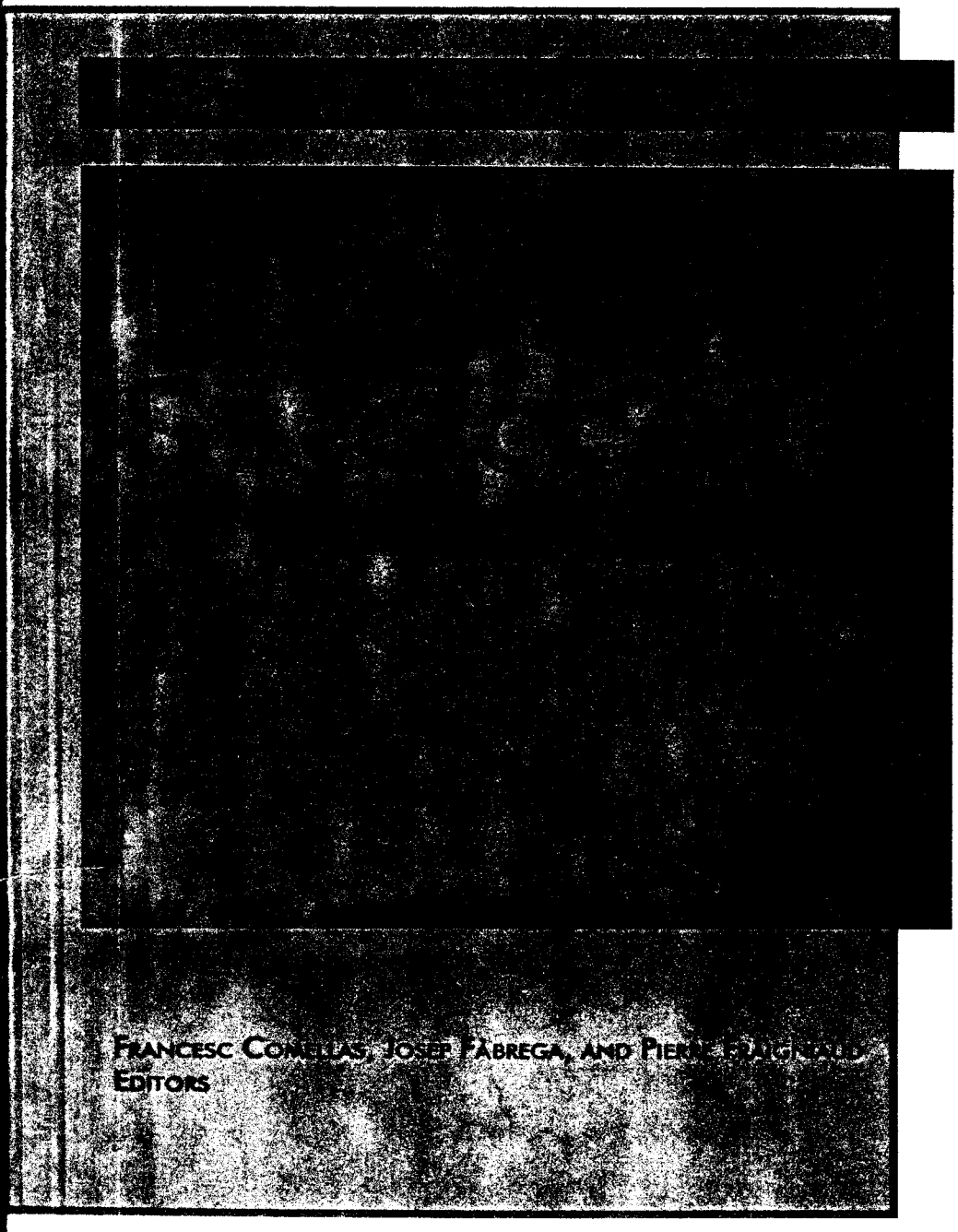
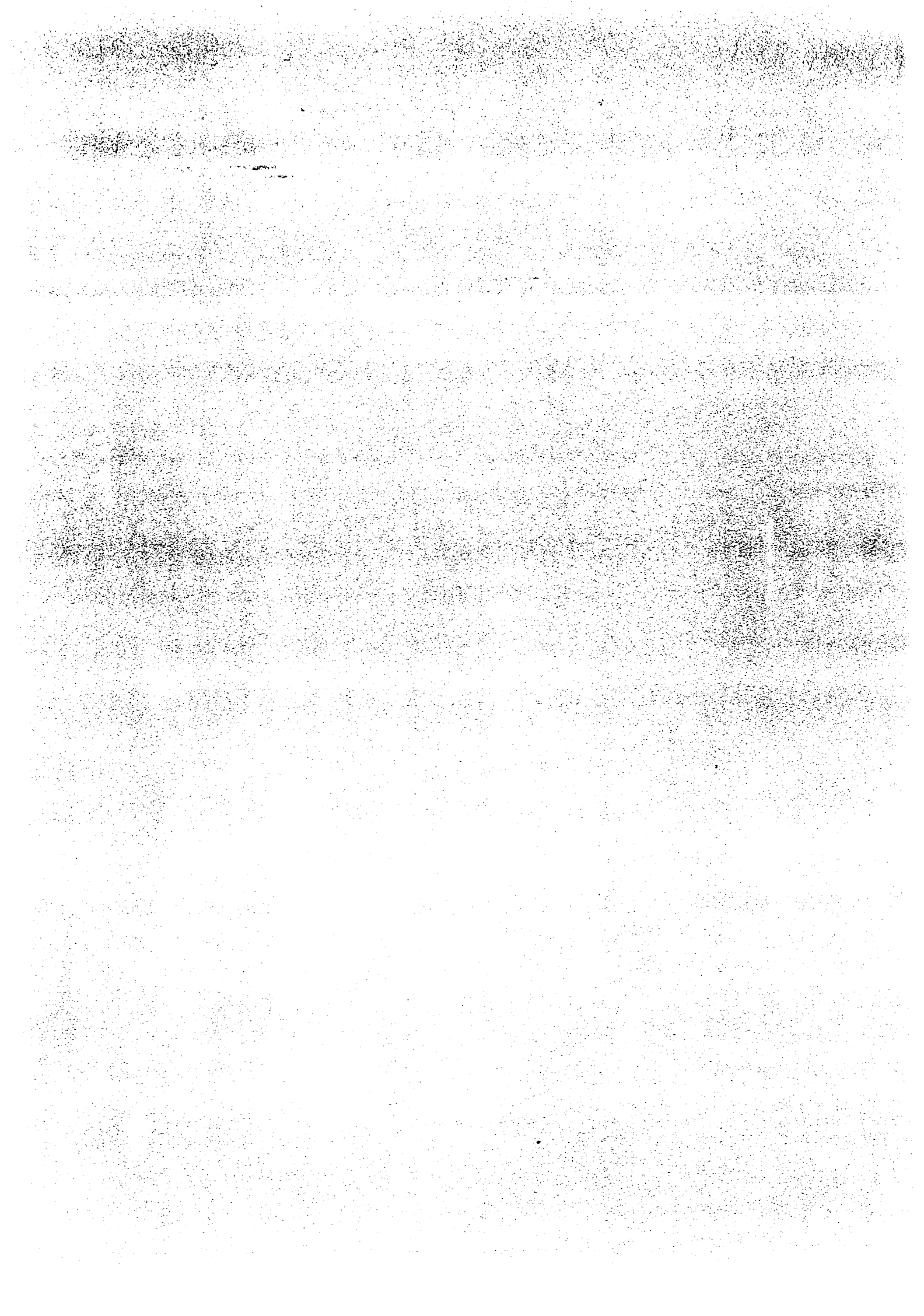


