Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots*

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

Abstract

From an engineering point of view, the problem of coordinating a set of autonomous, mobile robots for the purpose of cooperatively performing a task has been studied extensively over the past decade. In contrast, in this paper we aim at an understanding of the fundamental algorithmic limitations on what a set of autonomous mobile robots can or cannot achieve. We therefore study a hard task for a set of weak robots. The task is for the robots in the plane to form any arbitrary pattern that is given in advance. This task is fundamental in the sense that if the robots can form any pattern, they can agree on their respective roles in a subsequent, coordinated action. The robots are weak in several aspects. They are anonymous; they cannot explicitly communicate with each other, but only observe the positions of the others; they cannot remember the past; they operate in a very strong form of asynchronicity.

We show that the tasks that such a system of robots can perform depend strongly on their common knowledge about their environment, i.e., the readings of their environment sensors. If the robots have no common knowledge about their environment, they cannot form an arbitrary pattern. If each robot has a compass needle that indicates North (the robot world is a flat surface, and compass needles are parallel), then any odd number of robots can form an arbitrary pattern, but an even number cannot (in the worst case). If each robot has two independent compass needles, say North and East, then any set of robots can form any pattern.

1 Introduction, Definitions, and Overview

1.1 Autonomous Mobile Robots

We study the problem of coordinating a set of *autonomous*, mobile robots in the plane. The coordination mechanism must be totally *decentralized*, without any central control. The robots are *anonymous*, in the sense that a robot does not have an identity that it can use in a computation, and all robots execute the exact same algorithm. Each robot has its own, *local view* of the world. This view includes a local Cartesian coordinate system with

*This work has been performed while the first and third authors have been visiting ETH Zürich and while the second and fourth authors have been visiting Carleton University. We also gratefully acknowledge the partial support of the Swiss National Science Foundation, the Natural Science and Engineering Research Council of Canada and the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche du Québec.

†School of Information, Technology and Engineering, University of Ottawa - flocchin@site.uottawa.ca

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

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

¶Theoretische Informatik, ETH Zürich - pw@inf.ethz.ch

origin, 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 sides of the axes. The robots do not have a common understanding of the *handedness (chirality)* of the coordinate system that allows them to consistently infer the orientation of the y axis once the orientation of the x axis is given; instead, knowing North does not distinguish East from West. The robots observe the environment and move; this is their only means of communication and of expressing a decision that they have taken. The only thing that a robot can do is make a *step*, where a step is a sequence of three actions. First, the robot observes the positions of all other robots with respect to its local coordinate system. Each robot is viewed as a point, and therefore the *observation* returns a set of points to the observing robot. The robot cannot distinguish between its fellow robots; they all look identical. In addition, the robot cannot detect whether there is more than one fellow robot on any of the observed points; we say, it cannot detect *multiplicity*. Second, the robot performs an arbitrary *local computation* according to its algorithm, based only on its *common knowledge* of the world (assumed e.g. to be stored in read-only-memory and to be read off from sensors of the environment) and the observed set of points. Since the robot does not memorize anything about the past, we call it *oblivious*. For simplicity, we assume that the algorithm is *deterministic*, but it will be obvious that all of our results hold for nondeterministic algorithms as well (randomization, however, makes things different). Third, as a result of the computation, the robot either stands still, or it moves (along any curve it likes). The *movement* is confined to some (potentially small) unpredictable, nonzero amount. Hence, the robot can only go towards its goal along a curve, but it cannot know how far it will come in the current step, because it can fall asleep anytime during its movement. While it is on its continuous move, a robot may be seen an arbitray number of times by other robots, even within one of its steps.

The system is totally *asynchronous*, in the sense that there is no common notion of time. Each robot makes steps at unpredictable time instants. The (global) time that passes between two successive steps of the same robot is finite; that is, any desired finite number of steps will have been made by any robot after some finite amount of time. In addition, we do not make any timing assumptions within a step: The time that passes after the robot has observed the positions of all others and before it starts moving is arbitrary, but finite. That is, the actual move of a robot may be based on a situation that lies arbitrarily far in the past, and therefore it may be totally different from the current situation. We feel that this assumption of *asynchronicity within a step* is important in a totally asynchronous environment, since we want to give each robot enough time to perform its local computation.

