Gathering of Asynchronous Oblivious Robots with Limited Visibility

Paola Flocchini[1], Giuseppe Prencipe[2], Nicola Santoro[3], and Peter Widmayer[4]

[1] University of Ottawa, flocchin@site.uottawa.ca
[2] Università di Pisa, prencipe@di.unipi.it
[3] Carleton University, santoro@scs.carleton.ca
[4] ETH Zürich, pw@inf.ethz.ch

Keywords: Distributed Algorithms, Coordination and Control, Mobile Robots.

Abstract. In current robotics research, the trend has been to move away from the design and deployment of few, rather complex, usually expensive, application-specific robots. Instead, the interest has shifted towards the design and use of a large number of "generic" robots which are very simple, with very limited capabilities and, thus, relatively inexpensive. In particular, each robot is only 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 sensing, computing, moving and being inactive.
A very important set of questions refer to determining the robots capabilities; that is how "simple" the robots can be [3] to perform the required task.
In this paper we are interested in *gathering*: the basic task of having the robots meet in a same location (not determined in advance). We study the problem in the most general case of an asynchronous system of robots with limited visibility, drastically differing from previous solutions to the problem based on the assumptions that the robots must be capable of performing all the sensing, computing and moving actions instantaneously.

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, rather complex, usually expensive, application-specific robots. Instead, the interest has shifted towards the design and use of a large number of "generic" robots which are very simple, with very limited capabilities and, thus, relatively inexpensive.

In particular, each robot is only 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 sensing, computing, moving and being inactive (e.g., see [2, 7–9]). On the other hand, the robots should be able, together, of performing rather complex tasks. Examples of typical basic tasks are *gathering*, *leader election*, *pattern formation*, *scattering*, etc.

A very important set of questions refer to determining the robots capabilities; that is how "simple" the robots can be to perform the required task [3]. In computational terms, this question is to identify the factors which influence solvability of a given problem (the task).

These questions have been extensively studied both 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 [4, 6, 10, 12]).

In general and more realistically, robots can sense only a surrounding with a radius of bounded size. This setting, called the *limited visibility* case, is understandably more difficult, and only few algorithmic results are known [1, 11].

In this paper we are interested in *gathering*: the basic task of having the robots meet in a same location (the choice of the location is arbitrary). Since the robots are modeled as points in the plane, the task of robots gathering is also called the *point formation problem*. Gathering (or point formation) has been investigated both experimentally and theoretically. In particular, in the limited visibility setting, Ando et al. [1] presented a gathering algorithm for indistinguishable robots which are placed on a plane without any common coordinate system; their algorithm does not require the robots to remember observations nor computations performed in the previous steps. Their result implies that gathering can be performed with limited visibility by very simple robots: *anonymous*, *oblivious* and *disoriented*.

Their solution, however, is based on a very strong "atemporal" assumption on the duration of the robots' actions: their robots must be capable in every cycle to perform all the sensing, computing and moving *instantaneously*.

This assumption has many consequences crucial for its correctness. For example, since movement is instantaneous, a robot can *not* be seen by the others while moving (and its temporary position mistaken for a destination location); since sensing and computing is instantaneous, a robot always has available the correct current situation of its neighborhood.

Note that, since instantaneous movement is not physically realizable, their solution is only of theoretical interest.

In this paper, we study the gathering problem in the most general case of an *asynchronous* system of robots with limited visibility, where both their computations and their movement requires a *finite* but otherwise unpredictable amount of time.

The question motivating our investigation is whether point formation is possible in such a system. Since in these systems gathering is *unsolvable* if the robots are disoriented (i.e., have no common system of coordinates), we shall restrict ourselves to systems with sense of direction (i.e., the robots share the same coordinate system).

In this paper we show that indeed anonymous oblivious robots with limited visibility can gather within a finite number of moves even if they are fully asynchronous. In fact, we describe a new algorithm for solving the point formation problem in the asynchronous setting by anonymous oblivious robots with limited visibility. We then prove its correctness showing that the robots will gather in a point within a finite amount of time. 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 in length (but not infinitesimally small).

In other words, we show that gathering can be performed by simpler robots with fewer restrictions than known before, provided they have a common coordinate system.

¿From a theoretical point of view, this result proves that, with respect to the gathering problem, "sense of direction" has the same computational power as "instantaneous actions".

