Università di Pisa
Dipartimento di Informatica

Technical Report: TR-00–11

Achievable Patterns by an Even Number of Autonomous Mobile Robots

Giuseppe Prencipe
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40, 56100 - Pisa, Italy
e-mail: prencipe@di.unipi.it

August 17, 2000

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 887111 FAX: +39 050 887226

Achievable Patterns by an Even Number of Autonomous Mobile Robots

Giuseppe Prencipe
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40, 56100 - Pisa, Italy
e-mail: `prencipe@di.unipi.it`

August 17, 2000

Abstract

The distributed coordination and control of a set of autonomous, mobile agents/robots is a widely studied topic in a variety of fields, such as engineering, artificial intelligence, and artificial life. In the majority of the studies, the problem has been approached from an empirical point of view. A different approach to the problem has been introduced in [6], where the authors analyze the distributed coordination and control of a set of autonomous, mobile robots from a computational point of view. With this purpose, they defined a model where the world was inhabited by a set of totally autonomous, mobile, memoryless entities that were requested to form a generic pattern. From that study, it turned out that whether or not such a task could be accomplished depended on the amount of common knowledge the robots had; in particular, on the direction and orientation of the local coordinate system. Among the issues left open, was which kind of patterns a set of such mobile units can form when they are even in number and agree on the orientation and direction of only one axis. This paper answers that question.

1 Introduction

In this paper, we deal with the problem of coordinating and controlling a set of *autonomous*, *anonymous*, *memoryless* robots that asynchronously move on a plane. They all execute the same deterministic algorithm and are required to accomplish some predetermined task, specifically to form a given pattern. To study such a set of mobile units, we adopt the model first introduced by Flocchini *et al.* in [6], in which the authors study the problem from a computational point of view. That is, they try to understand under what conditions the set of robots can accomplish the given task, and what kind of capabilities they must have. This approach to the problem is quite different from that taken in typical studies that can be found in the engineering or artificial intelligence fields. Flocchini *et al.* defined a model, later called CORDA in [12], in which the robots were *weak* mobile units: they have no central control, they move independently from each other, they do not have any means of direct communication, and no capability to remember the past (the model will be described in more detail in Section 3). Motivations on the use of such a model and on the utility of such a study can be found in [6, 7].

In [6] the *arbitrary pattern formation* problem was studied. From that study, it was shown that whether or not an arbitrary pattern could be formed depended on the amount of (the initial) common knowledge the robots have, regarding the direction and orientation of the local coordinate system. Among the issues left open, there was the following: what kind of pattern can be formed by an even number of robots? In fact, in that paper the following negative result was proven:

Theorem 1.1 ([6]). *In a system with n anonymous robots that agree only on one axis direction and orientation, the arbitrary pattern formation problem is unsolvable when n is even.*

The above theorem states that an even number of robots cannot form an arbitrary pattern; that is, that there are certain patterns which they cannot form. Informally, a general pattern can not be formed by an even number of units, because there can be an initial symmetric configuration (the definition of *configuration* will be given below) that can not be broken deterministically. Flocchini *et al.* [6] did not, however, explore which patterns or class of patterns –if any– *could* be formed by a set of even-numbered robots, notwithstanding the possible symmetric configuration the robots can be in at the beginning. This is a question that is addressed instead in this paper.

In Section 2 a brief overview on the related work is given. In Section 3 the formal definition of the model is presented. In Section 4 the problem under study is stated and an oblivious algorithm that lets the robots solve it is provided. Finally, in Section 5 we draw some conclusions.

2 Related work

In recent years, interest in the problem of coordinating and controlling a set of autonomous, mobile robots has increased considerably. The reason for this growing interest is the shift from approaching the problem using a group of few, powerful units, to using a group of many, not-so-powerful ones. The advantages of using the latter approach are many: the simpler units are less expensive; there is an increase in fault tolerance; and the tasks can be completed more quickly. Among the interesting studies that have been done in a variety of fields, we can cite, in the engineering area, the Cellular Robotic System (CEBOT) of Kawaguchi *et al.* [8], the Swarm Intelligence of Beni *et al.* [3], and the Self-Assembly Machine ("fructum") of Murata *et al.* [10]. A remarkable effort on the study of the problem has been conducted also in the AI area, analyzing, for example, social interaction leading to group behavior [9]; selfish behavior of cooperative robots in animal societies [11]; or primitive animal behavior in pattern formation [2] (for a survey, refer to [4]).

