psi'}$ and, by Lemma 4.3, $\overline{Fd_1}$ must lie above $\overline{FM}$; hence, p_1 (that is on $\overline{Fd_1}$) and d_1 must be within $\triangle(F, K, M)$, and claim **b** follows.

Finally, since d_1 is to the left of r at time t_0, by applying Lemma 4.4 on segment $\overline{Fd_1}$, we have that $\overline{p_1\ r(t_0)} \leq V$, and claim **c** follows. Hence the basis of the induction holds.

Inductive Step Let the claim hold for $1 \leq l \leq i$. By inductive hypothesis, $r(t_{i-1})$ is to the right of Ψ^i and is visible by s when it looks at time t_{i-1}. Furthermore, by Lemma 4.5, d_i (and, thus p_i) can not be to the right of $r(t_{i-1})$. This means that, when s, after stopping at p_i, performs a *Look* operation at time t_i, r will be visible (by inductive hypothesis on claim **c**) and it will be to the right of s (recall that the movements are continuous). Then, according to the Algorithm, s can only perform a Horizontal or a Diagonal Move; therefore d_{i+1} is to the right of Ψ^i. Furthermore, by Lemma 4.5, d_{i+1} can not be to the right of $r(t_i)$. Hence, since the movement of r is continuous, s can not reach $r(t_i)$, and claim **a** follows.

Moreover, $\overline{\Psi^i\Psi'} < \overline{\Gamma\Psi'}$ and, by Lemma 4.3 the angle β computed by s on p_i at time t_i must be greater than or equal to the angle computed by r when it looked at time t from point L; hence, since by construction the segment $\overline{FM}$ is parallel to $\overline{LH'}$, $\overline{p_i d_{i+1}}$ is above $\overline{FM}$, but cannot be above $\overline{FK}$ (because the algorithm does not allow "up" movements). Therefore claim **b** follows.

Furthermore, since d_{i+1} is to the left of $r(t_i)$, claim **b** holds, and Ψ^{i+1} can not be to the right of d_{i+1}, by applying Lemma 4.4 on segment $\overline{p_i d_{i+1}}$, we have that $\overline{p_{i+1} r(t_i)} \leq V$, and claim **c** follows, completing the proof by induction.

□

By Claim 4.2, it follows that all the stops $p_0, p_1, \ldots p_k$ that s performs while r is moving towards H' are inside $\triangle(F, K, M)$; hence, by Lemma 4.4, they are within distance V from r during its move. Thus, when r stops at H' at time t'' and then *Look*s, it sees s on its left. According to the Algorithm, from t'' onwards, as long as s is to the left of r, r can be only in $\mathbb{W}(\cdot)$, $\mathbb{L}(\cdot)$, $\mathbb{C}_\emptyset(\cdot)$, or $\mathbb{M}_\emptyset(\cdot)$. Therefore, the first time that s stops after r reaches H' at time t'', say at time $\tilde{t}$, r and s will be mutually visible. Moreover, between t and $\tilde{t}$, by Lemma 4.4, we always have $\overline{rs} \leq V$, and the lemma follows with $\overline{t} = \tilde{t}$. □

Next, we consider the case when the two mutually visible robots are on different axes. Let B and C be respectively the upper and lower intersection between Ψ and $\mathcal{C}_r(t)$, and z be the intersection between $\mathcal{C}_r(t)$ and the line passing through $r(t)$ and H.

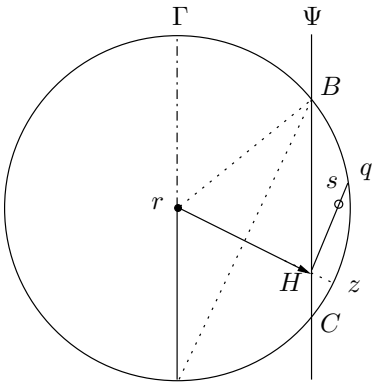


Figure 8: Lemma 4.10.

Lemma 4.10. *Let r and s be mutually visible at time t. Moreover, let s be located to the right of Ψ. Then there is a time $\overline{t} > t$ when r and s are mutually visible or are inert on the same point. Moreover, between t and $\overline{t}$, r and s are at distance at most V.*

Proof. Observe that the lemma trivially holds if $\mathcal{L}$ is not empty at time $First(r,t)$. In fact, according to the Algorithm, r will perform a null move and perform a new *Look* at time $t' > t$; since r is to the left of s from time t to t', according to the Algorithm, s can not move during this time. Thus, the lemma holds with $\overline{t} = t'$. We will thus assume that $\mathcal{L}$ is empty at time $First(r,t)$.