FRANCESC CONCELLAS, JOSEP FÀBREGA, AND PIERRE FRAIGNAUD
EDITORS

CARLETON SCIENTIFIC



Pattern Formation by Autonomous Robots without Chirality

PAOLA FLOCCHINI
University of Ottawa, Canada

GIUSEPPE PRENCIPE
Università di Pisa, Italy

NICOLA SANTORO
Carleton University, Canada

PETER WIDMAYER
ETH Zürich, Switzerland

Abstract

Consider a set of anonymous mobile robots, and the patterns they can collectively form in the plane. If each robot has a compass needle that indicates *North* but there is no agreement on East and West (i.e., no chirality), it is known that every arbitrary pattern given in input can be formed if and only if the number of robots is odd. In this paper we study what patterns can be formed by an even number of such robots, and provide a complete characterization. We identify the class of patterns which cannot be formed by an even number of anonymous robots without chirality, regardless of additional capabilities. We then construct a set of rules (distributed algorithm) which leads the robots to form any of the other patterns in finite time. The algorithm is collision-free, completely oblivious and fully asynchronous.

Keywords

Mobile computing, robots, distributed algorithms, oblivious computations, chirality.

1 Introduction

In this paper we consider a collection of identical *anonymous mobile robots*, and the patterns they can collectively form in the plane.

Following the current trend in robotics research, our robots are anonymous, have minimal capabilities, and exhibit an extremely simple behavior. Their actions consist in autonomously observing the environment, computing a destination point, and moving towards it. Their behavior is a simple continuous cycle of sensing, computing, moving and being inactive.

The ability to form geometric patterns is a fundamental one; it is usually an initial step which allows the robots to decide on their respective roles in a subsequent, coordinated action (e.g., forming a circle around an object which has to be removed, arranging themselves in a line so to sweep the terrain in formation, etc.).

Pattern formation by a collection of such robots has been studied by researchers in robotics and in artificial intelligence (e.g., see [1, 3, 4]). Concerns on computability and complexity of the problem have recently motivated also algorithmic investigations [2, 6, 7, 8]; the basic difference is that in [7, 8], any action by the robots, including moving, is *instantaneous*. In this paper, like in [2, 6], there is no such an assumption.

The capacity to solve the pattern formation problem seems to depend not only to the robots' (computing and motorial) capabilities but also and especially on the level of agreement they have on the coordinate system.

In fact, if the robots have *complete agreement* on a common coordinate system[1] (e.g., each robot has a compass), they can form *any* arbitrary pattern given in input; they can do so even if they are totally *asynchronous* in their actions (and inactions), and are completely *oblivious* (i.e., they do not remember any previous observation, computation, or action) [2]. On the other hand, if the robots have *no agreement* on the coordinate system, then the arbitrary pattern formation is *unsolvable* regardless of their capabilities (unbounded memory, instantaneous actions [2].

More interesting is the case when there is a *limited agreement* among the robots; that is, when they agree on the direction and orientation of one axis, but do not have a common understanding of the *handedness (chirality)* of the coordinate system[2]; thus knowing North does not distinguish East from West. In this case, an *odd* number of anonymous asynchronous oblivious robots can still form any arbitrary pattern given in input; on the other hand, if the number of robots is *even*, the arbitrary pattern formation problem is unsolvable regardless of their capabilities [2].

Since, with limited agreement, an even number of anonymous robots cannot form *every* pattern, the natural immediate questions are *what patterns, if any, can they form?, what capabilities they need to do so?, how can they do it?* In this paper we provide a complete constructive answer to these questions.

In fact, we first identify a class of patterns Ω that, with partial agreement, *cannot* be formed by an even number of anonymous robots, regardless of their capabilities. The impossibility result holds even allowing the robots to remember the past and to perform synchronous and instantaneous computations. We then present a set of rules (distributed algorithm) which, when executed (asyn-

[1] i.e., on the direction and orientation of the two axes but not necessarily on their origin nor on a common unit length.

[2] chirality allows to consistently infer the orientation of the Y axis once the orientation of the X axis is given.

chronously and independent
in finite time. The pattern fo
not require any knowledge
fully asynchronous (i.e., it
putation time of the robots).
the same position in the pla

2 Model and Bas

2.1 The Robots

We consider a system of a
sensing its immediate surro
and moving towards the co
of sensing, computing, mov

The robots are viewed
capabilities, which are able
sensors that let each robot
cal view of the world. This
will assume w.l.g. to be the
a Cartesian coordinate syst
axes, referred to as X and Y
on the two axes direction, b
be axis Y). Notice that such
distance nor on the location

The robots are *oblivio*
previous observation nor co

The robots are *anonymo*
their appearances, and they
be used during the comput
communication.

The robots are *fully as*
the amount of time spent i
inaction is finite but otherw

The robots execute the
the observed positions of t
destination point towards w

A robot is initially in a
dently from the other robo
its sensors which will retur

[3] i.e., activating the sensors and

[4] We do not require the robots t

chronously and independently) by the robots leads to form any input pattern $\mathbb{P} \notin \Omega$ in finite time. The pattern formation algorithm is *completely oblivious* (i.e., it does not require any knowledge by the robots of past computations or observations), *fully asynchronous* (i.e., it does not make any assumption on the speed or computation time of the robots), and *collision-free* (i.e., two robots will never occupy the same position in the plane).

2 Model and Basic Properties

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 sensing, computing, moving and being inactive.

The robots are viewed as points, and modeled as units with computational capabilities, which are able to freely move in the plane. They are equipped with sensors that let each robot observe the positions of the others and form its *local view* of the world. This view includes a unit of length, an origin (which we will assume w.l.g. to be the position of the robot in its current observation), and a Cartesian coordinate system (comprising the directions of the two coordinate axes, referred to as X and Y, and their orientations). We assume the robots agree on the two axes direction, but on the orientation of only one of them (w.l.g, let it be axis Y). Notice that such an agreement does not imply agreement on the unit distance nor on the location of the origin.

The robots are *oblivious*, meaning that they do not need to 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 need to 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[3], in computation, in movement, and in inaction is finite but otherwise unpredictable.