1.2 Pattern Formation

In this paper, we concentrate on the particular coordination problem that requires the robots to form a specific geometric pattern, the *pattern formation* problem. 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 (just like a flock of birds or a troupe of soldiers); it is interesting algorithmically, because if the robots can form any pattern, they can agree on their respective roles in a subsequent, coordinated action.

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. The robots are said to *form the pattern*, if the actual positions of the robots coincide with the points of the pattern, where the pattern may be *translated*, *rotated*, *scaled*, and *flipped* into its mirror position 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. Note that in our algorithms, we do not need to and we will not make use of the possibility of rotating the pattern.

The pattern formation problem for *arbitrary* patterns is quite a general member in the class of problems that are of interest for autonomous, mobile robots. 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. This reflects the general direction of the investigation in this paper: What coordination problems can be solved, and under what conditions? 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 (reminiscent of bees in a bee dance). For oblivious robots, even this sending of information is impossible, since the others will not remember previous positions. Hence, our study is at the extreme end in two ways: The problem is extremely hard, and the robots are extremely weak.

In an attempt to understand the power of common knowledge for the coordination of robots, we study the pattern formation problem under several assumptions. We give a complete characterization of what can and what cannot be achieved. First, we show that for an arbitrary number of robots that know the direction and the orientation of both axes, the pattern formation problem can be solved. Here, knowing the direction of the x axis means that all robots know and use the fact that all the lines identifying their individual x axes are parallel. Similarly, knowing the orientation of an axis means that the positive side of that axis in the coordinate system coincides for all robots. As an example for an axis whose direction and orientation is known, each robot may have a built-in compass needle that points North, with all compass needles parallel, consistently for all robots. Note that knowledge of the directions and orientations of both axes does not imply knowledge of the origin or the unit of length. In order to solve the pattern formation problem, the robots, however, have to agree on both, origin and unit length. We propose a simple algorithm that achieves this agreement. Second, we study the case of the robots knowing one axis direction and orientation. We show that the pattern formation problem can be solved whenever the number of robots is odd, and that it is in general unsolvable when the number of robots is even. Third, we show that the situation is the same, if one axis direction is known, but not the orientation of the axis. Fourth, we show that if no axis direction (and therefore also no orientation) is known, the problem cannot be solved in general.

1.3 Related Work

The problem of controlling a set of autonomous, mobile robots in a distributed fashion has been studied extensively, but almost exclusively from an engineering and from an artificial intelligence point of view. In a number of remarkable studies (on social interaction leading to group behavior [6], on selfish behavior of cooperative agents in animal societies [7], on primitive animal behavior in pattern formation [2], to pick just a few), algorithmic aspects were somehow implicitly an issue, but clearly not a major concern, let alone the focus, of

the study.

We aim at identifying the algorithmic limitations of what autonomous, mobile robots can or cannot do. An investigation with this flavor has been undertaken within the AI community by Durfee [4], who argues in favor of limiting the knowledge that an intelligent agent must possess in order to be able to coordinate its behavior with others. The work of Suzuki and Yamashita [12, 10, 11] is closest to our study (and, with this focus, a rarity in the mobile robots literature); it gives a nice and systematic account on the algorithmics of pattern formation for robots, under several assumptions on the power of the indivdual robot. The models that we use differ from those of [12, 10, 11] in the fact that our robots are as weak as possible in every single aspect of their behavior. The reason is that we want to identify the role of the robots' common knowledge of the world for performing a task. In contrast with [12, 10, 11], we do not assume that on a move, we know ahead of time the limited, but nonzero distance that a robot travels. We do not assume that the distance that a robot may travel in one step is so short that no other robot can see it while it is moving. We do not assume that the robots have a common handedness, called *sense of orientation* in [12, 10].