¿From a practical point of view, this result has fundamental consequences. In fact, it allows to substitute a theoretically interesting but physically unrealizable motorial and computing capability requirement (instantaneous actions) with a property (sense of direction) which is both simple and inexpensive to provide (e.g., by a compass).

The paper is organized as follows. In Section 2 the model under study is formally presented. In Section 3 the notations used in the paper and some useful geometric lemmas are introduced. The gathering algorithm is described in Section 4, and in Section 5 its correctness is proven. Due to space limitations, some of the proofs are omitted and can be found in [5].

2 The Model

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 sensing, computing, moving and being inactive.

The robots are modeled as units with computational capabilities, which are able to freely move in the plane. They are viewed as points, and are equipped with sensors that let each robot observe the positions of the others with respect to its local coordinate system. Each robot can see only a portion of the plane; more precisely, it can observe whatever is at most at a fixed distance V from it (*limited visibility*).

Each robot has its own *local view* of the world. This view includes a local Cartesian coordinate system with origin, unit of length, and the *directions* of two coordinate axes, together with their *orientations*, identified as the positive and negative sides of the axes. In this paper we assume that the robots share the same coordinate system (*sense of direction*); however, they do not necessarily agree on the location of the origin (that we can assume, without loss of generality, to be placed in the view of a robot in its own current position), nor on the unit distance.

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 communication occurs in a totally implicit manner. Specifically, it happens by means of observing the change of its fellows' positions in the plane while they execute the algorithm.

Summarizing, the robots are *oblivious*, *anonymous*, and with limited visibility; they do however have a common coordinate system.

They 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*); at any point in time, asynchronously and independently from the other robots, it observes the environment in its area of visibility (*Look*), it calculates its destination point based only on the current locations of the observed robots (*Compute*), it then moves towards that point (*Move*) and goes back to a waiting state. The sequence: *Wait (W)* - *Look (L)* - *Compute (C)* - *Move (M)* will be called a *computation cycle* of a robot.

The robots are fully *asynchronous*. In particular, the amount of time spent in a computation, in a movement, and in inactivity is finite but otherwise unpredictable. Moreover, a robot moving towards the computed destination can stop after an unpredictable amount of space, provided is neither infinite, nor infinitesimally small (unless it reaches its destination). More precisely, the only assumptions made are the following:

Assumption A1 Any robot will complete its cycle in an amount of time which is finite and bounded from below.

Assumption A2 The distance traveled by a robot in a move is finite and bounded from below (unless the destination is closer than the bound).

As a consequence, the (global) time that passes between two successive movements of the same robot is finite; furthermore, while a robot is moving, it can be seen an unpredictable but finite number of times by another robot.

3 Notations and Geometric Lemmas

We first define sets related to which state a robot is at a given time during the computation.

$W(t)$ and $L(t)$ are the set of all the robots that are respectively in state W and L at time t.

$C(t) = C_\emptyset(t) \cup C_+(t)$ is the set of all the robots that at time t are computing. The set $C_\emptyset$ contains those robots whose computation's result is to stay still (we say that they execute a *null movement*), while C_+ contains those robots whose computation's result is some destination point (we say that they will execute a *real movement*).

$M(t) = M_\emptyset(t) \cup M_+(t)$ is the set of all the robots that at time t are executing a movement. The set $M_\emptyset(t)$ contains the robots executing a *null movement* (they stay still); $M_+(t)$ contains those executing a *real movement* (they are effectively moving towards a destination).

We define *circle of visibility* $\mathcal{C}_i(t)$ of a robot r_i at time t the circle of radius V centered in r_i, if $r_i \in L(t)$. Otherwise $\mathcal{C}_i(t) = \mathcal{C}_i(t')$, where $t' = \max\{\bar{t}|r_i \in L(\bar{t})\}$.

In other words, if a robot is *Observing*, its circle of visibility is the circle of radius V centered in itself; otherwise, it is the circle of radius V centered in the location of its most recent *Look phase*. Where no ambiguity arises, the parameter t in $\mathcal{C}_i(t)$ will be omitted.

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 points on a circle; with arc(AB) we indicate the smallest arc on the circle passing through A and B. r indicates a generic robot in the system (when no ambiguity arises, r is used also to represent the point in the plane occupied by robot r); capital italic letters indicate regions (e.g. $\mathcal{L}$, $\mathcal{R}$); given a region, we denote by $|\cdot|$ the number of robots in that region.