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 (*Look*) by activating its sensors which will return a snapshot[4] of the positions of all other robots with

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

[4] We do not require the robots to be able to detect *multiplicity* (i.e. whether there is more than one

respect to its local coordinate system. (Since robots are viewed as points, their positions in the plane is just the set of their coordinates). The robot then *calculates* its destination point (*Compute*) according to its deterministic, oblivious algorithm, based only on the observed locations of the robots. It then *moves* towards that point (*Move*); if the destination point is the current location, the robot stays still (*null movement*). After the move, the robot becomes waiting again. The sequence *Wait - Look - Compute - Move* forms a *cycle* of a robot.

There are two limiting assumptions excluding the *infinite*: 1. The amount of time required by a robot to complete a cycle is not infinite, nor infinitesimally small. 2. The distance traveled by a robot in a cycle is not infinite, nor infinitesimally small (unless it brings the robot to the destination point).

As no other assumptions 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. As no other assumptions on space exists, the distance traveled by a robot in a cycle is unpredictable.

2.2 The Pattern Formation Problem

We study the pattern formation problem for arbitrary geometric patterns, where a pattern $\mathbb{P}$ is a set of points $p_0, \ldots, p_n$ (given by their Cartesian coordinates) in the plane. The pattern is known initially by all robots in the system. Initially, the robots are in arbitrary positions, with the only requirement that no two robots be in the same position, and that, of course, the number of points prescribed in the pattern and the number of robots are the same.

Let a *configuration (of the robots)* at time t be a set of robots' positions at time t, one position per robot, with no position occupied by more than one robot. Given a pattern $\mathbb{P}$ and a configuration $\mathbb{C}$, let P_f^i be the set of positions of the robots as viewed in the local coordinate system of the robot r_i. The robots are said to *have formed* $\mathbb{P}$, if there exists a transformation $\mathcal{T}$, where $\mathcal{T}$ can be *translation*, *rotation*, *scaling*, or *flipping* into mirror position, such that, $\forall i$, $\mathcal{T}(P_f^i) = \mathbb{P}$. In other words, the final positions of the robots must coincide with the points of the input pattern, where the formed pattern may be *translated*, *rotated*, *scaled*, and *flipped* into its mirror position with respect to the input pattern $\mathbb{P}$ in each local coordinate system. A *final configuration* for $\mathbb{P}$ is a configuration of the robots in which the robots form the desired pattern $\mathbb{P}$. The *arbitrary pattern formation problem* is the problem of devising an algorithm which will lead the robots to a final configuration in a finite number of moves for any pattern and any configuration. Consider a set of patterns $\mathfrak{T}$. Given an arbitrary initial configuration of the robots and an arbitrary pattern $\mathbb{P} \in \mathfrak{T}$, a *pattern formation algorithm* for $\mathbb{P}$ is an oblivious deterministic algorithm that brings the robots in

robot on any of the observed points, included the position where the observing robot is).

the system to a final configuration f
a pattern formation algorithm is *coll*
no two robots that occupy the same

Another problem that we will r
problem: the robots in the system a
ber of cycles, all the robots determi
called the leader. A deterministic alg
a leader in a finite number of cycles
leader election algorithm.

In the following we will denote h
calligraphic letters (e.g., $\mathcal{C}$), we indi
ber of robots in a given region. With
plane, and by $|\cdot|$ the number of rob
some of the proofs will be omitted.

2.3 Basic Properties

First notice that two robots can trivia
is a segment). Therefore, we will a
the system.

Theorem 1 *[2] The arbitrary patt*
anonymous robots without chirality

Interestingly, the proof of the uns
(and lack of chirality) and not on ob
the arbitrary pattern formation alg
chronous. To better understand the
we will re-prove it in terms of leade

Theorem 2 *When n is even, there*
the leader election problem without

which yields Theorem 1 as a corolla

Since an arbitrary pattern can n
are now interested in determining w
this case. Thus, from now on, we w
system is even.

3 Unformable Pattern

In this section we describe a class
number of anonymous robots with

the system to a final configuration for $\mathbb{P}$ in a finite number of cycles. We say that a pattern formation algorithm is *collision-free*, if, at any point in time t, there are no two robots that occupy the same position in the plane at t.

Another problem that we will refer to in the following is the *leader election* problem: the robots in the system are said to elect a leader if, after a finite number of cycles, all the robots deterministically agree on (choose) the same robot l, called the leader. A deterministic algorithm that lets the robots in the system elect a leader in a finite number of cycles, starting from any configuration, is called a *leader election algorithm*.

In the following we will denote by $r.x$ and $r.y$ the coordinates of robot r. With calligraphic letters (e.g., $\mathcal{C}$), we indicate regions of the plane, and by $|\cdot|$ the number of robots in a given region. With capital letters (e.g., L) we indicate lines in the plane, and by $|\cdot|$ the number of robots on a given line. Due to space restrictions, some of the proofs will be omitted.

2.3 Basic Properties

First notice that two robots can trivially form any pattern (the only possible pattern is a segment). Therefore, we will assume $n > 2$, with n the number of robots in the system.

Theorem 1 *[2] The arbitrary pattern formation problem is solvable by $n > 2$ anonymous robots without chirality if and only if n is odd.*

Interestingly, the proof of the unsolvability for n even relies only on anonymity (and lack of chirality) and not on obliviousness or asynchrony. On the other hand, the arbitrary pattern formation algorithm for n odd is both oblivious and asynchronous. To better understand the negative result for an even number of robots, we will re-prove it in terms of leader-election.

Theorem 2 *When n is even, there exists no deterministic algorithm that solves the leader election problem without chirality.*

which yields Theorem 1 as a corollary.

Since an arbitrary pattern can not be formed by an even number of robots, we are now interested in determining which class of patterns, if any, can be formed in this case. Thus, from now on, we will assume that the number n of robots in the system is even.

3 Unformable Patterns

In this section we describe a class of pattern that cannot be formed by an even number of anonymous robots with limited agreement. A pattern $\mathbb{P}$ is *symmetric* if

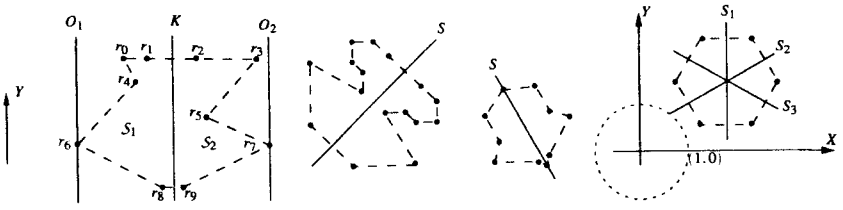


Figure 1: (a) An unachievable asymmetric pattern. In this example, the sorted sequence of pairs of robots from the proof of Lemma 3 is the following: (r_1,r_2), $(r_0,\emptyset)$, $(r_3,\emptyset)$, $(r_4,\emptyset)$, $(r_5,\emptyset)$, (r_6,r_7), (r_8,r_9). In this case r_0 would be elected as the leader. (b) An achievable pattern with one *empty axis*. (c) An unachievable pattern. (d) An achievable pattern that has three *empty axes*. Note that this pattern has also axes of symmetry passing through vertices. In this case, following the procedure in Footnote 6, the routine `Choose` $(\mathbb{P})$ of Algorithm 1 chooses S_2.

it has at least one axis of symmetry S; that is, for each $p \in \mathbb{P}$ there exists exactly another point $p' \in \mathbb{P}$ such that p and p' are symmetric with respect to S (see Figure 1.b, c and d).