The most radical deviation from previous models may, however, be our assumption of *asynchronicity within one step*. In contrast, [12, 10, 11] assume the *atomicity of a step*: A robot moves immediately after it has observed the current situation, with all awake robots moving at the same clock tick (some robots may be asleep). This difference influences the power of the system of robots so drastically that in general, algorithms that make use of atomicity within one step do not work in our model; in particular, this is true for the work in [12, 10, 11].

2 Knowledge of Both Axis Directions and Orientations

For the case in which the directions and orientations of both axes are common knowledge, the robots can form an arbitrary given pattern as follows. First, each robot for itself identifies the first and second point of the pattern, in lexicographic order with respect to the coordinates in which the pattern is given. Second, each robot identifies the first and second robot position in lexicographic order in the world. This can be done since all robots have the same notion of lexicographic order, as a result of their knowledge of both axis directions and orientations. Third, the first and second robots put themselves in positions matching the first and second pattern points, where the pattern may be translated and scaled, but not rotated or flipped. This can be done by moving the first robot only, in such a way that it always remains first. Now, the first two robot positions identify the translation and scaling of the pattern, and they make this visible for all the robots. Fourth, all the other robots go to points of the pattern. This can be done by moving one robot after the other step by step to a pattern point. The sequence of robots is chosen to guarantee that after one robot has made even only a small move towards its destination, no other robot will move before that one has reached its destination.

More precisely, each robot executes the following algorithm in each step.

Algorithm 1 (Both axis directions and orientations).
Input: An arbitrary pattern P described as a sequence of points $p_1, \ldots, p_n$, given in lexicographic order. The directions and orientations of the x axis and the y axis is common knowledge.

Begin
$\alpha :=$ `Angle(`p_1`,` p_2`)`;
Give a lexicographic order to all the robots in the system,
including myself, say from left to right and from bottom to top;
A :=First robot in the order;
B :=Second robot in the order;
$\beta :=$ `Angle(`A`,` B`)`;
(1) **If** I am B
Then `Do_nothing()`
Else If I am A
Then If $\alpha = \beta$
(2) **Then** `Do_nothing()`
(3) **Else** `Go_Into_Position(`A`,` B`,` α`)`
Else %I am neither A nor B%
If $\alpha = \beta$
(4) **Then** $Unit := \overline{AB}$; % all the robots agree on a common unit distance %
(5) $Final\_Positions :=$ `Find_Final_Positions(`A`,` B`,` $Unit$`)`;
If I am on one of the $Final\_Positions$
(6) **Then** `Do_nothing()`
Else $Free\_Robots := \{robots\ not\ on\ one\ of\ the\ Final\_Positions\}$;
$Free\_Points := \{Final\_Positions\ with\ no\ robots\ on\ them\}$;
`Go_To_Points(`$Free\_Robots$`,` $Free\_Points$`)`;
(7) **Else** `Do_nothing()`
End

`Angle(`p`,` q`)` computes the angle between the positive horizontal axis passing through p and the segment $\overline{pq}$.

`Do_nothing()` terminates the local computation and the current step of the calling robot.

`Go_Into_Position(`A`,` B`,` α`)` orders A to move so as to achieve angle α with B while staying lexicographically first.

`Find_Final_Positions(`A`,` B`,` $Unit$`)` figures out the final positions of the robots according to the given pattern,
and the positions of A and B. The common scaling of the input pattern is defined by the common unit distance $Unit$.

`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(`$Free\_Robots$`,` $Free\_Points$`)`
Begin
$(r, p) :=$ `Minimum(`$Free\_Robots$`,` $Free\_Points$`)`;
(8) **If** I am r
Then `Move(`p`)`
Else `Do_nothing()`
End

`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 one smaller in the lexicographic order is chosen.