Lemma 1. *Every internal chord of a general triangle has length less or equal to the longest side of the triangle.*

Lemma 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 is less or equal to V.*

Lemma 3. *Let $\overline{OB}$ be the radius of a circle centered in O and D be a point on the circle such that $B\widehat{O}D = \beta$, with $0 \leq \beta \leq 90°$. Then $\overline{pC} \leq \overline{BC}$, $\forall p \in arc(BD)$ and $\forall C \in \overline{OD}$. (see figure 1.b)*

4 The Algorithm

Let us call *Universe* (U) the smallest isothetic rectangle containing the initial configuration of the robots and let us call *Right* and *Bottom* respectively, the rightmost and the bottom most side of U.

The idea of the algorithm is to make the robots move either towards the bottom or towards the right of the Universe (a robot will never move up or to its left), in such a way that, after a finite number of steps, they will gather at the bottom most lower most corner of the Universe.

A robot r can move only if it does not see any robot neither to its left nor above on its vertical axis. Several situations could arise depending on the positions of the robots in its area of visibility:

- If r does not see any robot, 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 towards the right.

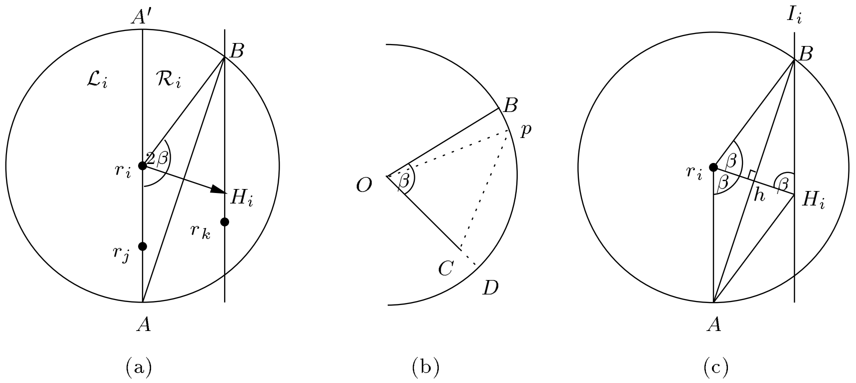

Fig. 1. (a) The Notation used in Algorithm 1; (b) Lemma 3; (c) Lemma 6.

Recall that $\mathcal{C}_i$ is the circle of visibility of robot r_i. Let $\overline{AA'}$ be the vertical diameter of such region; let $\mathcal{R}_i$ and $\mathcal{L}_i$ denote the regions to the right and to the left of r_i, respectively (see Figure 1). Let $S_p = \overline{r_iA'}$ and $S_o = \overline{r_iA}$.

Algorithm 1 (`Gathering`)

$Extrem := (|\mathcal{L}_i| = 0 \wedge |S_p| = 0)$;
If I am $\neg Extrem$ **Then**
  `Do_nothing()`;
Else
  If $(|\mathcal{R}_i| = 0 \wedge |S_o| = 0)$ **Then**
    `Do_nothing()`;
  If $|\mathcal{R}_i| = 0$ **Then**
    $r_j :=$ nearest visible robot on S_o;
    `Move(`r_j`)`.
  If $(|\mathcal{R}_i| \neq 0 \wedge |S_o| = 0)$ **Then**
    $I_i :=$ `Nearest()`;
    $H_i :=$ `H_Destination(`I_i`)`;
    `Move(`H_i`)`.
  If $|\mathcal{R}_i| \neq 0$ **Then**
    $I_i :=$ `Nearest()`;
    `Diagonal_Movement(`I_i`)`.

`Nearest()` returns the vertical axis on which the robot in $\mathcal{R}_i$ with the nearest axis to r_i lies.

`H_Destination(`I_i`)` returns the intersection between I_i and a line parallel to the x direction and passing through r_i.

`Move(p)` terminates the local computation of the calling robot and moves it towards p.

In the last case of the Algorithm 1, r_i sees somebody below it and somebody to its right, therefore, to avoid losing some robots, it has to move diagonally, as indicated by the following routine.

