Discrete Bee Dance Algorithms for Pattern Formation on a Grid

Noam Gordon, Israel A. Wagner and Alfred M. Bruckstein
Center for Intelligent Systems, CS Department
Technion — Israel Institute of Technology
32000 Haifa, Israel
{ngordon, wagner, freddy}@cs.technion.ac.il

Abstract
This paper presents a solution to the problem of pattern formation on a grid, for a group of identical autonomous robotic agents, that have very limited communication capabilities. The chief method of communication between the agents is by moving and observing their positions on the grid. The proposed algorithm is a sequence of several coordinated “bee dances” on the grid, through which the agents broadcast information and cooperate in order to reach agreements and resolve problems due to their indistinguishability.

1 Introduction
In recent years, the interest in distributed mobile robotic systems is growing rapidly. Recognizing the benefits of distributed computing with regard to robustness, performance and cost as well as the amazingly complex feats successfully performed by colonies of social insects, an increasing number of researchers are trying to design and analyze distributed systems of multiple mobile autonomous robots, hoping to gain similar advantages. For a good discussion and overview of the area see [3].

A fundamental problem in the field of distributed robotics is the Formation Problem, i.e., the problem of coordinating a group of mobile agents (robots) to form a desired spatial pattern. This is important, for example, when deploying a group of mobile robots or a self-arranging array of sensors. By forming a pattern, tasks can be allocated, groups created, leaders elected etc. Therefore, this problem was considered by many, including [2, 6, 9, 10]. Many of these works mainly address engineering aspects and design heuristics or behaviors that seem to work well in simulation or in real robot teams.

Suzuki, Yamashita and others ([1, 8]) took a more theoretical approach, and analyzed fundamental algorithmic questions regarding formation: Given a group of autonomous, anonymous and homogenous mobile agents, with no explicit
communication except the mutual observation of their movements, what kinds of spatial patterns can they be programmed to create? One of the main results is that under several assumptions, which seem to be very natural, a pattern is achievable if and only if it is purely symmetrical, i.e., a perfect regular polygon (or a point), or several concentric ones. The basic impossibility argument is that the agents may initially happen to be distributed in a symmetrical pattern, and the agents may happen to be perfectly synchronous forever. Then, since they all perform the same algorithm, they will always see the same view and make the same movements. Thus, the symmetry will never break. Suzuki and his colleagues point out the close connection between formation and agreement problems — On one hand, by agreeing on a coordinate system, the agents can easily form any pattern. On the other hand, the agents can use formations to agree on many things. For example, if they form a straight line, they can agree on an $x$ axis or a leader. They indeed use such ideas of communicating information through movements in their proposed algorithms. This form of communication by observing others’ movement is common in nature. A fine example is that of scouting bees that communicate their findings — the location of nectar — through dance-like motions.

Several other researchers were inspired by Suzuki et al’s work. Notably, Flocchini, Prencipe and others discussed similar questions, using a model which differs mainly w.r.t. the asynchrony of the agents’ actions, and concentrates on oblivious (memoryless) agents ([5, 7]). They showed analogous impossibility results for asymmetric patterns and discussed the differences between their model and that of Suzuki et al. Defago ([4]) used a similar model and discussed the formation of a circle by oblivious agents. In most works, special consideration was given to the point formation or gathering problem, since a point may be thought of as the most fundamental of all patterns.

In this work, we deal with the same questions and model of Suzuki et al. However, we assume that the world is a (discrete) grid, instead of the continuous plane. The impossibility results mentioned above are indeed very strong and elegant. They are based on the assumption that there can be symmetry in the system, which may remain unbroken for an indefinite time, in fact, as long as the agents are synchronous. However, we believe that, in real robots, it is reasonable to assume that the probability for asynchronous autonomous robots to remain synchronous for a long time is negligibly small. Thus, by adding this assumption to our model, we are able to give a rather strong possibility result — a “generic” algorithm which solves the formation problem for any arbitrary pattern, within a finite expected time. In fact, we can show that all situations of stagnation due to symmetry in our algorithm are meta-stable, that is, the symmetry will break sooner or later.

We begin by defining our model and problem. We show how the point formation problem can be solved in our model. Then we present an algorithm which through a chain of “bee dances”, leads the agents toward agreement on a common coordinate system and a total ordering, and form any preprogrammed desired pattern.
2 Preliminaries

We first present a formal definition of the world model and the problem we are discussing.

2.1 Model definition

The world consists of an infinite rectangular grid \((\mathbb{Z}^2)\) and \(n\) point agents living in it. We assume that, initially, the agents occupy distinct positions and they do not have a common coordinate system. Each agent sees the world (the locations of all agents) with respect to its own private coordinate system. Time is discrete (\(t = 0, 1, \ldots\)). At each time step, each agent may be either awake or asleep, and the agent has no control over the scheduling of its waking times. A sleeping agent does nothing and sees nothing, i.e., it is unaware of the world’s state. When an agent wakes up, it sees the locations of all agents, and moves to an adjacent point on the grid (i.e., a 4-neighbor) or stays in place, according to its algorithm. The algorithm’s input is the agent’s current view of the world and possibly some private internal memory which may be updated by the algorithm. There are no occlusions and no collisions. Several agents may occupy the same point. All agents are anonymous: they cannot be distinguished by their appearance, and homogenous: they don’t have any individuality (such as a name or id) and they all perform the same algorithm.

Regarding the waking times of the agents, we make two important assumptions. First, no agent will sleep forever, i.e., it will always wake up again. Second, we say that the agents are strongly asynchronous: For any subset \(G\) of the agents and in each time step, there is a non-zero probability that \(G\) will be the set of waking agents. This implies that the expected time for any group of agents to remain synchronous (i.e., always wake up together) is finite.

The agents’ only means of tele-communication is through movement and mutual observation of their positions. In addition, they do have a minimal form of local (zero-range) communication: each agent has a binary flag, whose state can be observed only by agents in the same point. We found it necessary to assume this additional ability of the agents, in order to enable them to gather in a single point and then to be able to move out of it. Without such an ability, an agent would have no way of knowing if the others have already witnessed the gathering (since the sleeping time of an agent is unbounded).

This model is exactly identical to that of Suzuki et al, except for the change from a continuous to a discrete world, the addition of the binary flags to the agents and the strong asynchrony assumption.

2.2 Problem definition

Given a pattern or collection of coordinates \(F = (q_1, \ldots, q_n)\) on the grid, the Formation Problem is the problem of finding a distributed algorithm under the assumptions presented above, such that from any initial distribution of the agents, they will eventually arrange themselves in the desired pattern \(F\). Note
that in the other cited works, “eventually” means “in strictly finite time”. In our context, it is “in finite expected time”, which is a weaker condition. This interpretation was defined by Suzuki and Yamashita as the weaker Convergence Problem.

Since there is no absolute coordinate system, the desired pattern may be formed at any location and in any orientation in the world. Furthermore, since the agents do not initially agree on the handedness (or “chirality”) of their coordinate systems, a “mirror image” of the desired pattern may also be formed.