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

Theorem 2.1. *With Algorithm 1, the robots correctly form the input pattern P.*

Proof. We distinguish two cases.

1. At the beginning we have $\alpha \neq \beta$. In this case the only robot that can move at the beginning is A (steps (1), (3), and (7)). This situation holds until $\alpha = \beta$, that is when A and B are placed in the plane on a line parallel to the line that the first two robots in the input pattern P form, where the coordinate system of the input pattern is identified with the local coordinate system for each robot. From now on, A never moves again (step (2)). At this point, all the other robots agree on a common unit distance. Subsequently, they figure out uniquely the final positions of the pattern in the plane (see `Find_Final_Positions(`A`, `B`, `$Unit$`)` and steps (4), (5)).

   If a robot is on a final position, it never moves again (step (6)). Among those robots not in a final position, routine `Minimum(`$Free\_Robots$`, `$Free\_Points$`)` allows only one robot at a time to move (step (8)) towards its final position. Since the number of points in the pattern coincides with the number of robots in the system, eventually each robot will occupy a pattern point position.

2. At the beginning we have $\alpha = \beta$. This is a subcase of the previous case, hence the proof follows similarly.

□

Corollary 2.1. *With common knowledge of two axis directions and orientations, a set of autonomous, anonymous, oblivious, mobile robots can form an arbitrary given pattern.*

3 Knowledge of One Axis Direction and Orientation

Let us now look at the case when only one axis direction and orientation are known. As an aside, note that this case would trivially coincide with the first one, if the robots would have a common handedness (or sense of orientation, as Suzuki and Yamashita call it [12, 10]). We first show that in general, it is impossible to break the symmetry of a situation. We will then show that for the special case of an odd number of robots, symmetry can be broken and an arbitrary pattern can be formed.

Theorem 3.1. *In a system with n anonymous robots that agree only on one axis direction and orientation, the pattern formation problem is unsolvable when n is even.*

Proof. Let us call the known axis the y axis. For the sake of contradiction, let $\mathcal{A}$ be a deterministic algorithm for solving the pattern formation problem, and let P be a specific pattern of n points, with a unique topmost point p (topmost refers to the maximum y coordinate value). The task for the robots is to form P by executing algorithm $\mathcal{A}$. 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 $\widehat{r}$. 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,