Algorithm 2 (`Diagonal_Movement(`I_i`)`)

$B :=$ upper intersection between $\mathcal{C}_i$ and I_i;
$A :=$ point on S_o at distance V from me;
$2\beta = A\widehat{r_i}B$;
If $\beta < 60°$ **Then**
    $B :=$ `Rotate(`r_i, B`)`.
$H_i :=$ `D_Destination(`V, I_i, A, B`)`;
`Move(`H_i`)`.

`Rotate(`r_i, B`)` rotates the segment $\overline{r_i B}$ in such a way that $\beta = 60°$ and returns the new position of B.

With `D_Destination(`V, I_i, A, B`)`, r_i computes its destination in the following way: the direction of its movement is given by the perpendicular to the segment $\overline{AB}$; $H_i = \min\{$ V, the distance of I_i according the direction of movement$\}$.

5 Correctness

In this Section we will prove the correctness of the algorithm by first showing that the robots which are mutually visible at any point of the computation, will stay mutually visible until the end of the computation, and concluding that at the end of the computation all robots will gather in one point. We first introduce some lemmas. From Assumptions A1 and A2 it directly follows that:

Lemma 4. *Let r_i and r_j be two generic robots and let t and $t' > t$ two moment of the computation. If $r_i \in L(t)$, $r_i \in L(t')$, $r_j \in M(t)$, $r_j \in M(t')$, $r_j \in \mathcal{C}_i(t)$ and $r_j \in \mathcal{C}_i(t')$, then r_j can not be in the same point in t and t'.*

Moreover, from the Gathering algorithm it follows that:

Lemma 5. *Let r_j and r_i two arbitrary robots, with r_i to the right of r_j at time t. If $r_j \in L(t)$ and $\overline{r_j r_i} \leq V$, then r_j can not pass r_i in one step.*

Let us consider a generic robot r_i executing the algorithm. Let β be the angle between the vertical axis of r_i and the direction of its movement ($A\widehat{r_i}H_i$ in Figure 1.c).

Lemma 6. *The segment $\overline{r_i H_i}$ is always smaller or equal to V. Moreover, $\overline{BH_i} = \overline{AH_i} = V$ and $\overline{pH_i} \leq V$, $\forall\ p \in \overline{r_i A}$.*

Thus, $\diamond(A, r_i, B, H_i)$ is a parallelogram. We now introduce the definition of visibility graph. The *visibility graph* $G = (N, E)$ of the robots is a graph whose node set N is the set of the input robots and, $\forall r_i, r_j \in N$, $(r_i, r_j) \in E$ iff r_j and r_j are initially at distance smaller than the visibility radius V. We first show that the visibility graph must be connected in order for the algorithm to be correct.

Lemma 7. *If the visibility graph G is disconnected, the problem is unsolvable.*

Thus, in the following we will always assume that G is connected.

5.1 Preserved Visibility

In this Section we prove that the visibility graph is preserved during the entire execution of the algorithm. We prove so by introducing the notion of mutual visibility and by showing that the robots which are connected in the visibility graph (i.e., those which are initially within distance V) will eventually become mutually visible, and that two robots that are mutually visible at some point in the algorithm will stay mutually visible until the end of the computation.

Informally speaking, we say that two robots are mutually visible if each robot includes the other one in its computation, namely each of them had seen the other one during its observation phase. Formally, two robots r_1 and r_2 are *mutually visible* at time t iff

- $r_1 \in (L(t) \cup C_\emptyset(t) \cup M_\emptyset(t)) \wedge r_2 \in \mathcal{C}_1(t) \wedge r_2 \in (W(t) \cup L(t))$, or
- $r_2 \in (L(t) \cup C_\emptyset(t) \cup M_\emptyset(t)) \wedge r_1 \in \mathcal{C}_2(t) \wedge r_1 \in (W(t) \cup L(t))$.