Let H be the destination point r computes as a consequence of this *Look* operation (the argument holds for both Horizontal and Diagonal Move). Let us first prove that $\forall l \in \overline{rH}$, $\overline{ls(t)} \leq V$. Side $\overline{Hq}$ of the triangle $\triangle(r,H,q)$, where q is the intersection point of $arc(BC)$ with the line passing through H and $s(t)$, has length no more than V, because the distance from r to q is V, and q lies on H's side of the bisector between r and H. Hence, the longest side of the triangle $\triangle(r,H,q)$ is $\overline{rq}$ with length V, and therefore by Lemma 2.1 the length of $\overline{rs(t)}$ is no more than V. Again by Lemma 2.1, the length of any chord $\overline{ls(t)}$ for l on $\overline{r(t)H}$ in triangle $\triangle(r,H,s(t))$ is no more than V. In other words, while r is moving towards H, its distance from $s(t)$ is at most V.

Consider now s. By definition, it is to the right of Ψ and in $\mathcal{C}_r(First(r,t))$. According to the Algorithm, s cannot perform any movement, as long as r is on its left. By Lemma 4.5, the destination point H can not be to the right of $s(t)$; hence r will be to the left of $s(t)$ while it is moving. Hence $s \in \mathbb{W}(\cdot) \cup \mathbb{L}(\cdot) \cup \mathbb{C}_\emptyset(\cdot) \cup \mathbb{M}_\emptyset(\cdot)$ from time t until r ends its move, say at time t''; in particular, $s(t) = s(t'')$. Since when r stops at time t'' we have that, by definition, $r \in \mathbb{W}(t'')$, the lemma holds at time $\overline{t} = t''$. □

The above lemmas show that, if two robots are mutually visible, they will continue to be so at future times until they become inert on the same point. Notice that a pair of robots that becomes inert on the same point, and thus are *not* mutually visible at that time, may later become mutually visible again.

Mutual visibility is defined at discrete points in time, and these points are different for different pairs of robots; still it is possible to capture globally and in a continuous way the information about the distance between robots contained in the mutual visibility

relationship. Let us consider the relationship of *visual neighboring* defined as follows.

Two robots x and y are visual neighbors at time t if one of the following two conditions holds:

1. they were mutually visible at some time $t' \leq t$ and have not become inert on the same point in the interval of time (t', t).

2. there exists a visual neighbor z of x at time t and y and z are inert on the same point at time t.

Notice that, according to the first condition, when two visual neighbors become inert on the same point at time t, they are no longer visual neighbors at time t; when that happens, according to the second condition, the visual neighbors of each become visual neighbors of both.

Let $G(t)$ be the undirected graph corresponding to the visual neighboring binary relation; i.e. $(x, y) \in E(G(t))$ iff x and y are visual neighbors at time t.

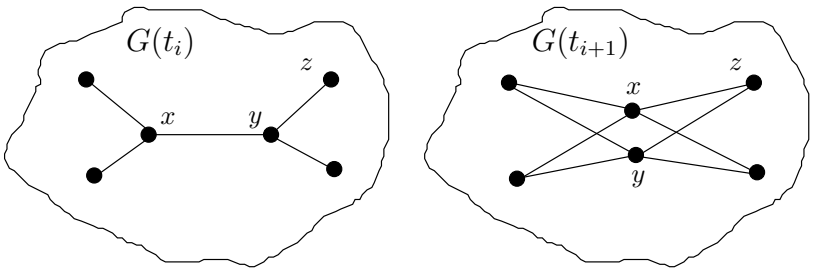
Lemma 4.11. *At time $t \geq 0$, if the robots are not all inert on the same point, then $G(t)$ is a subgraph of $D(t)$.*

Proof. We must show that if $(x, y) \in E(G(t))$ then their distance at t is at most V. By definition of the visual neighboring relation, all robots that are mutually visible are visual neighbors; thus, by Lemmas 4.9 and 4.10, they will remain visual neighbors and at distance at most V until they become inert on the same point.