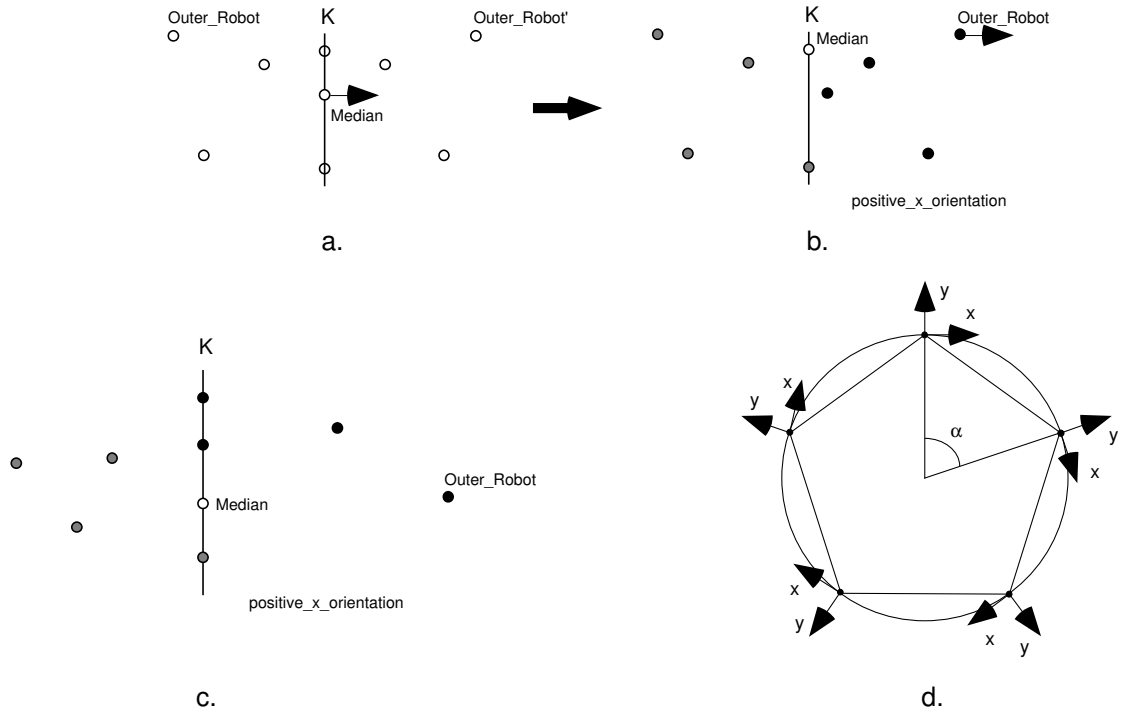

Figure 1: Breaking Symmetry from (a) to (b); defining the sides (c); the unbreakable symmetry of a 5-gon (d).

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. This never leads to a situation with a unique topmost robot, and hence the pattern cannot be formed. □

In contrast, we now show that for breaking the symmetry, it is enough to know that the number n of robots is odd. We make use of the fact that either the robots are in a symmetric initial situation, in which there is a unique middle robot that will move in order to break the symmetry, or the situation is not symmetric in the very beginning, and hence the asymmetry can be used to identify an orientation of the x axis. We feel that this technique of symmetry breaking for mobile robots may have other applications, and hence it may be of independent interest. Note that the direction only of the x axis can be determined locally by each robot, since it is just orthogonal to the direction of the y axis or, in other words, cuts the 180° angle in half.

Algorithm 2 (One axis direction and orientation).
Input: An arbitrary pattern P described as a sequence of points $p_1, \ldots, p_n$, given in lexicographic order. The direction and orientation of the y axis is common knowledge.

Begin
If We are in a final configuration **Then** `Do_nothing()`;
p_m := `Median_Pattern_Point(`P`)`; % median pattern point in x direction %
p := `Outermost_Pattern_Point(`P`)`; % outermost pattern point with respect to the median %

Pattern_Unit_Length := Horizontal distance in the pattern coordinate system between the vertical lines through p_m and p;
K := `Median_Robot_Line()`; % through the median robot position %
(*Outer_Robot*, *Outer_Line*, *positive_x_orientation*):= `Outermost_Robot_Position(`K`)`;
Median_Robot := `Find_Median_Robot(`K`)`;
Final_Positions := `Find_Final_Positions(`*Median_Robot*, *positive_x_orientation*, *Pattern_Unit_Length*, `Distance(`K, *Outer_Line*`)/2)`;
Free_Points := {*Final_Positions on my side with no robots on them*};
If I am the *Median_Robot*
Then `Do_nothing()`
Else If I am the *Outer_Robot*
Then If *Free_Points* contains just one (last) free point
Then `Move_Towards(`*last free point*`)`
Else `Do_nothing()`
Else If I am on one of the *Final_Positions*
Then `Do_nothing()`
Else *Free_Robots* := robots on my side not on *Final_Positions*;
`Go_To_Points(`*Free_Robots*, *Free_Points*`)`

End

`Median_Pattern_Point(`P`)` finds the median point in direction of x in the input pattern P according to the local orientation of the axis. The ordering is given left-right, bottom-up.

`Outermost_Pattern_Point(`P`)` finds the point in the input pattern P that lies on the vertical line farthest from the vertical line through the median. If more than one point exists, then the highest and rightmost is chosen according to the local orientation of the axis.

`Median_Robot_Line()` returns the vertical line through the median robot position. Note that the position of this line does not depend on the local orientations of the x axes of the robots.