The proof of the unsolvability result of Theorem 2 is useful to better understand which kind of patterns can not be formed, hence which kind of pattern formation algorithm can not be designed. In fact, the ability to form a particular type of patterns would imply the ability to elect a robot in the system as the leader. More formally,

Theorem 3 *There exists no pattern formation algorithm that lets the robots in the system form (a.) an asymmetric pattern, or (b.) a symmetric pattern that has all its axes of symmetry passing through a vertex.*

Proof.

Part a. By contradiction, let $\mathcal{A}$ be a pattern formation algorithm, and let $\mathbb{P}$ be an arbitrary asymmetric pattern of n points. We now show that $\mathcal{A}$ is a leader election algorithm. Let Ψ be the final configuration after they execute the algorithm, starting from an arbitrary initial configuration. Since the robots in the system agree on the direction and orientation of the Y axis, it is possible for them to elect a leader. In fact, let O_1 and O_2 be respectively the vertical axis passing through the outermost robots in Ψ (all the robots must agree on these two axis, since they agree on the orientation of Y), and let K be the vertical axis equidistant from O_1 and O_2 (e.g., see Figure 1.b). K splits the plane in two regions, S_1 and S_2. If some robots are on K, the highest on K can be elected as a l
can distinguish two cases:

1. $|S_1| \neq |S_2|$. In this case, the region as the positive side o figuration, it is possible to e one).
2. $|S_1| = |S_2|$. In this case, for e $x \in S_2 \bigcup \emptyset$, as follows. Let h $p_K(r)$ indicates the *proximit* distance between r and K. I $h(r_j)$, and $p_K(r_i) = p_K(r_j)$, th we build pairs for each $r_j \in S_2$ if (r_j,r_i) is defined, we can so respect to the height and the (e.g., see in Figure 1.b):

$$(r_i,\emptyset) > (r_j,\emptyset) \Leftrightarrow h(r$$
$$(h$$
$$(r_i,\emptyset) > (r_j,r_h) \Leftrightarrow h(r_i$$
$$(h$$
$$(r_i,r_j) > (r_h,\emptyset) \Leftrightarrow h(r_i$$
$$(h$$
$$(r_i,r_j) > (r_h,r_k) \Leftrightarrow h(r_i$$
$$(h$$

We observe that the set of pair entation of the X axis; moreo hypothesis, there must exist at l can elect as a leader the robot ir

Hence, $\mathcal{A}$ would be a leader election

Part b. By contradiction, suppose there e that lets the robots form a symmetric metry passing through some vertices configuration. After the robots run $\mathcal{A}$ whose positions correspond to the ver hence, Ψ must be symmetric with all some vertices (robots' positions). We

1. Ψ is not symmetric w.r.t. any a same argument of Part a. can be elected.

highest on K can be elected as a leader. Suppose that no robot is on K. We can distinguish two cases:

1. $|\mathcal{S}_1| \neq |\mathcal{S}_2|$. In this case, the robots can agree on the most populated region as the positive side of X; hence, starting from any initial configuration, it is possible to elect a leader (e.g., the topmost rightmost one).
2. $|\mathcal{S}_1| = |\mathcal{S}_2|$. In this case, for each robot $r_i \in \mathcal{S}_1$, we build a pair (r_i, x), $x \in \mathcal{S}_2 \bigcup \emptyset$, as follows. Let $h(r)$ indicates the height of robot r, and $p_K(r)$ indicates the *proximity* of robot r to K, that is the horizontal distance between r and K. If there exists $r_j \in \mathcal{S}_2$ such that $h(r_i) = h(r_j)$, and $p_K(r_i) = p_K(r_j)$, then $x = r_j$; otherwise $x = \emptyset$. Analogously, we build pairs for each $r_j \in \mathcal{S}_2$. Given that (r_i, r_j) is defined if and only if (r_j, r_i) is defined, we can sort all the pairs in descending order, with respect to the height and the proximity of the robots to K. Namely, (e.g., see in Figure 1.b):

$$
\begin{aligned}
(r_i,\emptyset) &> (r_j,\emptyset) \Leftrightarrow h(r_i) > h(r_j) \vee \\
&\qquad (h(r_i) = h(r_j) \wedge p_K(r_i) < p_K(r_j)) \\
(r_i,\emptyset) &> (r_j,r_h) \Leftrightarrow h(r_i) > h(r_j) \vee \\
&\qquad (h(r_i) = h(r_j) \wedge p_K(r_i) < p_K(r_j)) \\
(r_i,r_j) &> (r_h,\emptyset) \Leftrightarrow h(r_i) > h(r_h) \vee \\
&\qquad (h(r_i) = h(r_h) \wedge p_K(r_i) < p_K(r_h)) \\
(r_i,r_j) &> (r_h,r_k) \Leftrightarrow h(r_i) > h(r_h) \vee \\
&\qquad (h(r_i) = h(r_h) \wedge p_K(r_i) < p_K(r_h))
\end{aligned}
$$

We observe that the set of pairs obtained is independent from the orientation of the X axis; moreover, since Ψ is asymmetric w.r.t K by hypothesis, there must exist at least a pair with an $\emptyset$. It follows that we can elect as a leader the robot in the first pair that has $\emptyset$ as an element.

Hence, $\mathcal{A}$ would be a leader election algorithm, contradicting Theorem 2.

Part b. By contradiction, suppose there exists a pattern formation algorithm $\mathcal{A}$ that lets the robots form a symmetric pattern $\mathbb{P}$ that has all its axes of symmetry passing through some vertices in $\mathbb{P}$, starting from an arbitrary initial configuration. After the robots run $\mathcal{A}$, they are in a final configuration Ψ whose positions correspond to the vertices of $\mathbb{P}$ (up to scaling and rotation); hence, Ψ must be symmetric with all its axes of symmetry passing through some vertices (robots' positions). We distinguish two cases.

1. Ψ is not symmetric w.r.t. any axis Y' parallel to Y. In this case, the same argument of Part a. can be used to conclude that a leader can be elected.

2. Ψ is symmetric w.r.t. some Y' parallel to Y. Since by hypothesis Y' must pass through a vertex, a leader can be elected (e.g., the topmost robot on Y').

Hence, $\mathcal{A}$ would be a leader election algorithm, contradicting Theorem 2.