Most of these studies, however, did not focus on algorithmic aspects of the problem [7]. One which did have, however, an algorithmic flavor was by Durfee [5], who suggested to limit the knowledge that a robot must possess in order to better coordinate its behavior with others. However, the main studies, to our knowledge, that aim to better understand the power and the limitations of the distributed coordination and control of a set of autonomous, mobile robots from an algorithmic and computational point of view, are by Suzuki and Yamashita [1, 13, 14], and by Flocchini *et al.* [6, 7]. In order to analyze the problem in this light, they present formal models focused on the understanding of the algorithmics of several problems, in particular pattern formation, under several assumptions on the power of the individual robot. The main objective of these studies is to highlight which kind of capabilities/features (available amount of memory, need or kind of common knowledge, sensors' power) the units must have in order to efficiently coordinate and reach some common goal they are required to achieve. In both [1, 14] and [6, 7], are defined models to formally describe the mobile units and the environment in which they operate. There are, however, substantial differences between the two models, which are analyzed in depth in [12]. In this paper we will study the pattern formation problem under the model of [6, 12], that will be described in detail in the next section.

3 The model

In this section, we describe the model that will be used in this paper, [6, 12]. The robots we consider are: *homogeneous* (they all follow the same set of rules); *autonomous* (there is no a priori central authority, and each robot's computing capabilities are independent from the others'); *asynchronous* (there is no central clock, no a priori synchronization, no a priori bounds on processing or motorial speed); *mobile* (robots are allowed to move on a plane); *anonymous* (they are a priori indistinguishable); and *oblivious* (they do not explicitly remember the past). Moreover, there are no explicit direct means of communication: the communication occurs in a totally implicit manner, through the modification to the environment, namely through the change of robots' positions in the plane during their movements.

These assumptions make the robots simple and rather "weak" in light of current engineering technology. But, as already noted, the interest in this study is to approach the problem from a *computational* point of view; in fact, by assuming the "weakest" robots, it is possible to analyze the strengths and weaknesses of the distributed control, and better understand which kind of power (limitations) such robots have in the totally asynchronous environment in which they operate. Furthermore, this simplicity can also lead to some advantages. For example, the lack of ability to remember what has been computed in the past, gives to the system the nice property of self-stabilization [7].

Each robot has its own *local view* of the world. This view includes a local Cartesian coordinate system with: origin (that we assume without loss of generality placed in the current position of the robot); unit of length; and the *directions* of two coordinate axes, identified as X axis and Y axis, together with their *orientations*, identified as the positive and negative sides of the axes. Notice, however, that the local views could be totally different making it impossible for the robots to reach an agreement on directions or on distances.

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 (*Observe*), 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. A computational *cycle* is defined as the sequence of the *Wait-Observe-Compute-Move* actions; the "life" of a robot is then a sequence of computational cycles. More formally, a cycle is constituted by the following four *phases*:

1. **Wait** The robot is idle. A robot cannot stay infinitely idle.

2. **Observe** The robot observes the environment by taking a snapshot of all other robots' positions with respect to its local coordinate system. Each robot r is viewed as a point, and therefore its position in the plane is given by its coordinates, indicated as the pair $(r.x, r.y)$. The *observation* returns a set of robots' positions (i.e. points) to

the observing robot. In addition, the robot cannot in general detect whether there is more than one fellow robot on any of the observed points; we say, it cannot detect *multiplicity.* Two different models can arise depending on whether we assume that a robot can see all the other robots in the system (called *Unlimited Visibility* model) or that a robot can see only the robots that are at most at some fixed distance from it (*Limited Visibility* model). In this paper we will deal with the Unlimited Visibility model.

3. **Compute** The robot performs a *local computation* according to its algorithm. We assume that the algorithm is deterministic, and takes in input only the observed set of points: we say that the algorithm is *oblivious.* The algorithm is said to be *non oblivious*, if it takes in input also the positions of the robots observed in the past; in this paper, we will deal exclusively with oblivious algorithms[1]. The result of the computation can be a destination point or `do_nothing()`.

4. **Move** As a result of the computation, the robot either stands still (`do_nothing()` was the result of the computation), or it moves (along any curve it likes, towards the point computed in the previous phase). The robot moves towards the computed destination of an unpredictable amount of space, which we assume neither infinite, nor infinitesimally small (see Assumption A2 below). Hence, the robot can only go towards its goal along a curve, but it cannot know how far it will go in the current cycle, because it can stop anytime during its movement.

Notice that, because of the obliviousness assumption, both the result of the computation and that of the observation phase will not be available to a robot at its next computational cycle.