`Outermost_Robot_Position(`K`)` uniquely determines a robot that is outermost with respect to K. It does so by breaking symmetry of the situation, if necessary, in the following way (see Figure 1(a) and (b)):

`Outermost_Robot_Position(`K`)`
Begin
(*Outer_Robot*, *Outer_Robot*$'$, *unique*) := `Outer_Two_Robots(`K`)`;
If *not unique*
Then If `Symmetric(`K`)`
Then If I am the median of the points on K
Then `Move(`*to my right by* ϵ`)`
Else `Do_nothing()`
Else *positive_x_orientation* := `Outermost_Asymmetry(`K`)`;
Outer_Robot := Choose between *Outer_Robot* and *Outer_Robot*$'$ the one that lies on the *positive_x_orientation*;
If I am *Outer_Robot*
Then `Move(`*away from* K *by* ϵ`)`
Else `Do_nothing()`
Else *positive_x_orientation* := side on which *Outer_Robot* lies;

Outer_Line := vertical line on which *Outer_Robot* lies;
Return (*Outer_Robot*, *Outer_Line*, *positive_x_orientation*)
End

`Outer_Two_Robots(`K`)` finds either two or a unique single topmost robot(s) that lie(s) on the farthest vertical line(s) from K. The variable *unique* tells whether the robot found has been unique.

`Symmetric(`K`)` returns *True* if the current configuration of the robots in the system is symmetric with respect to K, in the following sense. The configuration is called *symmetric with respect to* K, if there is a perfect matching for all the robots not on K, such that any two matched robots lie on two vertical lines that are in symmetric position with respect to K (see Figure 1(a)).

`Outermost_Asymmetry(`K`)` identifies, for an asymmetric configuration, the unique halfplane with respect to K in which a robot r lies that (in some matching) has no symmetric partner with respect to K, and that is on the farthest vertical line from K among all robots with this property (see Figure 1(b)).

`Find_Median_Robot(`K`)` finds the median robot position in the current configuration of the robots. This median robot position splits the robot positions into two equal size subsets, of size $(n-1)/2$, defined as follows. One subset is in the halfplane of *positive_x_orientation*, including the points on K that are *above* the median robot position, and the other subset is in the other halfplane of K, including the points on K that are *below* the median robot position; from now on, we will call these the two *sides* (see Figure 1(c)). According to this definition, the median robot position is unique, and each robot (even if it lies on K) can decide to which side it belongs.

`Find_Final_Positions(`*Median_Robot*, *Side*, *Pattern_Unit_Length*, *World_Unit_Length*`)` returns the set of final positions of the robots according to the given pattern, based on the agreement on *Median_Robot* and on *positive_x_orientation*. The common scaling of the input pattern is defined by identifying *Pattern_Unit_Length* with *World_Unit_Length*.

`Distance(`K, L`)` returns the horizontal distance between the two vertical lines K and L.

3.1 Correctness

To see that the above algorithm solves the pattern formation problem for an arbitrary pattern, we argue as follows. First, we show that the robots initially arrive at an agreement configuration, by breaking symmetry if necessary. Then, they translate and scale the pattern with respect to the median and the outermost point, and finally they move to their destinations. To present this argument in more detail, we start with a brief definition. A *configuration (of the robots)* is a set of robot positions, one position per robot, with no position occupied by more than one robot. An *agreement configuration* is a configuration of the robots in which all robots agree on a unique *median* robot and a unique *outermost* robot, as defined in the above routines `Find_Median_Robot(`K`)` and `Outermost_Robot_Position(`K`)`. A *final configuration* is a configuration of the robots in which the robots form the desired pattern. Note that a final configuration might or might not be an agreement configuration.