When two visual neighbors x and y become inert on the same point, say at time t, their neighbors can not distinguish between them; in particular, their distance to both of them is the same and clearly at most V. Furthermore, x and y become indistinguishable also from a behavioral view point: since x and y are inert on the same point at time t, whatever x may subsequently do could have been done by y instead. In other words, the current execution and the one in which x and y are switched are equally possible and indistinguishable by their neighbors. Thus, by Lemmas 4.9 and 4.10, every robot z that was mutually visible with x (resp. y) at time $t' \leq t$ and did not become inert on the same point with x (resp. y) during the interval (t', t), will become either mutually visible or inert on the same point with y (resp. x) and, from time t until then, its distance from y (resp. x) will be at most V. □

Lemma 4.12. *At time $t \geq 0$, if the robots are not all inert on the same point, then $G(t)$ is connected.*

Proof. Initially $G(0) = D(0)$, which is by definition connected. Over time, the graph changes are due only to two types of events. The first type of events is when two robots that are not visual neighbors, become mutually visible; in this case an edge is added, and thus existing connectivity is maintained. The second type of events is when two robots that are visual neighbors become inert on the same point. In this case, the visual neighbors of each of the two robots become connected by an edge to both, and the edge connecting the two robots is removed; thus we must just show that removal of such an edge



a. b.

Figure 9: Lemma 4.12: (a) robots x and y are visual neighbors at time t_i. (b) At time t_{i+1} they become inert on the same point p, and the graph G stays connected.

does not disconnect the graph. Let $t_1, t_2, \ldots$ be the times when such events occur, and let $t_0 = 0$. Let $G(t_i)$ be connected, $i \geq 0$. Let x and y be a pair of visual neighbors that become inert on the same point p at time t_{i+1} (see Figure 9). Let I be the set of robots that become inert on p at time t_{i+1}; by definition, I includes x and y. By connectivity of $G(t_i)$ it follows that, unless $I = N$ (i.e., at time t_{i+1} all robots have gathered on p), there exists at least a robot $z \notin I$ such that $(w, z) \in G(t_i)$ where $w \in I$; hence both (x, z) and (y, z) will be added to $G(t_{i+1})$ and the removal of (x, y) will not disconnect the graph. □

By Lemmas 4.11 and 4.12 we can conclude that:

Theorem 4.1. *The distance graph $D(t)$ is connected during the execution of the Algorithm.*

4.3 Finiteness

In this section we prove that the robots will gather in a point after a finite number of steps. Recall that the distance graph is connected at all times.

First of all, we show that, until all robots are on the same axis, some robot will indeed move towards the *Right* axis of the universe $\mathfrak{U}$.

Lemma 4.13. *Let Λ be a vertical axis with one or more robots on it, and with no robots to its left. If there are robots on the right of Λ, in a finite number of steps one of the robots on Λ will leave Λ. Otherwise, all the robots will reach the bottom-most robot on Λ.*

Proof. Let us consider first the case when there are robots to the right of Λ. Let r be a topmost robot on Λ that can see a robot to the right of Λ, and let $\mathbb{S}$ be the set of robots on Λ above r. If $\mathbb{S} = \emptyset$, according to the Algorithm, r will move "in the next step" and the lemma trivially follows. Let $\mathbb{S} \neq \emptyset$. Assume, by contradiction, that no robot in $\mathbb{S} \cup \{r\}$ leaves Λ in a finite number of steps and that there is a non empty subset $\mathbb{R} \subseteq \mathbb{S}$ of robots that do not reach r in a finite number of steps. Let $s \in \mathbb{R}$; since s does not reach r, there exists at least a point on Λ, above r, which s does not pass in a finite number of steps. Let $p(s)$ be the topmost such point, let p be the topmost of $p(s)$ for all $s \in \mathbb{R}$, and let $\mathbb{R}' \subseteq \mathbb{R}$ be the set of robots that never pass p. Within finite time, all robots in $\mathbb{S} - \mathbb{R}'$ will be below p; consider a time t when only robots in $\mathbb{R}'$ are above p. Since the cycle time of

any robot is finite (Assumption A2), and by Theorem 4.1 the connectivity of the distance graph is always preserved, the topmost robot in $\mathbb{R}$ must perform a movement in finite time; its move must be vertical because the robot is not allowed to leave Λ by hypothesis. It follows that all robots in $\mathbb{R}'$ will get closer and closer to p. After a finite number of steps, all the robots in $\mathbb{R}'$ will be at distance smaller than δ from p; let $t' > t$ be a time when this happens. Consider now a point p' above p and below the bottommost (at time t') robot in $\mathbb{R}$. We know that all robots in $\mathbb{R}'$ must pass p' in a finite number of moves. Since there are no robots between p' and p, the first robot s which passes p' must have as its destination a point below p. Since the distance between s and p is smaller than δ, s will pass p (Assumption A1) leading to a contradiction. Hence, within finite time, $\mathbb{S}$ will become empty, and the Lemma holds.