In addition, we have the following assumptions on the behavior of a robot:

A1 (No Infinite Sleep) A robot can not Wait indefinitely.

A2 (Minimal Step) There is a lower bound $\delta_r > 0$ on the distance a robot r can travel, unless its destination is closer than δ_r. In this case it will reach its destination point in one cycle.

The environment in which the robots operate is totally *asynchronous*, in the sense that there is no common notion of time, and a robot observes its fellows at unpredictable time instants. Moreover, it is not made any assumption on how much a cycle of each robot lasts, neither on the time elapsed by a robot to execute one of the four phases of each cycle (for the sake of uniformity, we use this approach also with the Observe phase, even if its result is a snapshot of all the robots' positions). It is only assumed that each cycle is completed in finite time, and that the distance traveled in a cycle is finite; moreover, the distance is not infinitesimally small (unless the robot reaches its destination).

Notice that since other robots could obviously move during the computing phase of a given robot, the movement of this could be based on a past situation which is not valid anymore at the moment of the actual move.

[1]We refer also to the robots as *oblivious*, because of this feature of the algorithms they execute.

4 The Problem and its solution

In this paper, we concentrate on the particular coordination problem that requires the robots to form a specific geometric pattern, the *pattern formation* problem. In particular, we answer the following question: given an even number of robots, what kind of patterns are achievable in a finite number of cycles when there is *common knowledge* only on the direction and orientation of one axis[2], say Y?

We study this problem for arbitrary geometric patterns, where a pattern is a set of points (given by their Cartesian coordinates) in the plane. The pattern is known initially by all robots in the system. For instance, we might require the robots to place themselves on the circumference of a circle, with equal spacing between any two adjacent robots, just like kids in the kindergarten are sometimes requested to do. We do not prescribe the position of the circle in the world, and we do not prescribe the size of the circle, just because the robots do not have a notion of the world coordinate system's origin or unit of length.

All the robots are given in input the same pattern P, and they are required *to form* it in finite time. We call P a *symmetric pattern* if it has at least one axis of symmetry S, that is, for each $p \in P$ there exists exactly another point $p' \in P$ such that p and p' are symmetric with respect to S (see Figure 1.c, d and e). Let P^i_f be the set of final positions of the robots as viewed in the local coordinate system of the robot r_i. The robots are said to *form the pattern*, if there exists a transformation $\mathcal{T}$, where $\mathcal{T}$ can be *translation*, *rotation*, *scaling*, or *flipping* into mirror position, such that $\mathcal{T}(P^i_f) = P$, $\forall i$. 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 P in each local coordinate system. Initially, the robots are in arbitrary positions, with the only requirement that no two robots are in the same position, and that, of course, the number of points prescribed in the pattern and the number of robots are the same. The only means for the robots to coordinate is the observation of the others' positions; therefore, the only means for a robot to send information to some other robot is to move and let the others observe. For oblivious robots, even this sending of information is impossible, since the others will not remember previous positions.

This problem has been investigated quite a bit in the literature, mostly as an initial step that gets the robots together and then lets them proceed in the desired formation; it is interesting algorithmically, because if the robots can form any pattern, they can agree on their respective roles in a subsequent, coordinated action. It includes, as special cases, many coordination problems, such as leader election: we just define the pattern in such a way that the leader is represented uniquely by one point in the pattern.

We know from [6] that an arbitrary pattern can not be formed by an even number of robots (Theorem 1.1). Therefore, we want to find out which class of patterns can be formed in this case, if any. From now on, we will assume that the robots in the system have common knowledge on the direction and orientation of only the Y axis[3], and that the number n of robots in the system is even. We are ready now to introduce some definitions that will be useful in the following.

Definition 4.1. Let a *configuration (of the robots)* be a set of robot positions, one position

[2]By "common knowledge on the direction and orientation of Y", we mean that at the beginning "Everyone knows that everyone knows that ... that everyone knows what is the direction and the positive orientation of Y".

[3]This implies that there is common knowledge also on the direction of the X axis, but not on its orientation.

per robot, with no position occupied by more than one robot. A *final configuration* is a configuration of the robots in which the robots form the desired pattern.

Definition 4.2 (Pattern Formation Algorithm). A *pattern formation algorithm* is an oblivious deterministic algorithm that brings the robots in the system to a final configuration in a finite number of cycles, independently from the initial configuration.

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. Similarly to the pattern formation algorithm, we can define

Definition 4.3 (Leader Election Algorithm). An oblivious deterministic algorithm that lets the robots in the system elect a leader in a finite number of cycles, given any initial configuration, is called a *leader election algorithm.*

The following lemma states the unsolvability of the leader election problem under the assumptions under study.