Lemma 3.1. *If the robots are not in a final configuration and not in an agreement configuration, they will reach an agreement configuration within one move.*

Proof. In Algorithm 2, the robots will never disagree on the vertical axis K of the median robot position. If the robots are neither in a final configuration nor in an agreement configuration, the robots disagree on at least one, the outermost robot position or the median robot position. If they do not disagree on the outermost robot position ($unique = True$, see Figure 1(c)), then they will not disagree on the median (by `Find_Median_Robot(`K`)`). If they, hence, disagree on the outermost robot position (see Figure 1(a) and (b)), and the robot positions are symmetric with respect to K, the median robot position is uniquely identifiable. The routine `Outermost_Robot_Position()` will move the median robot, thereby break the symmetry, and uniquely identify an outermost robot. If the robot positions were not symmetric in the first place, then the outermost robot is unique anyway. Note that if there is still a robot in the symmetric position of the outermost robot, the latter moves a bit outwards to make the subsequent computations easier. □

Lemma 3.2. *In any agreement configuration, the number of robots on each side equals the number of final positions on that side.*

Proof. From the choice of the median robot and the way we split the set of robots in `Find_Median_Robot(`K`)`, we conclude that the lemma holds initially. The number of robots on a side does not change in the routine `Go_To_Points()`, and the lemma follows. □

Lemma 3.3. *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 2 makes sure that from any initial configuration that is not final and not an agreement configuration, we reach an agreement configuration (see Lemma 3.1). The *Median_Robot* after the agreement never moves, and the other robots move only in such a way that it remains always the median robot, due to Lemma 3.2. The robot *Outer_Robot* after the agreement does not move, except in the very last step, when all other robots have reached their final positions according to the routine `Go_To_Points()`. Hence, after a number of agreement configurations, a final configuration will eventually be reached. □

From these lemmas, we conclude:

Theorem 3.2. *With Algorithm 2, the robots correctly form the input pattern P.*

Corollary 3.1. *With common knowledge of one axis direction and orientation, an odd number of autonomous, anonymous, oblivious, mobile robots can form an arbitrary given pattern. An even number of robots cannot form an arbitrary given pattern.*

4 Knowledge of One Axis Direction

Note that the difference to the previous section is only the lack of knowledge about the axis orientation. For solving the pattern formation problem in this case, we can use an algorithm similar to the one used in Section 3. We easily observe that with slight modifications in Algorithm 2, the agreement on the orientation of the x axis could have been achieved without using the knowledge of the orientation of the y axis. More precisely, we can do the following:

1. Find the vertical line K on which the median robot lies (as before, this is independent from the direction of x).

2. Find the *Outermost* robots with respect to K. Since we do not know the orientation of the given axis, let's say the y axis, it is possible that we find more than one outermost robot; there are at most four of them, on both sides of K to the top and to the bottom. In this case, we will detect an (outermost) asymmetry with respect to K as before, or create it as follows. If the configuration is symmetric with respect to K, the median robot is uniquely identified, and it moves by some small amount $\epsilon > 0$ to its right, breaking the symmetry. So now, as in the previous section, all the robots agree on the positive direction of the x axis (the side where the outermost asymmetry lies), and at most 2 outermost robots remain (the bottom and top ones on the positive x side, say).

3. The same technique and argument now applies to the x axis as the given one. In this way, an agreement also on the orientation of the y axis can be reached. Now, we can select a unique *Outermost* robot out of the at most two that were remaining, and let it (for convenience) move by ϵ outwards.

4. The robots can compute their unique final positions and go towards them, in the same way as in Algorithm 2.

Using Theorem 3.1, we therefore conclude:

Theorem 4.1. *With common knowledge of one axis direction, an odd number of autonomous, anonymous, oblivious, mobile robots can form an arbitrary given pattern, while an even number cannot.*

5 No Knowledge

We will now show that giving up the common knowledge on at least one axis direction leads to the inability of the system to form an arbitrary pattern.