Consider now the case when there are no robots on the right of Λ. Let r be the bottom-most robot on Λ, then, by the same argument as above, all the robots on Λ will reach r within finite time, and the Lemma holds. □

Since the destination point that a robot computes might be at a distance *smaller* than δ, the existence of an infinite sequence of infinitesimally small movements is not ruled out by the above lemma. The next Lemma shows that such a situation is impossible: within finite time, all robots in the system go onto the *Right* axis of the universe $\mathfrak{U}$.

Lemma 4.14. *All robots will reach Right in a finite number of steps.*

Proof. Assume by contradiction, that some robots never reach *Right*. This means that there are some axes that will not be passed by all the robots that were to their left at the beginning of the algorithm: we call them *limit axes*. Let Ψ be the leftmost such axis. Let $\mathbb{A}$ be the sets of robots, initially to the left of Ψ, that will become arbitrarily close to Ψ but never reach it; let $\mathbb{B}$ be the sets of robots, initially to the left of Ψ, that will pass Ψ within finite time; and let $\mathbb{C}$ be the sets of robots, initially to the left of Ψ, that will reach Ψ without ever moving to its right.

Let t be a time when all robots in $\mathbb{B}$ have passed Ψ and those in $\mathbb{C}$ have reached Ψ; that is, at time t the only robots to the left of Ψ are those in $\mathbb{A}$.

Consider first the case when $\mathbb{A} = \emptyset$. In this case, by Lemma 4.13, after a finite number of moves one of the robots in $\mathbb{C}$ will leave Ψ: a contradiction.

Let then $\mathbb{A} \neq \emptyset$. Consider a vertical axis Ψ' to the left of Ψ, at distance $\delta' < \delta \sin 60°$ from Ψ. Since Ψ is the leftmost limit axis, each $r \in \mathbb{A}$ will be to the right of Ψ' within finite time; observe that, once on the right of Ψ', r must stop at least once since, by definition, it does not reach Ψ. Let $t' > t$ be a time when all robots in $\mathbb{A}$ have stopped at least once to the right of Ψ'; and let Ψ'' be an axis, between Ψ' and Ψ, such that at time t' no robot in $\mathbb{A}$ is to its right. Since Ψ'' is not a limit axis, the robots of $\mathbb{A}$ will pass Ψ'' within finite time. Since at time t' there are no robots between Ψ'' and Ψ, the first robot $r \in \mathbb{A}$ that passes Ψ'' must have as its destination a point to the right of Ψ or on Ψ. According to the Algorithm, r will move on a straight line at an angle β, ($60° \leq \beta \leq 90°$); such a line intersects Ψ at a point H. Since this move by r is started from a point S to the right of Ψ' (possibly on Ψ''), then $\overline{SH} < \frac{\delta'}{\sin\beta} < \frac{\sin 60°}{\sin\beta} \cdot \delta \leq \delta$. Thus, by Assumption A1, in this move r will reach Ψ: a contradiction.

Hence no limit axis Ψ exists. That is, within finite time, all robots will reach *Right*. □

By Assumption A2, Lemmas 4.13 and 4.14 it follows that:

Theorem 4.2. *In a finite number of steps, and in finite time, all the robots in the system gather in a point on Right.*

5 Conclusions and Open Problems

In this paper we have analyzed the gathering problem for groups of autonomous mobile robots, and presented a gathering algorithm (to be executed by each individual robot) which solves the problem within *finite time*. Our solution operates in *fully asynchronous* settings and can be achieved by robots which are anonymous, oblivious and with limited visibility, provided they have orientation.

The results of this paper open many interesting research directions. Coordination problems should be studied for more powerful robots, where, for instance, the robots are not totally oblivious, and can remember a small amount of information. Simple tasks should be investigated under different conditions on the environment, such as the presence of obstacles, or robots moving on uneven terrains. The impact that sensorial errors and inaccuracies have on the correctness of the algorithms should be studied in detail. New algorithms are needed for different assumptions on the visibility power of the robots; for instance, the accuracy of the robots' ability to detect the other robots' positions might decrease with the distance. New problems (other than gathering) need to be solved, like *scattering* (i.e., the robots start from the same location and their goal is to evenly scatter on the plane), *rescue* (i.e., the robots have to find a small object which is not initially visible), *exploration* (i.e., the robots have to gather information about the environment, with the purpose, for example, of constructing a map), and many more.