Lemma 4.1. *There exists no deterministic algorithm that solves the leader election problem, when n is even.*

Proof. For the sake of contradiction, let $\mathcal{A}$ be a leader election algorithm. Now consider a particular initial situation of the robots in which, intuitively speaking, each robot has a symmetric partner with respect to the Y axis. More precisely, assume that initially, for every robot r there exists a symmetric partner robot $\widehat{r}$ such that the directions of the X axis of r and the X axis of $\widehat{r}$ are opposite, and the view of the world is the same for r and $\widehat{r}$ (see Figure 1.a). Now, for any move that r can make in its local coordinate system by executing algorithm $\mathcal{A}$, we know that $\widehat{r}$ can make the same move in its local coordinate system. If both of them move in the exact same way at the same time, they again end up in symmetric positions. Therefore, by letting all the robots move at the same time, with symmetric partners moving in the same way, we always proceed from one symmetric situation to the next, hence each robot will always have a partner that is indistinguishable from it. This never leads to a situation where the robots can deterministically agree on a unique robot, hence the leader cannot be elected. $\square$

The unsolvability result of Lemma 4.1 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 some kind of patterns would imply the ability to elect a robot in the system as the leader. More formally,

Lemma 4.2. *There exists no pattern formation algorithm that lets the robots in the system form*

a. *an asymmetric pattern (refer to Figure 1.b), or*

b. *a symmetric pattern that has all its axes of symmetry passing through a vertex (refer to Figure 1.c).*

Proof.

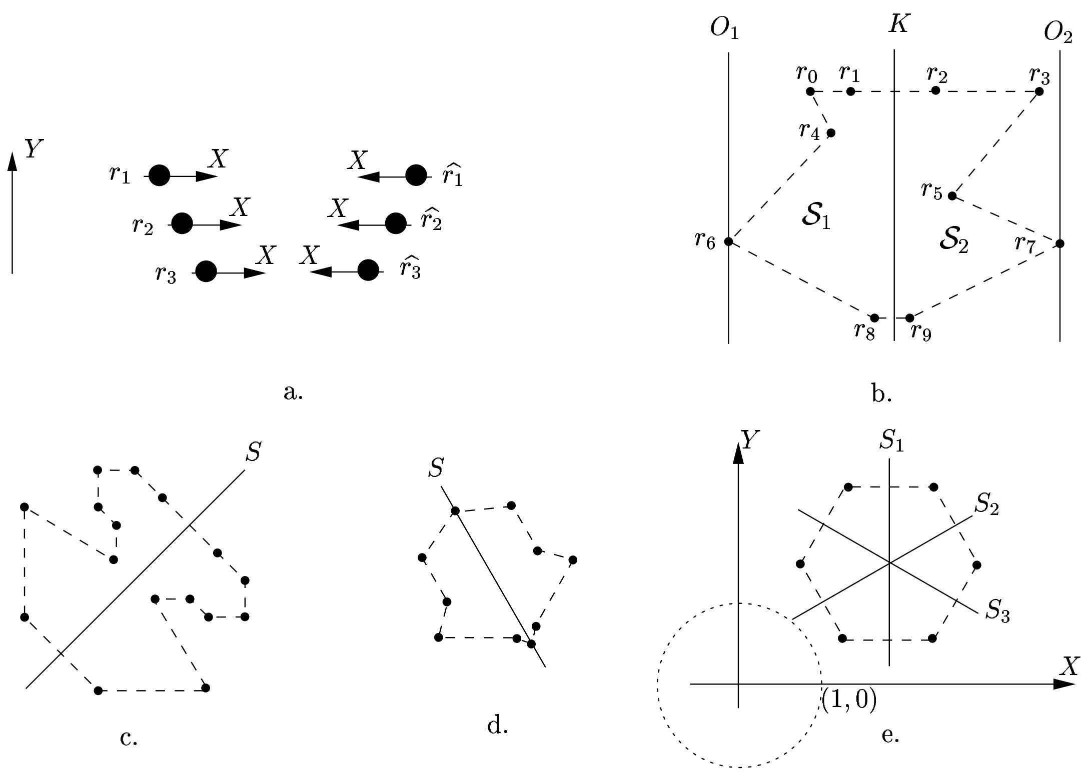

Figure 1: (a) Proof of Lemma 4.1: each r_i has the same view of the world of $\widehat{r_i}$, for $i = 1, 2, 3$. (b) An unachievable asymmetric pattern. In this example, the sorted sequence of pairs of robots from the proof of Lemma 4.2 is the following: (r_1, r_2), $(r_0, \emptyset)$, $(r_3, \emptyset)$, $(r_4, \emptyset)$, $(r_5, \emptyset)$, (r_6, r_7), (r_8, r_9). In this case r_0 would be elected as the leader. (c) An achievable pattern with one axis of symmetry not passing through any vertex. (d) An unachievable pattern. (e) An achievable pattern that has three axes of symmetry not passing through any vertex. Note that this pattern has also axes of symmetry passing through vertices. In this case, the routine `Choose(`P`)` of Algorithm 1 would choose the axis S_2.