□

Let us call $\mathfrak{T}$ the class containing all the arbitrary patterns, and $\mathfrak{P} \subset \mathfrak{T}$ the class containing only patterns with at least one axis of symmetry not passing through any vertex (e.g., see Figures 1.b and 1.d); let us call *empty* such an axis. Theorem 3 states that if $\mathbb{P} \in \mathfrak{T} \setminus \mathfrak{P}$, then $\mathbb{P}$ can not be formed; hence, according to Part b. of the previous theorem, the only patterns that might be formed are symmetric ones with at least one *empty axis*. In the following, we prove that all these patterns can actually be formed. In particular, we present an algorithm that lets the robots form exactly these kind of patterns, if local rotation of the pattern is allowed.

4 Forming Formable Patterns

4.1 The Algorithm

In this section we present an algorithm that let the robots form symmetric patterns with at least one *empty axis*. The idea behind the algorithm is as follows. First, the robots compute locally an *empty axis* say S, of the input pattern $\mathbb{P}$, and then rotate $\mathbb{P}$ so that S is parallel to the common understanding of the orientation of Y. Second, the robots elect the two topmost outermost[5] robots in the observed robots' positions, $Outer_1$ and $Outer_2$. If this is not possible (i.e., all the robots are on the same vertical axis), the second topmost robot on this axis moves to its right, so that $Outer_1$ and $Outer_2$ can be correctly computed. Then, $Outer_1$ and $Outer_2$ move until they are at the same height: these two robots will never move again. When this happens, the vertical axis K is computed: it is the median between the vertical axis passing through the positions of $Outer_2$ and $Outer_2$; K splits the plane in two halves, called $Left$ and $Right$. At this point, it is possible to compute the set of final positions of the robots: these are the positions that the robots must occupy in order to correctly solve the problem. In order to compute these positions, $\mathbb{P}$ is scaled with respect to the distance between $Outer_1$ and $Outer_2$, and is translated with respect to the positions of these two robots. We note that, by definition of $\mathbb{P}$, the number of final positions in $Left$ and in $Right$ is $n/2$. The robots' positions in each half can be sorted using the ordering defined in the proof of Theorem 3. Thus, the first $n/2$ robot in $Left$ are directed towards final positions in $Left$, and the first $n/2$ robots in $Right$ towards final positions in $Right$. If in one half there are more robots than final destination, the *extra* robots are directed towards K.

[5] That is, among the outermost robots, the two topmost ones.

Once there are no *extra* robots neither in L directed, from the topmost to the bottomm do not have any robots on them yet.

The routine `Choose(`$\mathbb{P}$`)` locally choos input pattern $\mathbb{P}$; since this is a local operatio every robot can be made to choose the same

`Rotate(`$\mathbb{P}$`,`S`)` locally rotates $\mathbb{P}$ in suc S chosen with `Choose(`$\mathbb{P}$`)` is parallel to t performed clockwise.

`Pattern_Length(`$\mathbb{P}$`)` returns the hori local unit distance, measured as the distance axes tangent to $\mathbb{P}$.

As already stated previously, the algorithn most robots, so that the input pattern can be t them. Unfortunately, if all the robots are on robots can not be located. In Line 5, the algo lar, routine `Same_Vertical_Axis(`H`)` let right, so that there are exactly two outermost t

`Same_Vertical_Axis(`H`)`

```
d:=dist(top(H),bottom(H));
Case |H|
  •n
    If I Am The Second Topmost Robot (
5:    p := Point To My Right At Horizo
      Move(p).
    Else
      do_nothing().
  •n − 1
10:  r := Robot Not On H;
    If I Am r Then
      If I Am Not At Horizontal Distance
        p := Closest Point To Me At Hor
        Move(p).
15:   Else
        return.
    Else
      If r Is Not At Horizontal Distance d
        do_nothing.
20:   Else
        return.
```

[6] For instance, starting from the point (1,0) on the unit coordinate system, the first *empty axis* that is hit moving c orientation of the X axis), after having translated the *empty a* the origin. In the example depicted in Figure 1.e, the axis S_2

Once there are no *extra* robots neither in *Left* nor in *Right*, the robots on K are directed, from the topmost to the bottommost, towards the final destinations that do not have any robots on them yet.

The routine `Choose(`$\mathbb{P}$`)` locally chooses an *empty axis* of symmetry in the input pattern $\mathbb{P}$; since this is a local operation, and $\mathbb{P}$ is the same for all the robots, every robot can be made to choose the same axis of symmetry [6].

`Rotate(`$\mathbb{P}$`,`S`)` locally rotates $\mathbb{P}$ in such a way that the axis of symmetry S chosen with `Choose(`$\mathbb{P}$`)` is parallel to the Y axis. The rotation is (locally) performed clockwise.

`Pattern_Length(`$\mathbb{P}$`)` returns the horizontal length of $\mathbb{P}$ according to the local unit distance, measured as the distance between the two outermost vertical axes tangent to $\mathbb{P}$.

As already stated previously, the algorithm locates the two outermost and topmost robots, so that the input pattern can be translated and scaled with respect to them. Unfortunately, if all the robots are on the same vertical axis H, these two robots can not be located. In Line 5, the algorithm handles this case. In particular, routine `Same_Vertical_Axis(`H`)` let one of the robots on H move to its right, so that there are exactly two outermost topmost robots. Namely, we have

`Same_Vertical_Axis(`H`)`

d :=`dist(top(`H`),bottom(`H`));`
Case $|H|$
  • n
    If I Am The Second Topmost Robot On H **Then**
5: p := Point To My Right At Horizontal Distance d From K;
      `Move(`p`).`
    Else
      `do_nothing().`
  • $n-1$
10: r := Robot Not On H;
    If I Am r **Then**
      If I Am Not At Horizontal Distance d from H **Then**
        p := Closest Point To Me At Horizontal Distance d From H;
        `Move(`p`).`
15:    **Else**
        `return.`
    Else
      If r Is Not At Horizontal Distance d From H **Then**
        `do_nothing.`
20:    **Else**
        `return.`

[6] For instance, starting from the point $(1,0)$ on the unit circle centered in the origin of the local coordinate system, the first *empty axis* that is hit moving counterclockwise (according to the local orientation of the X axis), after having translated the *empty axes* in such a way that they pass through the origin. In the example depicted in Figure 1.e, the axis S_2 would be chosen.