Since all the robots at the beginning are in W, from the above definition we have that the robots that at the beginning are within distance V will become mutually visible in finite time. That is, the following lemma holds:

Lemma 8. *Let r_i and r_j be two robots that at the beginning are within distance V. Robots r_i and r_j will become mutually visible in a finite number of steps.*

We now introduce a couple of lemmas which will be useful to prove that mutually visible robots will stay so until the end of the algorithm. Let r_i be a generic robot on an axis S. Let S' and S'' be two vertical axes to the right of S. We will denote by $\overline{SS'}$ and $\overline{SS''}$ the distances between the corresponding axis. Then we have:

Lemma 9. $\overline{SS'} < \overline{SS''} \Leftrightarrow \beta_{S'} > \beta_{S''}$*, where $\beta_{S'}$ and $\beta_{S''}$ are respectively the angles computed by the routines* `Diagonal_Movement(`S'`)` *and* `Diagonal_Movement(`S''`)` *(Figure 2.a).*

Lemma 10. *Let us consider the situation depicted in Figure 2.b, where F is a point at distance $\leq V$ from r_i on its axis (with $F \neq r_i$), H_i is the destination point of r_i. Let $\overline{ps}$ be a segment in $\triangle(F, M, K)$, with s to the right of p, and s' the projection of s over $\overline{r_i H_i}$. Then we have $\overline{ll'} \leq V$, $\forall\ l\ \in \overline{ps}$, $\forall\ l' \in \overline{s' H_i}$.*

We are now ready to show that, as soon as two robots becomes mutually visible, they will stay mutually visible. We first prove that this property holds when two mutually visible robots lie on the same vertical axis; and then we prove that it holds for two robots lying on different vertical axes.

In the next lemma we will refer to the notation introduced in Figure 1.a and Lemma 10.

Lemma 11. *Let r_i and r_j be robots which are mutually visible at time t; moreover, let they lie, at time t, on the same vertical axis with r_j being below r_i. There is a time $t' > t$ when r_i and r_j are mutually visible. Moreover, between t and t' $\overline{r_i r_j} \leq V$.*

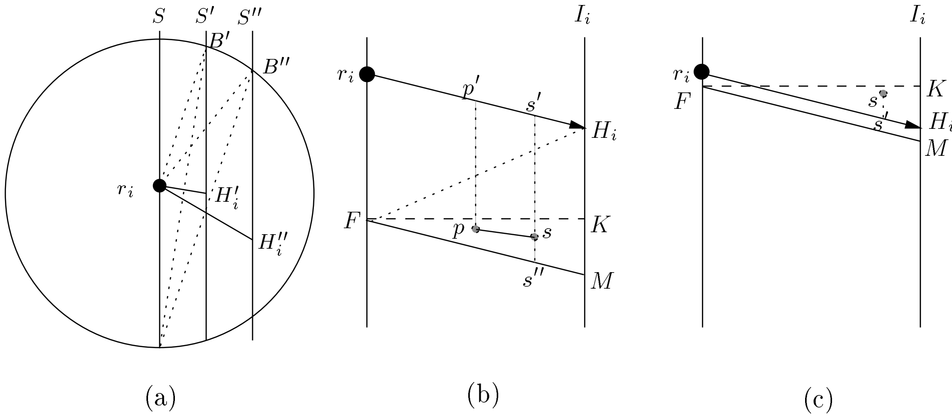

Fig. 2. (a) Lemma 9; (b) and (c) Lemma 10.

Proof. Let us first consider the case when $\mathcal{R}_i$ is empty. In such a case, r_i would clearly move towards r_j (shortening their distance), while r_j would not move. Since by Algorithm 1 r_i can not pass r_j, the first time r_i stops while it is moving towards r_j the mutual visibility definition holds, and the lemma follows.

Let us now consider the more interesting case when $\mathcal{R}_i$ is not empty. In the following we shall consider several situations:

Case i: r_j does not look until r_i reaches its destination H_i.
We have that $r_i \in W$ while r_i is moving towards H_i. Since $\overline{AH_i} = V$ (Lemma 6) and $\overline{r_iH_i} \leq V$ (Lemma 6), we have that, by Lemma 1 on $\triangle(r_iAH_i)$, the distance between r_i and r_j is always $\leq V$ while r_i is moving. Therefore, the first time r_i stops along its path (at most on H_i), the mutual visibility definition applies and the lemma follows.