Part a. Let $\mathcal{A}$ be a pattern formation algorithm, and let P be a specific asymmetric pattern of n points. The task for the robots is to form P by executing $\mathcal{A}$. Let $\mathcal{F}$ be the final configuration after they execute the algorithm. 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 $\mathcal{F}$ (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 (see Figure 1.b). K splits the plane in two regions, $\mathcal{S}_1$ and $\mathcal{S}_2$[4]. If some robots are on K, the highest on K can be elected as a leader, thus contradicting Lemma 4.1. So, let us suppose that no robot is on K. We can distinguish two cases:

1. $|\mathcal{S}_1| \neq |\mathcal{S}_2|$, the robots can agree on the most populated region as the positive side of X, hence it is now possible to elect as a leader the lexicographically smallest

[4]The names given to the regions is only for the sake of the proof. It can be different among the robots, since there is no agreement on the orientation of the X axis.

robot, for instance the topmost rightmost one, again contradicting Lemma 4.1.

2. $|\mathcal{S}_1| = |\mathcal{S}_2|$, for each robot $r_i \in \mathcal{S}_1$, we build a pair in the following way: we have (r_i, r_j), if there exists $r_j \in \mathcal{S}_2$ such that $h(r_i) = h(r_j)$, where $h(r)$ indicates the height of robot r, and $p_K(r_i) = p_K(r_j)$, where $p_K(r)$ indicates the *proximity* of robot r to K, that is the horizontal distance between r and K; otherwise we simply have $(r_i, \emptyset)$. Analogously, we build pairs for each $r_j \in \mathcal{S}_2$. Given that $(r_i, r_j) \equiv (r_j, r_i), \forall r_i \in \mathcal{S}_1, r_j \in \mathcal{S}_2$, we then sort in descending order all the pairs, with respect to the height and the proximity of the robots to K. More formally (see the example in Figure 1.b):

$$\begin{array}{ll}(r_i, \emptyset) > (r_j, \emptyset), \; if & h(r_i) > h(r_j) \vee (h(r_i) = h(r_j) \wedge p_K(r_i) < p_K(r_j)) \\ (r_i, \emptyset) > (r_j, r_h), if & h(r_i) > h(r_j) \vee (h(r_i) = h(r_j) \wedge p_K(r_i) < p_K(r_j)) \\ (r_i, r_j) > (r_h, \emptyset), \; if & h(r_i) > h(r_h) \vee (h(r_i) = h(r_h) \wedge p_K(r_i) < p_K(r_h)) \\ (r_i, r_j) > (r_h, r_k), if & h(r_i) > h(r_h) \vee (h(r_i) = h(r_h) \wedge p_K(r_i) < p_K(r_h))\end{array}$$

We observe that the set of pairs obtained is independent from the orientation of the X axis. Moreover, since it is hypothesized that $\mathcal{F}$ is asymmetric w.r.t K, there must exist at least a pair with an $\emptyset$. We can elect as a leader the robot in the first pair that has $\emptyset$ as an element. Therefore, $\mathcal{A}$ would be a leader election algorithm. This is a contradiction by Lemma 4.1.

Part b. Let us suppose there exists a pattern formation algorithm $\mathcal{A}$ that lets the robots form a symmetric pattern P that has all its axes of symmetry passing through some vertex in P. Therefore, after the robots run $\mathcal{A}$, they are in a final configuration $\mathcal{F}$ whose positions correspond to the vertices of P (apart from scaling and rotation), hence also $\mathcal{F}$ must be symmetric with all its axes of symmetry passing through some vertex (robot's position). We distinguish two cases.

$\mathcal{F}$ **is not symmetric w.r.t. any axis** Y' parallel to Y. By using an argument similar to that used in the previous Part a., we can conclude that a leader can be elected.

$\mathcal{F}$ **is symmetric w.r.t. some** Y' parallel to Y. Since we know that Y' must pass through a vertex, and that the positions of the robots are all distinct, we must have an unique topmost robot on Y', that can be elected as leader.