Algorithm 1 One axis direction and orientation, n even

Input: An arbitrary pattern $\mathbb{P}$ described as a sequence of points $p_1, \ldots, p_n$, given in lexicographic order. $\mathbb{P}$ is symmetric and has at least one *empty axis*.

```
S := Choose(ℙ);
P := Rotate(ℙ,S);
P_Length := Pattern_Length(ℙ);
H := Leftmost Vertical Axis With More Robots On It;
5: If (|H| = n Or |H| = n − 1) Then
     Same_Vertical_Axis(H);
   (Outer1, Outer2) := Outer_Most();
   If Outer1.y ≠ Outer2.y Then
     Fix_Outermosts(Outer1, Outer2).
10: Else
     If I Am Outer1 Or Outer2 Then
       do_nothing();
     Else
       K := Median_Axis(Outer1, Outer2);
15:    Final_Positions := Find_Final_Positions(K,P,S,
                                P_Length, Outer1, Outer2);
       If I Am On One Of The Final_Positions Then
         do_nothing();
       (Left, Right) := Sides(K);
20:    If I Am In Left Or In Right Then
         MySide := My_Side(K);
         Free_Points := {Final_Positions in MySide with no robots on them};
         Free_Robots := {Robots' posit. in MySide not on Final_Positions};
         If Free_Points ≠ ∅ Then
25:        Go_To_Points(Free_Robots, Free_Points);
         Else
           Choose_On_K(Free_Robots, K); *I am a Free_Robots but there
                                          are no Free_Points in MySide*
       If I Am On K Then
30:    If There Are Robots In (Left ∪ Right) Not On Final_Destinations
       Then
         do_nothing().
       Else
         r :=Highest(K);
         p := Point In Final_Positions Closest To r With No Robot On It;
35:      Go_To_Points({r},{p}).
```


where Move(p) terminates the local c
it towards the point p, using a straigh
d between the topmost (returned by t
bottom(H)) robot on H is computc
there are exactly n robots on H, the s
right until it is at an horizontal distan
$|H| = n - 1$ forces all the other robots t

Outermost() returns the curren
Since there is no agreement on the orie
$Outer_1$ and $Outer_2$.

Then, the algorithm calls the routi
that moves $Outer_1$ and $Outer_2$ until they
all the other robots are not allowed to n

The function Median_Axis(*Out*
which is the median axis between the
$Outer_2$.

Find_Final_Positions($K, P,$
tes the scaling and the translation of $\mathbb{P}$
$Outer_2$. These positions are computed
empty axis S computed in Line 1; furth
the way it has been rotated in Line 2,
points in $\mathbb{P}$ that are symmetric with re
$Outer_1$ and $Outer_2$ (which are symmetr
of the input pattern is defined by ident
distance between $Outer_1$ and $Outer_2$ (
their final positions).

The routine Sides(K) returns t
robots currently lying in the two halves
ular, it returns respectively the robots'
K, according to the local orientation of
returns the half of the plane where the c

Go_To_Points(*Free_Robots, Fre*
bots that is closest to a point in *Free_*
to avoid collisions. In fact, if r were to
destination p, it might collide with a r
the routine appropriately chooses an i
the invariant that r is the robot in *Free*
Namely (refer to Figure 2.a),

```
Go_To_Points(Free_Robots, Free_P
  (r,p) := Minimum(Free_Robots,
  If I Am r Then
    If No Robots Is On The Line Pa
      Move(p).
```

where `Move(p)` terminates the local computation of the calling robot and moves it towards the point p, using a straight movement. In other words, the distance d between the topmost (returned by `top(H)` and the bottommost (returned by `bottom(H)`) robot on H is computed by `dist(top(H),bottom(H))`; if there are exactly n robots on H, the second topmost robot r on H moves to its right until it is at an horizontal distance d from K (while r is moving, the case $|H| = n - 1$ forces all the other robots to stay still until r is at distance d from K).

`Outermost()` returns the current topmost outermost robots in the world. Since there is no agreement on the orientation of the X axis, it returns two robots, $Outer_1$ and $Outer_2$.

Then, the algorithm calls the routine `Fix_Outermosts(`$Outer_1$`,` $Outer_2$`)`, that moves $Outer_1$ and $Outer_2$ until they reach the same height. Until this happens, all the other robots are not allowed to move.

The function `Median_Axis(`$Outer_1, Outer_2$`)` returns the vertical axis K, which is the median axis between the vertical axes passing through $Outer_1$ and $Outer_2$.

`Find_Final_Positions(`$K, P, S, Pattern\_Length, Outer_1, Outer_2$`)` executes the scaling and the translation of $\mathbb{P}$ with respect to the positions of $Outer_1$ and $Outer_2$. These positions are computed in the following way: K is viewed as the *empty axis* S computed in Line 1; furthermore, by definition of $\mathbb{P}$ and because of the way it has been rotated in Line 2, there are exactly two outermost topmost points in $\mathbb{P}$ that are symmetric with respect to S: these two points are viewed as $Outer_1$ and $Outer_2$ (which are symmetric with respect to K). The common scaling of the input pattern is defined by identifying $Pattern\_Length$ with the horizontal distance between $Outer_1$ and $Outer_2$ (that are at the same height and already in their final positions).

The routine `Sides(`K`)` returns two sets, each containing the positions of robots currently lying in the two halves in which the plane is split by K: in particular, it returns respectively the robots' positions on the *Left* and on the *Right* of K, according to the local orientation of the calling robot's X axis. `My_Side(`K`)` returns the half of the plane where the calling robot currently lies.

`Go_To_Points(`$Free\_Robots, Free\_Points$`)` chooses the robot r in $Free\_Robots$ that is closest to a point in $Free\_Points$, say p, and moves r in such a way to avoid collisions. In fact, if r were to move following a straight line towards its destination p, it might collide with a robot on this path. Should this be the case, the routine appropriately chooses an intermediate destination point mantaining the invariant that r is the robot in $Free\_Robots$ closest to a point in $Free\_Points$. Namely (refer to Figure 2.a),

```
Go_To_Points(Free_Robots, Free_Points)
  (r, p) := Minimum(Free_Robots, Free_Points);
  If I Am r Then
    If No Robots Is On The Line Passing Through r And p Then
      Move(p).
```

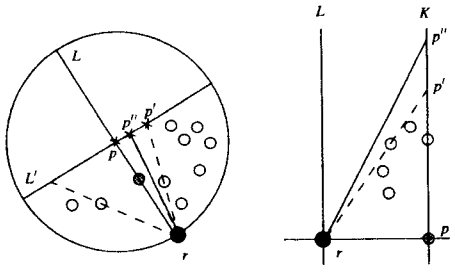


Figure 2: (a) Routine `Go_To_Points()` determines the destination point for r. The empty circles represent the points in *Avoid*, and the grey one represents the robot that does not allow r to perform a straight movement towards p; hence routine `Closest_Intersection()` is called in this case. The thick line is the path followed by r (in two cycles) to reach p. (b) Routine `Choose_On_K()` determines the destination point for r on K. The thick line is the path followed by r to reach K.

5: **Else**
  $L :=$ Line Passing Through r and p;
  $L' :=$ Line Orthogonal To L;
  $C_p :=$ Circle Centered In p Having Radius $\overline{rp}$;
  $C'_p :=$ Half Of C_p Delimited By L' That Contains r;
10: *Avoid* := Robots' Positions Inside C'_p;
  $p'' :=$ `Closest_Intersection(`$p, L'', Avoid, Free\_Points$`)`;
  If p'' Is Not Inside C_p **Then**
    $p'' :=$ Point On L' At Distance $\overline{rp}/2$ From p;{In this way, p'' is always chosen on L' and inside C_p}
15: `Move(`p''`)`.
Else
  `do_nothing()`.

where `Minimum(`$Free\_Robots, Free\_Points$`)` returns the pair (r, p) such that

$$\overline{rp} = \min_{\substack{f_r \in Free\_Robots \\ f_p \in Free\_Points}} \overline{f_r f_p}.$$

If more than one robot has minimum distance, the topmost robot that is closest to K is chosen. The core of this routine is in `Closest_Intersection()`, that looks for a point p'' on L' such that inside the triangle Δ passing through r, p, and p'' there are no robots; hence, r can move inside Δ avoiding collisions. Furthermore, `Closest_Intersection()` checks also that p'' is such that, while r moves towards it, p remains the point in *Free_Points* closest to r.