3 Point formation

We provide an algorithm which makes all agents gather in a single point on the grid. We use the most intuitive idea: each agent moves toward the location of the center of mass (or COM) of all agents (The center of mass of n points $p_1, \ldots, p_n$ is defined as $\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i$). Of course, “move toward” must be accurately defined when one lives on a discrete grid, as the agent must choose between only four possible directions (or not to move at all).

We begin with the simplest case, where the world is a one-dimensional grid ($\mathbb{Z}$). Denote the agent’s current position (in its own coordinate system) by $x_t$, the position of COM by $\bar{x}$ and the position of COM, relative to the agent, by $\Delta x = \bar{x} - x_t$. Algorithm 3.1, executed by each agent every time it wakes up, solves the one-dimensional case. The agent moves toward COM, unless it’s already within less than 1/2 unit from the agent (In the discrete world, this is equivalent to “being in the same cell” as the agent, where a cell is a unit square centered at a point on the grid).

**Algorithm 3.1** Point formation in one dimension

1: if $|\Delta x| < 1/2$ then
2:   Do not move. //Already close to COM.
3: else
4:   Move one step toward COM.

**Lemma 3.1.** Algorithm 3.1 solves the point formation problem in one dimension.

**Proof.** Let $(x_1, \ldots, x_n)$ be the current positions of the n agents. Denote $x_{\min} = \min_i x_i$, $x_{\max} = \max_i x_i$ and $\delta = x_{\max} - x_{\min}$. We will show that $\delta$ will decrease in finite expected time. There are four possible cases (See Figure 1):

1. $x_{\max} - \bar{x} < 1/2$: In this case, all of the agents in $x_{\max}$ will stay there, while all other agents (which are all to their left), including those at $x_{\min}$, will move right. Thus, eventually $x_{\min}$ will be emptied (and $x_{\min}$ will shift right), making $\delta$ decrease.

2. $\bar{x} - x_{\min} < 1/2$: This is symmetric to case 1.
3. \( x_{\text{min}} + 1/2 \leq \bar{x} \leq x_{\text{max}} - 1/2 \) and \( \delta > 1 \): As long as \( \bar{x} \) is within this range, agents in the extremes (\( x_{\text{min}} \) and \( x_{\text{max}} \)) will move inwards when they wake up. Thus, eventually either \( x_{\text{min}} \) and/or \( x_{\text{max}} \) cells become empty, effectively making \( \delta \) decrease. Alternatively, \( \bar{x} \) will “drift” out of this range, effectively changing to case 1 or 2 above. Yet another possibility is that the system will switch to case 4 below.

4. \( \bar{x} = x_{\text{min}} + 1/2 = x_{\text{max}} - 1/2 \) (implying \( \delta = 1 \)): This is a unique metastable state, where all of the agents reside in two neighboring cells, each one containing exactly \( n/2 \) agents. In this state, each waking agent will move to the other cell. In order for this symmetry to sustain, exactly the same amount of agents must wake up in both cells. Otherwise, \( \bar{x} \) will shift towards either \( x_{\text{max}} \) or \( x_{\text{min}} \), changing to case 1 or 2 above, respectively. Since we assume that the agents are strongly asynchronous, the expected time to remain in this state is finite.

We see that case 4 will eventually switch to case 1 or case 2, and in all other cases, \( \delta \) will eventually decrease. Therefore, beginning with any finite \( \delta \), it will eventually decrease to 0 (i.e., the agents will all gather in a single point).

In spite of its simplicity, Algorithm 3.1 is quite powerful, in the sense that it is oblivious and thus self-stabilizing (recovers from any finite number of errors, including even “Byzantine”, i.e. arbitrary, movements). Furthermore, as the following lemma states, the algorithm survives crash failures: If some agents “die” (i.e., will never move again), all the live agents will still eventually gather in a single point.

**Lemma 3.2.** In case of crash failures, Algorithm 3.1 still solves the point formation problem in one dimension for the live agents.

**Proof.** Lemma 3.2 is proven in a similar fashion to Lemma 3.1. Denote the center of mass of the “dead” agents by \( \bar{x}_D \). Observe that since the algorithm
depends only on \( \text{COM} \), and the calculation of \( \text{COM} \) is linear, we may regard all dead agents as one static “weight”, positioned in \( \bar{x}_D \). Let \( l \) be the number of living agents. We examine the equivalent situation where we have the \( l \) living agents plus one dead (possibly “heavy”) agent at \( \bar{x}_D \). We use the same notation as in the proof of Lemma 3.1 above, only that \( x_{\text{min}}, x_{\text{max}} \) and \( \delta \) relate to the \( l+1 \) agents. Additionally, we use the analogue notation \( x^L_{\text{min}}, x^L_{\text{max}} \) and \( \delta^L \), which refers only to the \( l \) living agents.

We now show that \( \delta^L \to 0 \). There are several possible cases:

1. \( x_{\text{max}} - \bar{x} < 1/2 \): In this case, all of the agents in \( x_{\text{max}} \) will stay there, while all other living agents (which are all to their left), including those at \( x_{\text{min}} \), will move right. If \( x_{\text{min}} \neq \bar{x}_D \), then \( x_{\text{min}} \) will eventually be emptied, making \( \delta \) decrease. Otherwise \( (x_{\text{min}} = \bar{x}_D) \), \( \delta \) will not decrease. However, \( \delta^L \) will decrease and the system will remain in this state \( (x_{\text{max}} - \bar{x} < 1/2) \). Thus, \( \delta^L \) will ultimately diminish to 0.

Note that if \( x_{\text{max}} = \bar{x}_D \) and \( x^L_{\text{max}} > x_{\text{max}} - 1 \), it is possible that \( \delta \) will increase, as living agents in \( x^L_{\text{max}} \) may “hop” over \( \bar{x}_D \). However, this can occur only once.

2. \( \bar{x} - x_{\text{min}} < 1/2 \): This is symmetric to case 1.

3. \( x_{\text{min}} + 1/2 \leq \bar{x} \leq x_{\text{max}} - 1/2, \ \delta > 1 \) and case 4 below doesn’t hold: As long as \( \bar{x} \) is within this range, living agents in the extremes \( (x_{\text{min}} \text{ and } x_{\text{max}}) \) will move inwards when they wake up. Thus, eventually either \( x_{\text{min}} \) and/or \( x_{\text{max}} \) cells become empty, making \( \delta \) decrease. Alternatively, \( \bar{x} \) will “drift” out of this range, effectively changing to case 1 or 2 above. Yet another possibility is that the system will be in case 4 below.

4. \( \bar{x} = x^L_{\text{min}} + 1/2 = x^L_{\text{max}} - 1/2 \) (implying \( \delta^L = 1 \)): This is a unique metastable state, where all of the agents reside in two neighboring cells, and \( \text{COM} \) is exactly in the middle between those cells. In this state, each waking agent will move to the other cell. In order for this symmetry to sustain, exactly the same amount of agents must wake up in both cells. Otherwise, \( \bar{x} \) will shift towards either \( x^L_{\text{max}} \) or \( x^L_{\text{min}} \), changing to case 1, 2 or 3 above (In the latter case, \( \delta \) is guaranteed to eventually decrease). Since we assume that the agents are strongly asynchronous, the expected time to remain in this state is finite.