□

According to Part b. of the previous lemma, the only patterns that can be formed are symmetric ones with at least one axis of symmetry not passing through any vertex (see Figure 1.c and 1.e). In the following we describe an algorithm that lets the robots form these kind of patterns.

Algorithm 1 (One axis direction and orientation, n even).

Input: An arbitrary pattern P described as a sequence of points $p_1, \ldots, p_n$, given in lexicographic order. P is symmetric and has at least one axis of symmetry not passing through any vertex of P. The direction and orientation of the Y axis is common knowledge.

```
$S$ := Choose($P$);
$P$ := Rotate($P, S$);
$P\_Length$ := Pattern_Length($P$);
($Outer_1, Outer_2$) := Outer_Most();
If $Outer_1.y > Outer_2.y$ Then
  If I Am Not $Outer_2$ Then
    do_nothing();
  Else
    $p := (Outer_2.x, Outer_1.y)$;
    Move($p$);
If $Outer_1.y < Outer_2.y$ Then
  If I Am Not $Outer_1$ Then
    do_nothing();
  Else
    $p := (Outer_1.x, Outer_2.y)$;
    Move($p$);
If $Outer_1.y = Outer_2.y$ Then
  If I Am $Outer_1$ Or $Outer_2$ Then
    do_nothing();
  Else
    $K$ := Median_Axis($Outer_1, Outer_2$);
    If Some Robot Is On $K$ Then
      If I Am On $K$ Then
        $r$ :=Highest($K$);
        If I am $r$ Then
          ($Left, Right$) := Sides($K$);
          Move_Towards(Smallest($Left, Right$));
        Else
          do_nothing();
      Else
        do_nothing();
    Else
      ($My\_Side, Other\_Side$) := Sides($K$);
      If $|My\_Side| > |Other\_Side|$ Then
        $Close\_High$ := Closest($K, My\_Side$);
        If I Am $Close\_High$ Then
          Move_Towards($Other\_Side$);
        Else
          do_nothing();
      If $|My\_Side| < |Other\_Side|$ Then
        do_nothing();
      If $|My\_Side| = |Other\_Side|$ Then
        $Final\_Positions$ := Find_Final_Positions($K, P, S, My\_Side$,
                                  $P\_Length, Outer_1, Outer_2$);
        $Free\_Points$ := {$Final\_Positions$ in $My\_Side$ with no robots on them};
        If I Am On One Of The $Final\_Positions$ Then
          do_nothing()
        Else
        $Free\_Robots := \{Robots\ in\ My\_Side\ not\ on\ Final\_Positions\}$;
        Go_To_Points($Free\_Robots, Free\_Points$)
```

The routine Choose(P) locally chooses the "smallest" axis of symmetry in the input pattern P. Since this is a local operation, and P is the same for all the robots, every robot will choose the same axis of symmetry[5].

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

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

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

Move($\cdot$) and Move_Towards($\cdot$) are the two routines that allow a robot to move towards some destination. In particular, Move(p) terminates the local computation of the calling robot and moves it towards the point p, using a straight movement. Move_Towards($\cdot$) is executed to let a robot move towards the specific region passed as parameter. In this case, we assume that the robot moves horizontally towards the specified region, stopping if there is some other robot on the way.

The function Median_Axis(r_1, r_2) returns the vertical axis K, which is the median axis between the vertical axes passing through r_1 and r_2. The routine Sides(K) returns two sets, each containing the robots currently lying in the two sides in which the plane is split by K: if the calling robot is on K, it returns respectively the robots on the $Left$ and on the $Right$ of K, according to the local orientation of the calling robot's X axis; otherwise, it returns two sets, My_Side and $Other\_Side$, corresponding respectively to the robots lying on the half of the plane where the calling robot currently lies and the number of robots on the opposite half of the plane. By $|My\_Side|$ ($|Other\_Side|$) we indicate the number of robots currently lying in the region My_Side ($Other\_Side$). Smallest($\cdot$) returns the region, among the ones passed as parameter, with the lesser number of robots currently lying inside it. Closest(K, My_Side) returns the topmost robot that lies on My_Side and that is closest to K.

Find_Final_Positions ($K, P, S, My\_Side, Pattern\_Length, Outer_1, Outer2$) returns the set of final positions lying on the side of the plane where the calling robot currently lies, My_Side. These positions are computed in the following way: K is viewed as the axis of symmetry S computed in line 1; furthermore, because of the way P has been rotated in line 2, there are exactly two outermost topmost points in 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).