Theorem 5.1. *With no common knowledge, a set of autonomous, anonymous, oblivious, mobile robots cannot form an arbitrary given pattern.*

Proof. For the sake of contradiction, let $\mathcal{A}$ be a deterministic algorithm for solving the pattern formation problem with no common knowledge. We show that there is a desired pattern, an initial configuration of robots, and a sequence of steps that the robots can take according to $\mathcal{A}$, such that the robots never can form the desired pattern. Let the desired pattern be any pattern that is different from a regular n-gon and a single point, where n is the number of robots, and let the initial positions be such that the robots form a regular n-gon. Let $\alpha = 360°/n$ be the characteristic angle of the n-gon, and let the local coordinate system of each robot be rotated by α with respect to its neighbor on the polygon (see Figure 1(d)). In this situation, all the robots have the same view of the world. Now, for any move that any one robot can make in its local coordinate system by executing algorithm $\mathcal{A}$, we know that each robot can make the same move in its local coordinate system. If all of them move in the exact same way at the same time, they again end up in a regular n-gon or a single point. Therefore, by letting all the robots move at the same time in the same way, we always proceed from one regular n-gon or single point to the next. Hence, the desired pattern cannot be formed. □

6 Discussion

We have shown that from an algorithmic point of view, only the most fundamental aspects of mobile robot coordination are being understood. In a forthcoming paper, we propose two algorithms for the point formation problem for oblivious robots; the first one does not need any common knowledge, and the second one works with limited visibility, when two axes are known [5]. There is a wealth of further questions that suggest themselves. First, we have shown that an arbitrary pattern cannot always be formed; it is interesting to understand in more detail which patterns or classes of patterns can be formed under which conditions, because this indicates which types of agreement can be reached, and therefore which types of tasks can be performed. Second, in contrast with other researchers who have looked at modelling natural behaviors, our robots perform quite a complex computation in each step; it is interesting to understand in more detail the tradeoff between computation complexity and knowledge of the world. Third, the operating conditions of our robots have been quite restricted; it is interesting to look at more relaxed models, where for instance robots have a bounded amount of memory at their disposition, or they have a spatial extent, they collide as they move, or their camera rotates slowly when taking a picture, so that a robot may never see the world as it was at any time instant. Slightly faulty snapshots, a limited range of visibility [1], obstacles that limit the visibility and that moving robots must avoid or push aside, as well as robots that appear and disappear from the scene clearly suggest that the algorithmic nature of distributed coordination of autonomous, mobile robots merits further investigation.

References

[1] H. Ando, I. Suzuki, and M. Yamashita. Formation and Agreement Problems for Synchronous Mobile Robots with Limited Visibility. *Proc. IEEE Int. Symp. on Intelligent Control*, pages 453–460, August 1995.

[2] T. Balch and R. C. Arkin. Motor Schema-based Formation Control for Multiagent Robot Teams. *ICMAS*, pages 10–16, 1995.

[3] J. Desai, J. Ostrowski, and V. Kumar. Control of Formations for Multiple Robots. *IEEE International Conference on Robotics and Automation*, May 1998.

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

[5] P. Flocchini, G. Prencipe, N. Santoro, E. Welzl, and P. Widmayer. Point Formation for Oblivious Robots. *manuscript, in preparation*, 1999.

[6] M. J. Mataric. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. *From Animals to Animats 2: Int. Conf. on Simulation of Adaptive Behavior*, pages 423–441, 1993.

[7] L. E. Parker. Adaptive Action Selection for Cooperative Agent Teams. *Proc. Second Int'l. Conf. on Simulation of Adaptive Behavior*, pages 442–450, 1992.

[8] K. Sugihara and I. Suzuki. Distributed Motion Coordination of Multiple Mobile Robots. *Proc. 5th IEEE Int'l. Symp. Intelligent Control*, pages 138–143, 1990.

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

[10] I. Suzuki and M. Yamashita. Formation and Agreement Problems for Anonymous Mobile Robots. *Proceedings of the 31st Annual Conference on Communication, Control and Computing, University of Illinois, Urbana, IL*, pages 93–102, 1993.

[11] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots - Formation and Agreement Problems. *Proc. Third Colloq. on Struc. Information and Communication Complexity (SIROCCO)*, pages 313–330, 1996.

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