We see that in all cases (except for case 4, which is meta-stable), either \( \delta \) eventually decreases (and finally becomes 0), or \( \delta^L \) diminishes to 0. In either case, since \( 0 \leq \delta^L \leq \delta \), we have that ultimately \( \delta^L = 0 \). \( \Box \)

In two or more dimensions, the idea is the same, as shown in Algorithm 3.2 and illustrated in Figure 2. First an agent chooses along which dimension (axis) to move, and then moves toward \( \text{COM} \) along that dimension. Notice that in line 4 an agent may possibly need to choose between two options. This choice may be arbitrary, since the correctness of the algorithm does not rely on it. The situation is separable. If we look at a projection of the world on the \( x \)
Algorithm 3.2 Point formation in two dimensions

1: if $|\triangle x| < 1/2$ and $|\triangle y| < 1/2$ then
2: Do not move. //Already close to COM.
3: else
4: Choose a dimension $d \in \{x, y\}$ for which $|\triangle d| \geq 1/2$
5: Move one step toward COM along $d$.

Figure 2: A single step in Algorithm 3.2. The black dots are agents, the tiny circle is COM, and the arrows signify possible movement choices for each waking agent.

axis, it will look exactly as if the agents are performing the one-dimensional algorithm along that axis. Agents which choose to move along the $y$ axis will look as if they are asleep. Thus, according to Lemma 3.1, the agents will gather in COM along each dimension. The only catch here is that we must rule out the possibility that an agent will never choose to move along a specific axis, even while being far from COM. This is proven as follows. Assume that from some point in time there exists an agent which will never choose to move along the $x$ axis, even though $|\triangle x| \geq 1/2$. Then, the agent must always (hence infinitely often) choose to move along the $y$ axis. However, according to Lemma 3.2, the agent (along with all other agents, except for, maybe, some agents which never choose to move along the $y$ axis) will eventually reach $\bar{y}$ and stay there. From that point, the agent’s only choice would be to move along the $x$ axis. However, this is a contradiction. We have thus proven the following lemma.

Lemma 3.3. Algorithm 3.2 solves the point formation problem in two dimensions.

4 Generic asynchronous phases

The point formation algorithm is simple in the sense that every time an agent wakes up, it performs the same action (move toward COM). In what follows, we will show more elaborate algorithms, which consist of several phases. In each phase the agents behave differently and achieve different goals. Since the agents are asynchronous, they are generally unable to coordinate their progress through
their local phases. Still, under certain conditions, we can maintain *global phases* through which the algorithm will progress until successful completion.

Denote by $\mathcal{C}$ the configuration space (i.e., all possible states) of the world, and denote the current configuration (at time $t$) by $c_t$. Let $A_1, \ldots, A_m$ be a series of *mutually exclusive* subsets of $\mathcal{C}$, where the initial configuration of the system is $c_0 \in A_1$. Then we can guarantee that the system will eventually reach a configuration in $A_m$, if the following conditions hold for all $k = 1, \ldots, m - 1$:

1. If $c_t \in A_k$, then $c_{t+1} \in A_{k'}$ for some $k \leq k' \leq m$.

2. If $c_t \in A_k$, then eventually $c_{t'} \notin A_k$, for some $t' > t$.

We say that the system is in *phase* $k$ if and only if $c_t \in A_k$. The first condition guarantees that the system can only remain in the same phase or advance in phases, and the second condition guarantees that each phase will eventually end. Thus, by induction, the system will eventually reach phase $m$.

This simple global view will help us design complex algorithms by “chaining” together many simple phases, making sure that for each one the above conditions hold. We can optionally impose more conditions on a given phase, like, for example:

3. For any $k < k' \leq m$, $c_t \in A_{k'}$ may hold only after each agent performed at least one action during phase $k$.

4. There exists some function $f$ of configurations in $\mathcal{C}$, which an agent can evaluate from its local view, and for all $c \in A_k$ which occur at during a run of the algorithm, $f(c)$ has the same value.

The third condition ensures that phase $k$ will occur, and each agent will view at least one configuration in $A_k$. The fourth condition ensures that all agents will be able to evaluate the same value $\varphi = f(c_t)$ when they wake up during phase $k$. Thus, a phase for which both of these conditions hold can be used to make all agents reach agreement on some value $\varphi$.

## 5 Agreement on a coordinate system

In order to form an arbitrary pattern, the first step in our strategy is to make the agents agree on a common coordinate system. Beginning with an arbitrary initial configuration (in which the agents occupy distinct points), the agents will gather at a single point (as described above) and make it their common origin. Then, the agents will perform a series of voting procedures to agree on the direction and orientation of each axis.

### 5.1 Agreement on an origin

Algorithm 5.1, which makes the agents agree on a common origin, is a simple multi-phase algorithm. In the first phase, the agents gather in a single point,
by performing Algorithm 3.2. In the second phase, upon waking up, each agent raises its flag (to publicly acknowledge the choice of common origin), and waits until all other agents raise their flags as well or leave the origin. The third phase begins when all agents have raised flags. Each agent lowers its flag and leaves the origin to its next destination.

Algorithm 5.1 Agreement on an origin

1: //Phase 1
   Perform Algorithm 3.2 until all agents are in the same point.
2: //Phase 2
   Set current position as my origin.
3: Raise flag.
4: Wait until each agent has either raised its flag or left the origin.
5: //Phase 3
   Lower flag.
6: continue to the next algorithm (Leave origin).

Lemma 5.1. Algorithm 5.1 will make the agents eventually agree on a common origin, and move out of it.

Proof. We will analyze the algorithm using the terminology presented in Section 4. First, we define a series of global configuration sets $A_1, A_2, A_3$ in Table 1. We find it convenient to define the sets in terms of external state (agents’ positions) and internal state (agents’ internal memory). Throughout this paper, we denote by $p$ the local phase of an agent, i.e., its current progress through the algorithm, as marked in the pseudo-code. For example, in Algorithm 5.1, $p = 1$ corresponds to line 1, $p = 2$ corresponds to lines 2–4, and $p = 3$ corresponds to lines 5–6.

<table>
<thead>
<tr>
<th>Configuration set</th>
<th>External state</th>
<th>Internal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>Not all agents are in the same cell.</td>
<td>$p = 1$ for all agents.</td>
</tr>
<tr>
<td>$A_2$</td>
<td>All agents are in the same cell and not all agents have raised flags.</td>
<td>$p = 1$ or 2 for all agents, where $p = 2$ iff the agent has a raised flag.</td>
</tr>
<tr>
<td>$A_3$</td>
<td>Each agent has a raised flag iff it is in the origin.</td>
<td>$p = 2$ or 3 for all agents, where $p = 2$ iff the agent is in the origin.</td>
</tr>
</tbody>
</table>

Table 1: Global phases in Algorithm 5.1

It is straightforward to show that the sets $A_1, A_2, A_3$ are mutually exclusive. We now show that they also satisfy the conditions presented in Section 4. Assume that $c_t \in A_1$. Then all agents are performing Algorithm 3.2. Thus, at time $t + 1$, they may still be scattered (so that $c_{t+1} \in A_1$), or they may become concentrated in a single cell. Their flags are lowered at this point, so in this
case \( c_{t+1} \in A_2 \). Thus, the first condition holds. According to Lemma 3.3, the agents will eventually concentrate in a single cell. So, for some \( t' > t \), \( c_{t'} \in A_2 \) and the second condition holds as well.