Go_To_Points($Free\_Robots, Free\_Points$) chooses the robot in $Free\_Robots$ that is closest to a point in $Free\_Points$, and moves it as follows:

```
Go_To_Points($Free\_Robots, Free\_Points$)
  $(r,p)$ := Minimum($Free\_Robots$, $Free\_Points$);
```

[5]For instance, starting from the point $(1, 0)$ on the unit circle centered in the origin of the local coordinate system, the routine can return the first axis that is hit moving counterclockwise (according to the local orientation of the X axis), after having translated the axes of symmetry 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.

```
If I am $r$ Then
  Move($p$)
Else
  do_nothing()
```

Minimum($Free\_Robots$, $Free\_Points$) finds one of the *Free_Robots* that has the minimum Euclidean distance from one of the *Free_Points* (i.e. with no robot on it). If more than one robot has minimum distance, the topmost and closest to K is chosen. We note that this robot is unique, since the robots in *Free_Robots* and the points in *Free_Points* are all on the same side of the plane, *My_Side*.

4.1 Correctness of Algorithm 1

In this section we will prove that Algorithm 1 lets the robots form the patterns described in Lemma 4.2, and that it is a pattern formation algorithm. Let us introduce one more definition.

Definition 4.4 (Agreement Configuration). An *agreement configuration* is one in which all robots agree on the position of a vertical axis K in the plane. K has no robots on it, and splits the plane in two *sides*, each containing $n/2$ robots.

We first show that Algorithm 1 lets the robots reach an agreement configuration in a finite number of cycles.

Lemma 4.3. *If the robots are not in a final configuration and not in an agreement configuration, they will reach an agreement configuration in a finite number of cycles.*

Proof. After having chosen an axis of symmetry S on line 1 of the algorithm, a robot locally rotates P in such a way that S becomes parallel to Y. As it will be shown later in Lemma 4.6, this is a necessary operation. After this, the two outermost and highest robots in the world are localized ($Outer_1$ and $Outer_2$ at line 4), and until they are at the same height[6] all the other robots do not move. Once they reach the same height (line 17), $Outer_1$ and $Outer_2$ will never move again. At this point, 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 21). Since $Outer_1$ and $Outer_2$ moved a finite number of cycles to reach their final positions, the agreement on K is reached in a finite number of cycles.

Successively, starting from the highest robots on K (line 24), all the robots on this axis will move away in a finite number of cycles (line 27), while the others stay still (line 29). Hence, after a finite number of cycles, there will be no robot on K.

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively be the two sides created by K, and computed on line 33. Until $|\mathcal{S}_1| \neq |\mathcal{S}_2|$, only the highest and closest robot to K in the most populated side is allowed to move horizontally towards the less populated side (line 36 and 37), while the others stay still (line 39 and 41). Let us suppose that $r \in \mathcal{S}_1$ is such a robot. Since r moves towards K, if it does not pass K in one cycle, it will be the highest and closest robot to K once more, and, again, the only one allowed to move. Therefore, by Assumptions A1 and A2, in a finite number of cycles it will either reach K or pass it. In the first case, since r is the only one on K, line 24 is executed, and, being n even, r moves towards the less

[6]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.

populated region, $\mathcal{S}_2$. Hence, it passes K and $\mathcal{S}_1$ decreases its size by one unit in a finite number of cycles. The same situation holds in the second case.

Therefore, in a finite number of cycles, $|\mathcal{S}_1| = |\mathcal{S}_2|$, and the lemma follows. □

Moreover, we can prove the following

Lemma 4.4. *In any agreement configuration, the number of robots on each side equals the number of final positions on that side.*

Proof. In any agreement configuration, we have that on each side there are $n/2$ robots. Moreover, by Lemma 4.2, we know that P has at least one axis of symmetry S not passing through any vertex in P. Hence, S splits P in two patterns that are symmetric with respect to S, each of them with $n/2$ points. The lemma follows from the fact that in line 2 of Algorithm 1, P is rotated in such a way that S becomes parallel to Y, and from the way the routine `Find_Final_Positions(·)` computes the final positions. □

Thus, the robots on the two sides can now move in parallel. The following lemma states that after an agreement configuration is reached, a final configuration will eventually be reached.