If in *MySide* there are more robo
directed towards K by invoking routin
ensures that the movements towards K
2.b). In particular,
`Choose_On_K(`$Free\_Robots, K$`)`
If I Am The Topmost And Closest T
  $p :=$ Intersection Between K And
  sition;
  If No Robot Is On The Line Passi
    `Move(`p`)`.
  Else
    $L :=$ Vertical Line Passing Thr
    $\mathcal{R} :=$ Portion Of The Plane Ab
    L;
    Avoid := {Robots' Positions In
    p};
    Intersections := $\emptyset$
    For All $p' \in$ *Avoid* **Do**
      $L' :=$ Line Passing Through
      Intersections := *Intersection*
    End For
    $p' :=$ Topmost Point In *Intersec*
    $p'' :=$ Point On K Above p' At
    `Move(`p''`)`.
Else
  `do_nothing()`.

In other words, the robot r chooses a p
in the region $\mathcal{R}$ delimited by L (the ve
the calling robot) and K, maintaining t
topmost robot in *Free_Robots*.

When all the robots in *Left* and *Righ*
K, if any, are directed sequentially, from
available final positions. In particular, th
point in *Final_Positions*.

4.2 Correctness

We shall call an *even agreement config*
on the position of a vertical axis K in t
of the *Final_Positions* or on K, and (iii)
Final_Positions. Moreover, we define th
figuration as the number of robots on K.
the two halves in which the plane is divi
lets the robots reach an even agreement c

If in *MySide* there are more robots than final positions, the *extra* robots are directed towards K by invoking routine `Choose_On_K()` in Line 27. This routine ensures that the movements towards K are done without collisions (refer to Figure 2.b). In particular,

`Choose_On_K` (*Free_Robots*, K)

If I Am The Topmost And Closest To K in *Free_Robots* **Then**
  p := Intersection Between K And Horizontal Line Passing Through My Position;
  If No Robot Is On The Line Passing Through My Position And p **Then**
    `Move(`p`)`.
  Else
    L := Vertical Line Passing Through My Position;
    $\mathcal{R}$:= Portion Of The Plane Above My Position And Delimited By K And L;
    Avoid := {Robots' Positions Inside $\mathcal{R}$} $\cup$ {Robots' Positions On K Above p};
    Intersections := $\emptyset$
    For All $p' \in$ *Avoid* **Do**
      L' := Line Passing Through My Position And p';
      Intersections := *Intersections* $\cup$ {Intersection Between K And L'};
    End For
    p' := Topmost Point In *Intersections*;
    p'' := Point On K Above p' At Distance δ From p'.
    `Move(`p''`)`.
Else
  `do_nothing()`.

In other words, the robot r chooses a path the goes above all the robots that are in the region $\mathcal{R}$ delimited by L (the vertical line passing through the position of the calling robot) and K, maintaining the invariant to remain the (closest to K) topmost robot in *Free_Robots*.

When all the robots in *Left* and *Right* are on *Final_Positions*, all the robots on K, if any, are directed sequentially, from the topmost to the bottomost, towards the available final positions. In particular, the topmost robot on K chooses the closest point in *Final_Positions*.

4.2 Correctness

We shall call an *even agreement configuration* one in which (i) all robots agree on the position of a vertical axis K in the plane, (ii) each robot is either on one of the *Final_Positions* or on K, and (iii) there is at most one robot on each of the *Final_Positions*. Moreover, we define the *cardinality* of an even agreement configuration as the number of robots on K. In the following we will denote by *sides* the two halves in which the plane is divided by K. We first show that Algorithm 1 lets the robots reach an even agreement configuration in a finite number of cycles,

while avoiding collisions between robots.

Lemma 1 *If the robots are not in a final configuration, in a finite number of cycles the robots agree on two topmost outermost robots, $Outer_1$ and $Outer_2$. Let O_1 and O_2 be the vertical axes passing through the positions of $Outer_1$ and $Outer_2$, respectively. In a finite number of cycles $Outer_1$ and $Outer_2$ will be at the same height, and all the robots will agree on the position of the vertical axis K that is median between O_1 and O_2. Until this happens, any collision is avoided.*

Proof. After having chosen an axis of symmetry S on Line 1 of the algorithm[7], the robot locally rotates $\mathbb{P}$ in such a way that S becomes parallel to Y. We note that, since $\mathbb{P}$ is symmetric with respect to S, and S is an *empty axis*, there are exactly two topmost outermost points in $\mathbb{P}$. If the robots are all on the same vertical axis H, routine `Same_Vertical_Axis(H)` is called. In particular, let d be the distance between the topmost and the bottomost robot on H. The second topmost robot r on H moves at a distance d to its right, while all the other robots do not move. While r is moving, $|H| = n - 1$ and, according to Line 19 of the routine, all the robots do not move until r, in a finite number of movements, is at distance d from K. After this, the two outermost and highest robots in the world are localized ($Outer_1$ and $Outer_2$ at Line 7), and until they are at the same height[8] all the other robots do not move (Lines 3 and 9 in routine `Fix_Outermosts()`). Once they reach the same height (Line 10), $Outer_1$ and $Outer_2$ will never move again (Line 12). By construction, $O_1 \neq O_2$, and there are no robots on O_1 and O_2 that are above $Outer_1$ or $Outer_2$. Therefore, since according to routine `Fix_Outermosts()` $Outer_1$ and $Outer_2$, if they move, always move up and on O_1 and O_2, their movements can not produce any collision. After $Outer_1$ and $Outer_2$ reach the same height, all the robots can agree on the axis K that is in the middle between the vertical axis passing through $Outer_1$ and the vertical axis passing through $Outer_2$ (Line 14). Since $Outer_1$ and $Outer_2$ executed a finite number of cycles to reach their final positions, by Assumptions A1 and A2, the agreement on K is reached in a finite number of cycles. □

Lemma 2 *If the robots are not in a final configuration nor in an even agreement configuration, they will reach an even agreement configuration in a finite number of cycles, and without collisions.*

Moreover, we can prove the following

Lemma 3 *Given an even agreement configuration of cardinality greater or equal to one, within finite time another even agreement configuration of smaller cardinality will be reached, avoiding any collisions.*

[7] All the *Lines* refer to Algorithm 1, if not otherwise stated.

[8] We recall that we can talk about *same heights* because all the robots agree on the direction of Y, hence they can commonly agree on this.