Now assume that \( c_t \in A_2 \). The agents with raised flags will do nothing, and those with lowered flags will raise them, if they wake up. Thus, \( c_{t+1} \in A_2 \) or \( c_{t+1} \in A_3 \) if all flags are raised at last. This will indeed eventually happen, since our assumptions guarantee that each agent will always wake up again. Thus, the first two conditions hold. Furthermore, phase 2 may end only after each agent raises its flag. Thus, condition 3 holds as well. During this phase, each agent set its current position as its new origin. Since all agents reside in the same position, condition 4 holds and the agents agree on the position of the origin. In phase 3, we assume that all waking agents leave the origin when performing the next algorithm.

Since we assume that, initially, the configuration is in \( A_1 \), then all agents will eventually agree on a common origin, the configuration will be in \( A_3 \), and the agents will leave the origin.

The same procedure as in Algorithm 5.1 can be used as a generic “glue” that properly binds any two algorithms \( A \) and \( B \) in a sequence or a “chain of phases” (See Algorithm 5.2), if the following conditions hold:

1. The agents already have a common origin.
2. In the last phase of the first algorithm \( A \), the agents gather in the origin.
3. In the first phase of the second algorithm \( B \), the agents move out of the origin.

**Algorithm 5.2 Coordination in the origin**

1: //Phase 1
   Move according to the last phase of algorithm \( A \) until all agents are in the same point.
2: //Phase 2
   Raise flag.
3: Wait until all agents have either raised flags or left the origin.
4: //Phase 3
   Lower flag.
5: continue to algorithm \( B \) (Leave origin).

The first condition is necessary for the agents to agree on the gathering location. The second condition ensures that eventually the agents actually gather and the first phase terminates. The third condition is needed, because the second phase ends (and third phase begins) when all agents are in the origin. Furthermore, if an agent does *not* leave the origin, the other agents will be unable to tell whether this agent has woken up or not, so this phase will not be terminable.
5.2 Agreement on the axes

After agreeing on an origin, the agents vote on the direction of the x axis (i.e., horizontal or vertical). The voting procedure, which is also a simple chain of phases, is presented in Algorithm 5.3 and illustrated in Figure 3.

Algorithm 5.3 Agreement on the x axis direction

1: //Phase 1
   Move to (1, 0).
2: Wait until (0, 0) is empty.
3: if there are exactly \( n/2 \) agents on each axis then
4:   Move to (0, 0).
5:   Flip the axes in my local coordinate system.
6:   Goto 1.
7: //Phase 2
   Move to (2, 0).
8: if there are less than \( n/2 \) agents on the x axis then
9:   Flip the axes in my local coordinate system.
10: Wait until all 4 points adjacent to (0, 0) are empty.
11: //Phase 3
    Return to (0, 0)
12: Continue to Algorithm 5.2.

In the first phase, the agents leave the origin. Each agent moves exactly one step in its own positive x direction (i.e., to (1, 0)), and waits until the origin is eventually empty. At this stage, each waking agent compares how many agents reside on the x axis with how many reside on the y axis. If the quantities are equal, the agent “defects” by returning to the origin, going to (0, 1) and flipping the axes in its own coordinate system. This symmetry will sustain as long as an equal number of agents defect from each “camp”. However, since the agents are strongly asynchronous, the expected time until symmetry is broken is finite. The second phase begins when finally the origin is empty and those quantities are unequal. Then each agent moves one step further from the origin (i.e., to (2, 0)) to acknowledge the choice, and sets its local x axis to coincide with the axis that contains the majority of the agents. The agent will wait there until all agents empty the cells adjacent to the origin (signifying that all agents acknowledged the choice). In the third phase, each agent simply returns to the origin and Algorithm 5.2 is performed to coordinate the next algorithm.

Lemma 5.2. If the agents already have a common origin and they are all in it, then Algorithm 5.3 will make them agree on the directions of their axes.

Proof. As in the proof of Lemma 5.1, we use the terminology of Section 4 to define the global phases of the algorithm and verify that the last phase is reached. Table 2 defines the global configuration sets that correspond to the global phases of the algorithm. We also use \( p \) to denote the local phase of an agent. It is straightforward to verify that the sets \( A_1, A_2, A_3 \) are mutually exclusive. We now show that they satisfy the conditions presented in Section 4.
Assume that \( c_t \in A_1 \). If the origin is empty and there is a “tie” (\( n/2 \) agents on each axis), waking agents will return to the origin, so \( c_{t+1} \in A_1 \). If the origin is not empty, then those outside the origin will not move, while those in it will move out of it if they wake up. Thus, \( c_{t+1} \in A_1 \) if the origin is still not empty, or if the origin becomes empty, but there is a tie. Otherwise (the origin is empty and there is no tie), \( c_{t+1} \in A_2 \). Therefore, the first condition holds. The second condition holds as well, because a non-empty origin will eventually be emptied, and due to the strong asynchrony of the agents, the expected time for a tie to hold is finite (there is a non-zero probability that a different amount of agents will “defect” from each “camp”).

Assume that \( c_t \in A_2 \). For all agents in a distance of 2 units from the origin, \( p = 2 \) holds and they will not move as long as there are agents adjacent to the origin. For those agents, \( p = 1 \) and they will move away from the origin and set \( p = 2 \) if they wake up. Thus, if all of them wake up and move, then \( c_{t+1} \in A_3 \). Otherwise, \( c_{t+1} \in A_2 \). Therefore, the first condition holds. Of course, all agents will eventually wake up and move, so the second condition holds as well. Furthermore, each agent must wake up and move to a distance of 2 units from the origin in order for this phase to end. Thus, the third condition holds. Finally, each waking agent calculates and sets its new \( x \)-axis direction as the axis which contains most of the agents. The movements during this phase...
Configuration set | External state | Internal state
--- | --- | ---
$A_1$ | All agents are in the origin and its 4 neighboring cells. If the origin is empty, then there are $n/2$ agents on each axis. | $p = 1$ for all agents.

$A_2$ | All agents are on the axes, 1 or 2 units away from the origin. There exists an agent adjacent to the origin. The numbers of agents on each axis are not equal. | $p = 1$ or 2 for all agents, where $p = 1$ iff the agent is adjacent to the origin.

$A_3$ | Not all agents are in the origin. | $p = 2$ or 3 for all agents, where $p = 3$ iff the agent is in the origin.