Lemma 4.5. *From an agreement configuration, only an agreement configuration or a final configuration can be reached, and a final configuration will eventually be reached.*

Proof. Algorithm 1 makes sure that from any initial configuration that is not final and not an agreement configuration, we reach an agreement configuration (Lemma 4.3). $Outer_1$ and $Outer_2$, after the agreement, never move again. Furthermore, since they are viewed as the outermost and topmost point in P (in `Find_Final_Positions(·)`), the other robots move only in such a way that $Outer_1$ and $Outer_2$ remain always the outermost and topmost robots in the system. Moreover, after the two sides created by the agreement on K have equal size, no robot is allowed any more to cross K, therefore the agreement on K is kept. By Lemma 4.4, there is the exact number of final positions on each side, therefore all the other robots can reach them according to the routine `Go_To_Points()`. Hence, after a number of agreement configurations, a final configuration will eventually be reached. □

Therefore, we can state the following

Theorem 4.1. *Algorithm 1 is a pattern formation algorithm.*

Corollary 4.1. *An even number of autonomous, anonymous, oblivious, mobile robots that agree on the direction and orientation of only one axis, can form only symmetric patterns with at least one axis of symmetry not passing through a vertex of P.*

Line 2 of Algorithm 1 rotates the input pattern. Therefore, at the end of the algorithm, the robots will form a pattern that is obviously scaled (since the robots must find an agreement on the unit distance), but also rotated with respect to the input pattern P. This operation is indeed necessary, as stated by the following:

Lemma 4.6. *Let $\mathcal{F}$ be the final configuration of the robots after they execute a pattern formation algorithm, and let us assume that $\mathcal{F}$ is not the initial configuration. Then $\mathcal{F}$ can not be asymmetric, and $\mathcal{F}$ must be symmetric, with at least one axis parallel to Y and not passing through any robot's positions (vertices of $\mathcal{F}$).*

Proof. From Lemma 4.2, $\mathcal{F}$ must be clearly symmetric and with at least one axis not passing through any robot's positions (vertices of $\mathcal{F}$). If none of such axes were parallel to Y, by using an argument similar to the one used in the proof of Lemma 4.2.a, we would have a contradiction. $\square$

Therefore, if the input pattern has no axis of symmetry parallel to Y, any pattern formation algorithm must allow the local rotation of P. Hence, in the final configuration the robots will form a pattern that is necessarily rotated with respect to P. That is,

Corollary 4.2. *There exists no pattern formation algorithm that does not allow local rotation of P, and that lets the robots form a symmetric pattern P that has at least one axis of symmetry that is not parallel to Y, and does not pass through any vertex.*

As a concluding remark, we note that skipping the `Rotate(`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 axis of symmetry that is parallel to Y and not passing through any vertex.

5 Conclusions

In this paper we analyzed the pattern formation problem for an even number of autonomous, anonymous, memoryless mobile robots. We analyzed which kind of patterns are achievable by such a team of units when they agree only on one axis direction and orientation. The only patterns that can be formed must be symmetric, with at least one axis of symmetry not passing through a vertex of the pattern. Moreover, we described an oblivious, deterministic algorithm that lets the robots form such patterns in finite time.

A further development of this problem could be the study of which class of patterns can be formed in the more general case, when the robots do not have any kind of agreement neither on the direction nor on the orientation of any axis. In fact, in this case it has already been proven that general patterns can not be formed by any number of robots [6]; therefore it could be interesting to know which kind of patterns, hence agreement, can be reached by a set of autonomous, mobile robots in this case.

Acknowledgment

I would like to thank Paola Flocchini and Nicola Santoro for the discussions and comments that helped with the writing of this paper.

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 Automation*, 15(5):818–828, 1999.

[2] T. Balch and R. C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Trans. on Robotics and Automation*, 14(6), December 1998.

[3] G. Beni and S. Hackwood. Coherent Swarm Motion Under Distributed Control. In *Proc. DARS'92*, pages 39–52, 1992.

[4] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative Mobile Robotics: Antecedents and Directions. In *Int. Conf. on Intel. Robots and Sys.*, pages 226–234, 1995.

[5] E. H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *ICMAS*, pages 406–413, 1995.

[6] 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, 1999.

[7] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed Coordination of a Set of Autonomous Mobile Robots. In *IVS*, 2000.

[8] 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 Autom.*, pages 833–838, 1993.

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

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

[11] L. E. Parker. On the Design of Behavior-Based Multi-Robot Teams. *Journal of Advanced Robotics*, 10(6), 1996.

[12] G. Prencipe. A New Distributed Approach to Control and Coordinate a Set of Autonomous Mobile Robots: The CORDA Model. Technical Report TR–00–10, Università di Pisa, August 2000.

[13] K. Sugihara and I. Suzuki. Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots. *Journal of Robotics Systems*, 13:127–139, 1996.

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