At a more general level, there are interesting fundamental research questions. For example, how to compare different solutions for the same set of robots? So far, no complexity measure has been accepted; such a definition is part of our future research.

Finally, the results presented here are a further step towards a better understanding of the minimal robot capabilities enabling the collective resolution of a given global task. In particular, they show how one property ("orientation") has at least the same computational power for the gathering problem as another assumption ("instantaneous action"). This fact raises many intriguing questions, including: Are there other (simpler) properties comparable to these two? How do we compare different properties?

An important concern is clearly the one of the presence of possible faulty robots. The fault-tolerant issues have been recently addressed in [1, 11], but only in the unlimited visibility setting. An open problem is to examine them under limited visibility.

Acknowledgments

We are grateful to two referees for their suggestions that helped us to improve the presentation of the paper. This work has been supported in part by NSERC, MIUR, and by the Swiss BBW within the EU project COST 293.

References

[1] N. Agmon and D. Peleg. Fault tolerant gathering algorithms for autonomous mobile robots. In *Proc.* 15^{th} *Symposium on Discrete Algorithms (SODA '04)*, pages 1063–1071, 2004.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[3] H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. In *Proc. IEEE Symposium of Intelligent Control*, pages 453–460, 1995.

[4] T. Balch and R. C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transaction on Robotics and Automation*, 14(6), 1998.

[5] G. Beni and S. Hackwood. Coherent swarm motion under distributed control. In *Proc. Distributed Autonomous Robotic Systems (DARS '92)*, pages 39–52, 1992.

[6] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, 1997.

[7] Q. Chen and J. Y. S. Luh. Coordination and control of a group of small mobile robots. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2315–2320, 1994.

[8] M. Cieliebak. Gathering non-oblivious mobile robots. In *Proc. Latin American Conference on Theoretical Informatics (LATIN '04)*, LNCS 2976, pages 577–588, 2004.

[9] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the gathering problem. In *Proc.* 30^{th} *International Colloquium on Automata, Languages and Programming (ICALP '03)*, pages 1181–1196, 2003.

[10] M. Cieliebak and G. Prencipe. Gathering autonomous mobile robots. In *Proc.* 9^{th} *International Colloquium on Structural Information and Communication Complexity (SIROCCO '02)*, pages 57–72, 2002.

[11] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. In *Proc.* 12^{th} *Annual European Symposium on Algorithms (ESA '04)*, pages 228–239, 2004.

[12] R. Cohen and D. Peleg. Robot convergence via center-of-gravity algorithms. In *Proc.* 11^{th} *International Colloquium on Structural Information and Communication Complexity (SIROCCO '04)*, pages 79–88, 2004.

[13] X. Défago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proc. Workshop on Principles of Mobile Computing*, pages 97–104, 2002.

[14] J. Desai, J. Ostrowski, and V. Kumar. Control of formations for multiple robots. *IEEE International Conference on Robotics and Automation*, pages 2864–2869, May 1998.

[15] E. H. Durfee. Blissful ignorance: Knowing just enough to coordinate well. In *International Conference on MultiAgent Systems (ICMAS '95)*, pages 406–413, 1995.

[16] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Proc.* 10^{th} *International Symposium on Algorithm and Computation (ISAAC '99)*, pages 93–102, 1999.

[17] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous mobile robots with limited visibility. In *Proc.* 18^{th} *International Symposium on Theoretical Aspects of Computed Science (STACS '01)*, pages 247–258, 2001.

[18] D. Jung, G. Cheng, and A. Zelinsky. Experiments in realising cooperation between autonomous mobile robots. In *Proc.* 5^{th} *International Symposium on Experimental Robotics (ISER '97)*, pages 513–524, 1997.

[19] Y. Kawauchi, M. Inaba, and T. Fukuda. A principle of decision making of cellular robotic System (CEBOT). In *Proc. 1993 IEEE International Conference on Robotics and Automation*, pages 833–838, 1993.

[20] M. J. Matarić. *Interaction and Intelligent Behavior*. PhD thesis, MIT, May 1994.

[21] S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In *Proc. 1994 IEEE International Conference on Robotics and Automation*, pages 441–448, 1994.

[22] G. Prencipe. *Distributed Coordination of a Set of Autonomous Mobile Robots*. TD–4/02, University of Pisa, 2002.

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

[24] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation and agreement problems. In *Proc.* 3^{rd} *International Colloquium on Structural Information and Communication Complexity (SIROCCO '96)*, pages 313–330, Siena, 1996.

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