Table 2: Global phases in Algorithm 5.3

obviously do not affect this calculation (because they are only along the axes). Thus, the fourth condition holds and all agents agree on an $x$-axis direction. Assume that $c_t \in A_3$. If the 4 cells adjacent to the origin are empty, then waking agents 2 units away from the origin will advance to phase $p = 3$ and move into these cells on their way to the origin. Agents in the origin will not move. If those 4 cells are not empty, then waking agents in those cells will move into the origin. Other agents will not move. Either way, $c_{t+1} \in A_3$ unless all agents reach the origin (which means that the next phase in Algorithm 5.2 has been reached). Thus, the first condition holds. Obviously, all agents will eventually reach the origin, and the second condition holds.

Since that, initially, $c_t \in A_1$, we have proven that eventually the agents will agree on the directions of their axes and gather again in the origin.

At this stage, all agents’ local axes coincide, but their orientation (polarity) may differ. Thus, they perform an algorithm quite similar to Algorithm 5.3 to vote on the $x$ axis orientation. Each agent moves from the origin to $(1,0)$. Eventually all agents will reside in $(1,0)$ and $(-1,0)$. If exactly half of the agents reside in each of those points, agents will “defect” to the other point, until the symmetry is broken. Then, as above, they will all agree and move a further step from the origin (to acknowledge the agreement) and then return to the origin and perform Algorithm 5.2.

Finally, the agents repeat this voting procedure on the $y$ axis to agree on its orientation.

---

13
6 Agreement on a total ordering

After agreeing on a common coordinate system, the agents perform Algorithm 6.1 (discussed below) to agree on a total ordering. By this we mean that after the algorithm is performed, each agent will have a unique identity (or id, a natural number between 1 and n). The agents will still be anonymous (i.e., indistinguishable by their looks), but each agent will know its own id. The idea of the algorithm is that the agents “broadcast” each other what their initial position was, in terms of the newly agreed coordinate system. Since we assume that initially they occupied distinct points, we can use their initial distribution to define a lexicographic ordering of the agents (e.g. by comparing initial x coordinates and then initial y coordinates).

In the algorithm’s pseudo-code, denote an agent’s initial position by \((x_0, y_0)\). Also, define for any integer \(k\)

\[
\hat{k} = \begin{cases} 
  k & k < 0 \\
  k + 1 & \text{else}
\end{cases}
\]

Note that \(\hat{k}\) cannot be equal to 0.

**Algorithm 6.1 Agreement on a total ordering**

1: //Phase 1
   Move along the piecewise linear route \((0, 0) \rightarrow (3x_0, 0) \rightarrow (3x_0, \hat{y}_0) \rightarrow (3x_0 + 1, \hat{y}_0)\).
2: Wait until for each agent’s position \((x, y)\): \(x \equiv 1 \mod 3\) and \(y \neq 0\).
3: //Phase 2
   Calculate and memorize my id from the current configuration, as follows:
   For each agent’s position \((x, y)\) define \(x' = \lfloor x/3 \rfloor\). Sort the list of transformed coordinates \((x', y)\) lexicographically (by \(x\) and then by \(y\)). Set my id as the location of my own transformed coordinates in the sorted list.
4: Move to \((3x_0 + 2, \hat{y}_0)\).
5: Wait until for each agent’s position \((x, y)\): \(x \equiv 2 \mod 3\) or \(y = 0\).
6: //Phase 3
   Move along the piecewise linear route \((3x_0 + 2, \hat{y}_0) \rightarrow (3x_0 + 2, 0) \rightarrow (0, 0)\).
7: Continue to Algorithm 5.2.

In the first phase, each agent leaves the origin to a unique position \((3x_0 + 1, \hat{y}_0)\) (from which \((x_0, y_0)\) can be easily calculated by observing agents, as described in the code), and waits until all other agents assume position as well. Note the carefully chosen path to this destination (line 1). Along this path, only the destination \((3x_0 + 1, \hat{y}_0)\) satisfies both conditions \((x \equiv 1 \mod 3\) and \(y \neq 0\) stated in line 2. Thus, the second phase will begin only when all agents reach their destinations. Note that these destinations are distinct, since we assume that initially the agents occupied distinct points. In the second phase, each agent calculates its own unique id (as described in line 3), makes a single “acknowledgement” step (to \((3x_0 + 2, \hat{y}_0)\)), and waits for acknowledgement by all other agents (The acknowledgement step does not interfere with the id calculation which may be performed later by other agents). Only after all agents
make this step, will the third phase begin. In the third phase, all agents return
to the origin, along paths chosen so that each point along the path satisfies both
conditions in line 5. This way, each waking agent will be able to advance to this
phase and return to the origin.

**Lemma 6.1.** If all agents have a common coordinate system and are initially
in the origin, then Algorithm 6.1 will make them agree on a total ordering of
the agents.

**Proof.** As in our previous proofs above, we will analyze the algorithm using the
terminology of Section 4. Table 3 defines the mutually exclusive configuration
sets which correspond to the global phases of algorithm 6.1. We now show that

<table>
<thead>
<tr>
<th>Configuration set</th>
<th>External state</th>
<th>Internal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>All agents are on their way to their destinations (as in line 1 of the algorithm). Not all agents have reached their destination.</td>
<td>$p = 1$ for all agents.</td>
</tr>
<tr>
<td>$A_2$</td>
<td>For all agent positions $(x, y)$, $y \neq 0$ and either $x \equiv 1 \mod 3$ or $x \equiv 2 \mod 3$. There exists an agent with $x \equiv 1 \mod 3$.</td>
<td>$p = 1$ or 2 for all agents, where $p = 2$ iff the agent’s $x$-coordinate is 2 mod 3.</td>
</tr>
<tr>
<td>$A_3$</td>
<td>For all agent positions $(x, y)$, $y = 0$ or $x \equiv 2 \mod 3$. Not all agents are in the origin.</td>
<td>$p = 2$ or 3 for all agents, where $p = 2$ iff the agent is in $(3x_0 + 2, \hat{y}_0)$.</td>
</tr>
</tbody>
</table>

Table 3: Global phases in Algorithm 6.1

the conditions stated in Section 4 hold.
Assume that $c_t \in A_1$. Then those agents which reached their destinations $(3x_0 + 1, \hat{y}_0)$ will not move (because the condition stated in line 2 does not hold).
The other agents will advance toward their destinations. Thus, $c_{t+1} \in A_2$ if all
agents reach their destinations and $c_{t+1} \in A_1$ otherwise, so the first condition
hold. Obviously, all agents will eventually reach their destinations, so the second
condition holds as well.
Assume that $c_t \in A_2$. Then the agents for which $x \equiv 2 \mod 3$ will not move,
and the agents for which $x \equiv 1 \mod 3$ will move one step in the $x$ direction if
they wake up. If all do, then $c_{t+1} \in A_3$. Otherwise, $c_{t+1} \in A_2$. Therefore, the
first condition holds. Obviously, all agents will eventually wake up and make
that move, so the second condition holds as well. Furthermore, this phase ends
only after each agent has made that move (so there are no agents with $x \equiv 1 \mod 3$ left), so the third condition holds. Lastly, it is straightforward to see that
during this phase, each agent calculates the same total ordering of the agents, so
the fourth condition holds, and all agents eventually agree on the total ordering. Specifically, each agent then knows its own unique id.