Case ii: r_j looks while r_i is moving towards its destination H_i. Since r_i is on r_j's right, r_j can not perform a Vertical Move. Hence, r_j can either decide not to move (because it sees some robots above) or to move. In the first case the proof reduces to the one of Case *i*. On the other hand, r_j can decide to move after having looked. From Case *i* we know that r_j can see r_i on its right. Moreover, it might also see some other robots below it, that can be either on the same axis (r_j perform a Diagonal Move) or not (r_j performs an Horizontal Move). The following applies to both situations (Figure 3).
Let us call I_j^w the w^{th} axis, counting from S, from where r_j looks while r_i is still on its way towards H_i, and p_w the points on this axis from where r_j performs the *look* phases. Clearly $I_j^0 = S$ and $F = p_0$ coincides with the position of r_j on S. In the following we will prove by induction that

a. I_j^w is to the left of I_j^{w+1},

b. The destination point d_{w+1} that r_j computes when it is on I_j^w is inside $\triangle(F, K, M)$,

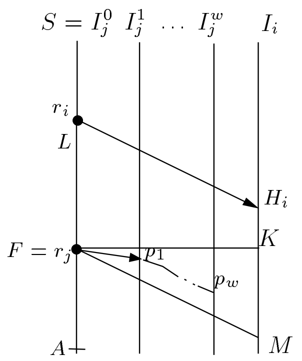

Fig. 3. Case ii of Lemma 11.

c. $\overline{p_{w+1}r_i} \leq V$, and I_j^{w+1} is to the left of r_i.

Basis Let d_1 be the first destination point r_j computes. Since r_i is on its right, r_j can only decide to perform a Diagonal Movement, therefore d_1 must be to the right of I_j^0, and as a consequence I_j^0 is to the left of I_j^1. Moreover, by Lemma 9 we know that $\overline{r_j d_1}$ must lie above $\overline{r_j M}$, hence p_1 (that is on $\overline{r_j d_1}$) must be whithin $\triangle(F, K, M)$. Finally, r_j can see r_i by hypothesis and at the beginning I_j^0 is to the left of r_i, and the basis of the induction follows.

Inductive Step Let us assume that all the statements are true for $1, \dots, w$. Since by inductive hypothesis I_j^w is to the left of r_i and r_j can see r_i from I_j^w, r_j can only decide to perform a Diagonal Movement, therefore d_{w+1} must be to the right of I_j^w and can not be after r_i (because of how `Diagonal_Movement(·)` works), and, as a consequence, I_j^w is to the left of I_j^{w+1}, and a. follows.

Moreover, since $\overline{I_j^w I_i} < \overline{SI_i}$ and , by Lemma 9, we have that $\overline{d_w p_{d+1}}$ must be above $\overline{FM}$ but cannot be above $\overline{FK}$ (because the algorithm does not allow "up" movements). Therefore the point b. follows.

Furthermore, since b. holds and I_j^{w+1} can not be after d_{w+1}, by Lemma 10 c. follows, and the induction is proved.

Now we know that all the stop r_j does while r_i is moving towards H_i are inside $\triangle(F, K, M)$, hence, by Lemma 10, whithin distance V from r_i. Thus we have that, when r_i reaches H_i, it can see r_j on its left, therefore, it can not move further. It follows that, until r_j is before it, r_i can be only in $L(\cdot)$, $C_\emptyset(\cdot)$, or $M_\emptyset(\cdot)$. Therefore, the first time that r_j stops after r_i reached H_i, say at time $t' > t$, r_i and r_j will be mutual visible. Moreover, between t and t', by Lemma 10 $\overline{r_i r_j} \leq V$, and the lemma follows. □

In the following lemma we show that if a robot sees some robots on its right, then it will never lose them during the computations. Let r_i be a robot in the system, R be the set of robots which are mutually visible with r_i at time t and that are located to the right of I_i, and r_k a robot in R (Figure 4). Moreover, let B and C be respectively the upper and lower intersection between I_i and $\mathcal{C}_i$, and H_i' be the intersection between $\mathcal{C}_i$ and the line passing through $\overline{r_i H_i}$.

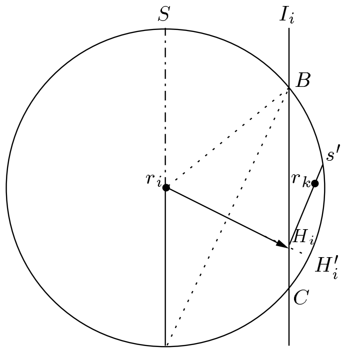

Fig. 4. Lemma 12.

Lemma 12. *There exists a time $t' > t$ after which r_i will be always mutually visible with the robots in R. Moreover, $\overline{r_i r^*} \leq V$, $\forall\ r^* \in R$.*

Proof. From Algorithm 1, we know that robots in R cannot perform any movement while r_i is on their left. Let t^* the time when r_i enters its Look phase and p be the destination point it computes. Clearly, p can not be to the right of any robot in R. In the following, we first prove that $\overline{lr^*} \leq V$, $\forall\ r^* \in R$ and $\forall l \in \overline{r_i p}$.