Therefore, by Lemmas 2 and 3, we ca

Theorem 4 *Algorithm 1 is a collision-fr* $\mathfrak{P}$.

Corollary 1 *An even number of autonc robots that agree on the direction and orie if and only if* $\mathbb{P} \in \mathfrak{P}$.

5 Remarks on Rotation

In Sections 3 and 4.1, we have provided terns which can be formed by an even nu they have agreement on the direction and terization assumes that the robots can local robots be incapable to perform this operatio surprisingly, the class of achievable patter of symmetric pattern with at least one *empt* to Y.

Theorem 5 *There exists no pattern formati rotation of the input pattern, and that lets* $\mathbb{P} \in \mathfrak{P}'$.

Proof. By contradiction, let $\mathcal{A}$ be an algo initial configuration, let the robots form a pa figuration of the robots for $\mathbb{P}$ after they exe lowed, also Ψ is symmetric with no *empty* the vertical axes passing through the two outc median between O_1 and O_2. If $K = O_1 = O_2$ leader can be elected (e.g., the topmost robc 2. Otherwise, if Ψ is symmetric with respect robot on K (by hypothesis, Ψ has no *empty a* of these robots can be elected as leader, con is not symmetric with respect to K: also in th following an approach similar to the one use contradicting again Theorem 2.

As a concluding remark, we note that ski Algorithm 1, we have a pattern formation a local rotation and allows the formation of a one *empty axis* that is parallel to Y. Hence, w

Therefore, by Lemmas 2 and 3, we can state the following

Theorem 4 *Algorithm 1 is a collision-free pattern formation algorithm for* $\mathbb{P} \in \mathfrak{P}$.

Corollary 1 *An even number of autonomous, anonymous, oblivious, mobile robots that agree on the direction and orientation of Y axis, can form a pattern* $\mathbb{P}$ *if and only if* $\mathbb{P} \in \mathfrak{P}$.

5 Remarks on Rotation

In Sections 3 and 4.1, we have provided a characterization of the class of patterns which can be formed by an even number of anonymous robots, provided they have agreement on the direction and orientation of the Y axis. This characterization assumes that the robots can locally *rotate* the input pattern. Should the robots be incapable to perform this operation, the characterization is different; not surprisingly, the class of achievable patterns is smaller. Let $\mathfrak{P}' \subset \mathfrak{P}$ be the class of symmetric pattern with at least one *empty axis*, and with no *empty axis* parallel to Y.

Theorem 5 *There exists no pattern formation algorithm that does not allow local rotation of the input pattern, and that lets the robots form a symmetric pattern* $\mathbb{P} \in \mathfrak{P}'$.

Proof. By contradiction, let $\mathcal{A}$ be an algorithm that, starting from an arbitrary initial configuration, let the robots form a pattern $\mathbb{P} \in \mathfrak{P}'$. Let Ψ be the final configuration of the robots for $\mathbb{P}$ after they execute $\mathcal{A}$. Since no local rotation is allowed, also Ψ is symmetric with no *empty axis* parallel to Y. Let O_1 and O_2 be the vertical axes passing through the two outermost robots, and K the vertical axis median between O_1 and O_2. If $K = O_1 = O_2$, then all the robot are on K, hence a leader can be elected (e.g., the topmost robot on K), thus contradicting Theorem 2. Otherwise, if Ψ is symmetric with respect to K, then there must be at least one robot on K (by hypothesis, Ψ has no *empty axis* parallel to Y); hence, the topmost of these robots can be elected as leader, contradicting Theorem 2. Therefore, Ψ is not symmetric with respect to K: also in this case a leader can be elected (e.g., following an approach similar to the one used in the proof of Theorem 3.a), thus contradicting again Theorem 2. □

As a concluding remark, we note that skipping the `Rotate(`$\mathbb{P}$`)` at Line 2 in Algorithm 1, we have a pattern formation algorithm that does not make use of local rotation and allows the formation of a symmetric pattern that has at least one *empty axis* that is parallel to Y. Hence, we can state the following

Corollary 2 *An even number of autonomous, anonymous, oblivious, mobile robots that agree on the direction and orientation of Y axis, can form a pattern* $\mathbb{P}$ *if and only if* $\mathbb{P} \in \mathfrak{P} \setminus \mathfrak{P}'$, *when no local rotation of* $\mathbb{P}$ *is allowed.*

Acknowledgements

This work has been partially supported by the Natural Science and Engineering Research Council of Canada, by the Swiss National Science Foundation, and by Nortel Networks.

References

[1] J. Desai, J. Ostrowski, and V. Kumar. Control of Formations for Multiple Robots. In *Proc. of 1998 IEEE Int. Conf. on Rob. and Autom.*

[2] 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. of ISAAC*, 1999.

[3] Y. Kawauchi, M. Inaba, and T. Fukuda. A Principle of Decision Making of Cellular Robotic System (CEBOT). In *Proc. 1993 IEEE Conference on Rob. and Autom.*, pages 833–838.

[4] M. J. Mataric. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. *From Animals to Animats 2*, pages 423–441, 1993.

[5] S. Murata, H. Kurokawa, and S. Kokaji. Self-Assembling Machine. In *Proc. 1994 IEEE Conference on Rob. and Autom.*, pages 441–448.

[6] G. Prencipe. *Distributed Control and Coordination of a Set of Autonomous Mobile Robots*. PhD thesis, Università di Pisa, 2001.

[7] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots - Formation and Agreement Problems. In *Proc. of SIROCCO*, pages 313–330, 1996.

[8] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *Siam J. Comput.*, 28(4):1347–1363, 1999.

Paola Flocchini is an Associate Professor at the School of Information, Technology and Engineering of the University of Ottawa. E-mail: flocchin@site.uottawa.ca

Giuseppe Prencipe is a Ph.D. student at the Computer Science Department of the University of Pisa. E-mail: prencipe@di.unipi.it

Nicola Santoro is a Full Professor at the School of Computer Science of the Carleton University. E-mail: santoro@scs.carleton.ca

Peter Widmayer is a Full Professor at the Theoretische Informatik of the ETH Zürich. E-mail: pw@inf.ethz.ch

Characterization of N
Multi-dimensional Lin
Scher

YASHAR G
University of Wate

Abstra

An *Interval routing scheme (IRS)* is a strategy for routing messages in a distrib node of the network is assigned an integer labeled with an interval. The interval assig the set of destination addresses of the me through e at v. A Multi-dimensional Inte generalization of IRS in which each nod label (which is a list of d integers for th assigned to the links of the network are a 1-dimensional intervals). The class of n which the intervals are not cyclic) is alre case [5]. In this paper, we generalize this r the class of networks supporting linear MI ber of dimensions d. We show that by inc supporting MLIRS is strictly expanded. W the class of networks supporting strict MLI the intervals assigned to the links incident label of v).

Keywords

Computer networks, interval routing schemes characterizati

1 Introduction

One of the most fundamental tasks in any net sages between pairs of nodes. The classical m a network is to store a *routing table* at each n

*This work has been supported in part by NSERC.