Assume that \( c_t \in A_3 \). Each agent with \( p = 3 \) will move toward the origin (along the route defined in line 6 of the algorithm). Each agent with \( p = 2 \) will advance to its third local phase and move as well, since the condition in line 5 holds. This condition will also hold after the agents move, so we have \( c_{t+1} \in A_3 \) unless all agents reach the origin (which means that the next phase in Algorithm 5.2 has been reached). Thus, the first condition holds. Obviously, all agents will eventually return to the origin, and the second condition holds as well.

Since that, initially, \( c_t \in A_1 \), we have proven that eventually the agents will agree on a total ordering (i.e., acquire unique ids) and gather again in the origin.

7 Formation of an arbitrary pattern

Assume that the agents have already agreed on a coordinate system and a total ordering. Then, given any desired pattern \( F = (q_1, \ldots, q_n) \), it can be formed by simply making each agent with id \( i \) go to \( q_i \) and stay there. Eventually, when all agents reach their destinations, the pattern will be formed. So, in order to solve the formation problem, given an arbitrary initial configuration, we chain all of the algorithms presented above to make the agents first agree on a coordinate system, then agree on a total ordering and finally form the pattern \( F \). Algorithm 7.1 summarizes the steps.

Algorithm 7.1 Formation of an arbitrary pattern

Given a desired pattern \( F = (q_1, \ldots, q_n) \), perform the following chain of algorithms:

1. Agreement on origin (5.1);
2. Agreement on \( x \) direction (5.3);
3. Coordination in the origin (5.2);
4. Agreement on \( x \) orientation (5.3 variant);
5. Coordination in the origin (5.2);
6. Agreement on \( y \) orientation (5.3 variant);
7. Coordination in the origin (5.2);
8. Agreement on a total ordering — Each agent attains a unique id \( i \in (1, \ldots n) \) (6.1);
9. Coordination in the origin (5.2);
10. Formation of the pattern (Each agent \( i \) goes to \( q_i \)).

Theorem 7.1. Algorithm 7.1 solves the formation problem for any arbitrary pattern.

Proof. Each step in Algorithm 7.1 is an execution of one of the algorithms described above, which have been proven to terminate. The “stitching” is done using Algorithm 5.2. It can be verified that each step begins and ends when all agents are gathered in the origin (except, of course, the first and last steps),
as required in Section 5.1. Effectively, by chaining all of these multi-phase algorithms, we have one long chain of phases, which satisfies the conditions presented in Section 4. It can be easily verified that all other requirements for each step are also satisfied. For example, Algorithm 5.3 for agreement on the axes directions is performed when the agents already agree on an origin. Algorithm 6.1 is performed when the agents already have a common coordinate system. Lastly, line 10 is performed when the agents also have unique ids. Thus, each target point $q_i$ in the desired pattern will eventually be occupied by exactly one agent, which will remain in it indefinitely. 

Note that all stages of the algorithm end in strictly finite time, except for possible meta-stability (due to symmetry) in the point formation stage (Algorithm 3.2) and the voting stages (Algorithm 5.3). However, by our assumption of strong asynchronicity, the expected duration of this meta-stability is finite. Furthermore, if $n$ is odd, no such symmetry can occur and, therefore, the algorithm will always terminate in strictly finite time.

8 Time analysis

The time it will take our algorithms to reach completion is a random variable which heavily depends on the probability distribution of the agents' waking times, as well as their initial positions.

We chose to model the waking time distribution in a way that we think is a reasonable depiction of how real asynchronous robots behave, and is also fairly simple to analyze. We assume that, at each time step, each agent has a probability $p \in (0, 1)$ of waking up (and a probability $q = 1 - p$ of being asleep), independently of the other agents. Notice that $p = 0$ means that the agents never move, and $p = 1$ means that the agents always move, i.e., they are always synchronous. A choice of $p \in (0, 1)$, however, yields a strongly asynchronous behavior, where the agents operate with independent inaccurate “clocks” or operation cycles, with a frequency of $1/p$ time units per cycle (as the time it takes an agent to wake up is geometrically distributed with parameter $p$ and expectation $1/p$).

Regarding the initial positioning, we make no assumptions. We denote by $L$ the side length of the smallest square enclosing the agents’ initial positions.

8.1 Gathering in a point

We begin with the one-dimensional case. We analyze the running time of Algorithm 3.1 by following the structure and definitions given in the proof of Lemma 3.1. We calculate an upper bound on the expected time it takes $\delta$ to decrease by one unit. Refer to the 4 possible cases mentioned in that proof.

In case 1, there are at least $n/2$ agents in $x_{\text{max}}$, so there are at most $n/2$ agents in $x_{\text{min}}$. The expected time for a single agent to wake up is $1/p$. Thus, the expected time it would take all agents in $x_{\text{min}}$ to wake up (and make $\delta$ decrease)
is bounded from above by \( n/2p \). Case 2 is symmetric. In case 3, either \( x_{\min} \) or \( x_{\max} \) contains at most \( n/2 \), so once again, the expected time for \( \delta \) to decrease is bounded by \( n/2p \).

In case 4, the symmetry will sustain if and only if an equal number of agents wake up in each of the two occupied cells. The probability for exactly \( k \) agents to wake up in each cell is \( (p^k q^{n/2-k})^2 \). Thus, the probability for the symmetry to sustain is

\[
p_{sym}(p) = \sum_{k=0}^{n/2} \left( p^k q^{n/2-k} \right)^2 = q^n \cdot \sum_{k=0}^{n/2} \left( \frac{p}{q} \right)^{2k} = q^n \cdot \left( \frac{p}{q} \right)^{n+2} - 1 = \frac{p^{n+2} - q^{n+2}}{p - q}.
\]

The time until symmetry is broken is geometrically distributed with parameter \( 1 - p_{sym} \), and its expectation is, therefore,

\[
E_{sym}(p) = \frac{1}{1 - p_{sym}(p)}.
\] (8.1)

Predictably, \( E_{sym} \) approaches infinity as \( p \) approaches 0 (immobile agents) or 1 (synchronous agents). It is symmetric around (and minimal at) \( p = 1/2 \).

Let \( \delta = L \) initially. Then, an upper bound on the total running time \( T_{3.1} \) of Algorithm 3.1 is

\[
E(T_{3.1}) < (L - 1) \cdot \frac{n}{2p} + E_{sym}(p).
\] (8.2)

Note that if \( n \) is odd, then case 4 will never occur, and the upper bound becomes \( (L - 1) \cdot n/2p \).