¿From Lemma 3, it follows that: $\forall p \in arc(BH_i'), pH_i \leq \overline{BH_i} = V$ (1). Moreover, $\overline{H_i C} = \overline{BC} - \overline{BH_i} \leq 2V - V = V$ and from Lemma 2 we have: $\forall p \in arc(H_i' C), pH_i \leq H_i C \leq V$ (2). Plugging (1) and (2) we obtain: $\forall p \in arc(BC), pH_i \leq V$ (3).

Let us now consider a robot $r_k \in sector(BCB)$ (that is in the area to the right of I_i and in $\mathcal{C}_i$) and let s' be the intersection between $arc(BC)$ and the line passing through H_i and r_k. We have that $\overline{H_i r_k} \leq \overline{H_i s'} \leq V$ (from (3)), $\overline{r_i r_k} \leq V$, and $\overline{r_i H_i} \leq V$. Therefore, applying Lemma 1 to $\triangle(r_i, r_k, H_i)$ we have that $\overline{q r_k} \leq V$, $\forall q \in \overline{r_i H_i}$. In conclusion, when r_i stops in p, say at time $t' > t$, it will see all the robots in R, that can only be in $L(t')$, $C_\emptyset(t')$, or $M_\emptyset(t')$, and the lemma follows. □

By Lemma 8, 11 and 12 we can conclude that:

Theorem 1. *The visibility graph G is preserved during the execution of the algorithm.*

5.2 Finiteness

In this section we will prove that, after a finite number of steps, the robots will gather in a point.

Lemma 13. *Let us suppose to have several robots on a vertical axis A and no robots to the left of A. If r is the topmost robot on A that can see a robot to the right of A, then, in a finite number of steps, either all the robots above r on A will reach r, or one of them will leave A.*

The next two lemmas show that all the robots in the system converge to the *Right* axis of the Universe, and actually reach it.

Lemma 14. *For any given vertical axis I before Right which is at any distance $d > 0$ from it, all the robots that are on the left of I at the beginning of the algorithm, will pass I in a finite number of steps.*

Lemma 15. *After a finite number of steps, all the robots in the system reach Right.*

The following lemma states what happens when all the robots lie on the same vertical axis: they will reach the bottom most robot on that axis in a finite number of steps.

Lemma 16. *If all the robots of the system lie on the same vertical axis A, then in a finite number of steps all the robots will reach the bottom most robot on A.*

We can finally conclude that:

Theorem 2. *In a finite number of steps, all the robots in the system gather in a point; the rightmost and bottom most corner of the universe.*

References

1. H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Trans. on Robotics and Autom.*, 15(5):818–828, 1999.
2. G. Beni and S. Hackwood. Coherent Swarm Motion Under Distributed Control. In *Proc. DARS'92*, pages 39–52, 1992.
3. E. H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *ICMAS*, pages 406–413, 1995.
4. 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 *ISAAC '99*, pages 93–102.
5. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Oblivious Gathering by A Set of Asynchronous and Autonomous Mobile Robots. Technical Report TR-00-00, Carleton University, 2000.
6. D. Jung, G. Cheng, and A. Zelinsky. Experiments in Realising Cooperation between Autonomous Mobile Robots. In *5th International Symposium on Experimental Robotics (ISER)*, June 1997.

7. Y. Kawauchi and M. Inaba and. T. Fukuda. A Principle of Decision Making of Cellular Robotic System (CEBOT). In *Proc. IEEE Conf. on Robotics and Automation*, pages 833–838, 1993.
8. M. J Matarić. *Interaction and Intelligent Behavior.* PhD thesis, MIT, May 1994.
9. S. Murata, H. Kurokawa, and S. Kokaji. Self-Assembling Machine. In *Proc. IEEE Conf. on Robotics and Autom.*, pages 441–448, 1994.
10. K. Sugihara and I. Suzuki. Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots. *Journal of Robotics Systems*, 13:127–139, 1996.
11. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots. In *Proc. of 3rd International Colloquium on Structural Information and Communication Complexity*, pages 313–330, Siena, 1996.
12. I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.