In two dimensions, the agents perform Algorithm 3.2. In line 4 of the algorithm, it is not stated how the agents choose between two possible movement directions, as it is irrelevant to the algorithm’s correctness. For the simplicity of our time analysis, we assume that when an agent has two possible choices in line 4, it simply tosses a fair coin, i.e., it randomly chooses one option with a probability of 1/2. Looking at the projection of the world on one of the axes, the agents seem to perform Algorithm 3.1 with wake-up probability \( p/2 \) instead of \( p \) (for those with two possible movement choices). If case 4 (symmetry) occurs along one dimension, it can take as much as either \( E_{sym}(p) \) or \( E_{sym}(p/2) \) expected time to break the symmetry, depending on the situation along the other dimension. Thus, an upper bound on the total running time \( T_{3.2} \) of Algorithm 3.2 is

\[
E(T_{3.2}) < 2 \cdot \left( (L - 1) \cdot \frac{n}{p} + \max \left( E_{sym}(p), E_{sym}(\frac{p}{2}) \right) \right).
\] (8.3)
8.2 Agreement and formation

In Algorithms 5.1 and 5.2, the agents remain in the origin until all agents wake up and raise their flags. The expected time for single agent to wake up is $1/p$. Thus, the expected time for all agents to wake up and raise flags is bounded by $n/p$. We denote this by

$$E(T_{5.2}) < \frac{n}{q}$$ (8.4)

where $T_{5.2}$ is the running time of Algorithm 5.2. We also note that

$$E(T_{5.1}) = E(T_{3.2} + T_{5.2})$$ (8.5)

where $T_{5.1}$ is the running time of Algorithm 5.1.

In the first phase of Algorithm 5.3, the agents vote on the $x$-axis direction by moving one step out of the origin. The expected time until all agents wake up and leave the origin is bounded by $n/p$. If $n$ is even and there is a tie, waking agents will return to the origin. The expected time until at least one agent wakes up and returns is less than $1/p$. Then, as before, the expected time to empty the origin is bounded by $n/p$. In total, each time a tie is created, we “pay” at most $(n + 1)/p$ expected time. The expected number of times this will happen is exactly $E_{sym}(p)$ as in (8.1). Note that if $n$ is odd then $E_{sym}(p) = 0$. Once a decision is made (and the second phase begins), each agent moves away from the origin. The expected time until all do so is bounded by $n/p$. Finally, in the third phase, they all return to the origin. The journey is two steps long, so the expected time until all return is bounded by $2n/p$. In total, the expected running time $T_{5.3}$ of Algorithm 5.3 is

$$E(T_{5.3}) < \frac{n}{p} + E_{sym}(p) \cdot \left(\frac{n + 1}{p}\right) + \frac{n}{p} + \frac{2n}{p}$$

$$= \frac{1}{p} \cdot ((E_{sym}(p) + 4)n + E_{sym}(p) + 1).$$ (8.6)

The expected time for agreement on the each axis’ polarity is, of course, the same.

In the first phase of Algorithm 6.1, each agent travels to its destination $(3x_0 + 1, \hat{y}_0)$ along a monotone path, so its length is the Manhattan distance (1-norm) of the destination from the origin. Given $L$, the side length of the smallest square enclosing the agents’ initial positions, the maximum possible absolute value of each of an agent’s initial coordinates $x_0, y_0$ is $L$. Thus, each agent’s path to its destination is

$$\|(3x_0 + 1, \hat{y}_0)\|_1 < 3L + 1 + L + 1$$

$$= 4L + 2.$$

Since the expected time for an agent to advance a single step is $1/p$, the expected time for an agent to reach its destination is bounded by $(4L + 2)/p$, and the expected time for all agents to do so is bounded by $(4L + 2)\cdot n/p$. 

19
In the second phase, each agent must wake up and move one step (to \((3x_0 + 2, y_0)\)). The time for all agents to do so is bounded by \(n/p\). Finally, in the third phase, each agent returns along a monotone path from \((3x_0 + 2, y_0)\) to the origin. Thus, the expected time for all agents to return is bounded by \((4L + 3) \cdot n/p\).

In total, the expected running time \(T_{6.1}\) of Algorithm 6.1 is

\[
E(T_{6.1}) < \frac{n}{p} (4L + 2) + \frac{n}{p} (4L + 3) = \frac{n}{p} (8L + 6) .
\]  

(8.7)

Finally, the time \(T_{\text{form}}\) it takes the agents to form the desired pattern \(F = (q_1, \ldots, q_n)\) in line 10 of Algorithm 7.1 depends on the distance of the destination points from the origin. Define

\[
|F| = \sum_{i=0}^{n} \|q_i\|_1 .
\]

Then, the expected time for all agents to reach their destinations (starting from the origin) is

\[
E(T_{\text{form}}) < |F| / p .
\]  

(8.8)

The total expected running time \(T_{7.1}\) of Algorithm 7.1 is the sum of all its parts:

\[
E(T_{7.1}) = E(T_{5.1} + 4 \cdot T_{5.2} + 3 \cdot T_{5.3} + T_{6.1} + T_{\text{form}}) < \frac{1}{p} (n \cdot (10L + 3E_{sym}(p) + 21) + 3E_{sym}(p) + 3 + |F|) + 2 \text{ max} \left( E_{sym}(p), E_{sym}(\frac{p}{2}) \right) = O \left( \frac{n \cdot (L + E_{sym}(p)) + |F| + E_{sym}(\frac{p}{2})}{p} \right) .
\]  

(8.9)

If \(n\) is odd, then \(E_{sym} \equiv 0\), and

\[
T_{6.1} = O \left( \frac{nL + |F|}{p} \right) .
\]

So, setting aside the symmetry breaking time, we see that, predictably, Algorithm 7.1 is at most linear with the number of agents \(n\), their initial distribution breadth \(L\), their operating rate (or “speed”) \(1/p\), and the size of the desired formation \(|F|\). A finer analysis may reveal that the dependence on \(n\) is actually sub-linear. We only provided a rough upper bound, by simply taking the time required for a single agent to perform an action and multiplying it by the number of agents performing it, as if the agents are acting sequentially. In other words, we did not take into full account the parallelism of the agents’ actions.

When \(n\) is odd, the higher \(p\) is the better. In fact, \(p = 1\) yields the optimal results in this case. When \(n\) is even, however, there is a chance that the agents
get “trapped” in symmetry in several phases of the algorithm. However, as long as $0 < p < 1$, the symmetric states are meta-stable, and the symmetry is bound to break sooner or later. Since the breaking time is geometrically distributed, it becomes negligible, given that $p$ is sufficiently far from 1. A choice of “almost 1”, say $p = 0.95$ (which implies $E_{\text{sym}}(0.95) = 1.006$ for $n = 100$), is enough to make the expected running time of the algorithm very short, including the time spent in attempting to break any possible symmetry.

9 Extensions

We now present two natural extensions to the presented solution. First, we explain how the algorithm can be extended to work on a hyper-grid of any dimension $k > 1$ (Curiously enough, we don’t have a solution for one dimension). We denote by $x^1, \ldots, x^k$ the axes (or dimensions) of the hyper-grid. Second, we show how the agents can form a sequence of arbitrary patterns.

9.1 Extending to higher dimensions

The basic structure of the algorithm doesn’t change. First, the agents gather in a point. Algorithm 3.2 is naturally extended by letting the agent choose any movement direction $d \in \{x^1, \ldots, x^k\}$ for which $|\Delta d| \geq 1/2$. After agreeing on the origin, the agents perform Algorithm 5.3 $k - 1$ times (instead of just once). First, they agree on the direction of $x^1$, then on the direction of $x^2$ and so on. The vote in this case is not binary, so there may be several majority groups. Upon waking up, each agent from one of these groups will “defect” to another majority group, until there is a single majority group left, which will then determine the direction of the axis. After the directions of $x^1, \ldots, x^{k-1}$ are chosen, the direction of $x^k$ is, of course, implied. Then the agents vote on the orientation of each axis, just as they did for $x$ and $y$ in two dimensions (The vote is binary).

Algorithm 6.1 is extended as follows. Each agent leaves the origin along the following piecewise linear path (line 1):

$$(0, \ldots, 0) \rightarrow (3x^1_0, 0, 0, \ldots, 0) \rightarrow (3x^1_0, x^2_0, 0, \ldots, 0) \rightarrow (3x^1_0, x^2_0, x^3_0, \ldots, 0) \rightarrow \cdots \rightarrow (3x^1_0, x^2_0, x^3_0, \ldots, x^k_0) \rightarrow (3x^1_0 + 1, x^2_0, x^3_0, \ldots, x^k_0).$$

The condition in line 2 becomes:

$x^1 \equiv 1 \mod 3$ and $x^j \neq 0$ for all $j = 2, \ldots, k$, for each agent’s position $(x^1, \ldots, x^k)$.

As in the two dimensional case, of all points along the path, only the destination point satisfies the condition. The rest of the analogy to the two dimensional case is straightforward.
9.2 Formation of a sequence of patterns

In many major sport event openings (e.g., in the Olympic games), many children run around the stadium, and create a sequence of huge spatial patterns in a coordinated fashion. The agents can handle a similar feat, using the simple approach of chained phases.

Given a sequence of patterns \((F_1, \ldots, F_m)\), (where \(F_k = (q_1^k, \ldots, q_n^k)\) for each \(k \in \{1, \ldots, m\}\) is a list of \(n\) coordinates), we predetermine for each agent \(i\) a path along which the agent will march between the points \((q_1^i, \ldots, q_m^i)\). Denote by \(F_0\) the initial pattern where all agents are in the origin. Algorithm 9.1 solves the problem by making each agent \(i\) move from \(q_{k-1}^i\) to \(q_k^i\), and wait until all others do so as well, in each phase \(k \in \{1, \ldots, m\}\).

**Algorithm 9.1 Formation of a sequence of patterns**

1. Perform all agreement stages as in Algorithm 7.1.
2. for \(l = 1\) to \(m\) do
   3. Move along the predetermined path \(q_{l-1}^i \rightarrow q_l^i\).
   4. Wait in \(q_l^i\) until all agents are in \(\bigcup_{j=1}^n [q_{l-1}^j \rightarrow q_{l+1}^j]\).

In order for the agents to advance correctly through the phases and form all desired patterns, we must ensure that they do not get “confused”. Denote by \([q_i^k \rightarrow q_{i+1}^k]\) all the points along agent \(i\)’s path from \(q_i^k\) to \(q_{i+1}^k\), inclusive. Denote by \([q_i^k \rightarrow q_{i+1}^k]\) the same set of points, excluding \(q_{i+1}^k\). We present and prove a sufficient condition for the correctness of the algorithm.

**Theorem 9.1.** Given a series of desired patterns \((F_1, \ldots, F_m)\), if for all agents \(i, j \in \{1, \ldots, n\}\) and phases \(k \in \{1, \ldots, m-1\}\),

\[
[q_{i-1}^k \rightarrow q_i^k] \cap [q_j^k \rightarrow q_{j+1}^k] = \emptyset,
\]

then by performing Algorithm 9.1, the agents will sequentially form all desired patterns, that is, for each pattern \(F_k\) there exists a time step \(t_k\) where the agents form \(F_k\).

**Proof.** In plain words, the condition states that the agents’ paths in consequent phases must not intersect. We assume that the agents have already agreed on a coordinate system and a total ordering, and are gathered in the origin, performing Algorithm 5.2, ready to spread out and form \(F_1\).

As before, we begin our proof by defining the global configuration sets which correspond to the algorithm’s phases. For each \(k \in \{1, \ldots, m\}\), we define \(A_k\) to be the set of all configurations where each agent \(i\) is in \([q_{i-1}^{k-1} \rightarrow q_i^k]\) and its loop counter \(l\) is equal to \(k-1\) if it is still in \(q_{i-1}^{k-1}\), or \(k\) otherwise. Furthermore, there exists an agent \(i\) which is in \([q_{i-1}^{k-1} \rightarrow q_i^k]\), i.e., not in \(q_i^k\).

It is easy to see that all sets \(A_1, \ldots, A_m\) are mutually exclusive. Consecutive sets in this series are mutually exclusive due to the assumption stated in the theorem. Otherwise, two sets in the series are mutually exclusive due to the different internal states (the loop counters \(l\)) of the agents.

Assume that \(c_t \in A_k\). Then, for each agent \(i\) there are three possible cases:
1. The agent is still in $q_i^{k-1}$ and with $l = k-1$. It will observe that all agents are in $\bigcup_{j=1}^n [q_j^{k-1} \rightarrow q_j^k]$ and will, therefore, advance to its next local phase $l = k$ and move to the next point along the route $q_i^{k-1} \rightarrow q_i^k$.

2. The agent is neither in $q_i^{k-1}$ nor in $q_i^k$. It will keep on moving along the route $q_i^{k-1} \rightarrow q_i^k$.

3. The agent is in $q_i^k$. Since there exists an agent $i'$ which is in $[q_i^{k-1} \rightarrow q_i^k)$, and according to our assumption $[q_i^{k-1} \rightarrow q_i^k) \cap \bigcup_{j=1}^n [q_j^k \rightarrow q_j^{k+1}] = \emptyset$, the condition in line 4 of the algorithm is not satisfied, and the agent will wait in place.

All agents remain on their respective routes. Therefore, the configuration remains in $A_k$ at time $t+1$, unless all agents arrive to their $k$-th destinations. In this case $c_{t+1} \in A_{k+1}$. Furthermore, all agents keep moving until they eventually reach their $k$-th destinations. Thus, the first and second conditions stated in Section 4 hold. Specifically, at the moment of transition from phase $k$ to phase $k+1$, all agents are in their $k$-th destinations, i.e., pattern $F_k$ is formed. Since, initially, $c_t \in A_1$, we have that for each $k \in \{1, \ldots, m\}$ exists a time step $t_k$ where the $k$-th phase is reached (i.e., $c_{t_k} \in A_k$), and pattern $F_k$ is formed. \(\square\)

Note that Theorem 9.1 only guarantees the correctness of Algorithm 9.1, given a set of patterns and the agents’ routes. It does not tell how the agents’ routes are planned, nor does it guarantee the existence of such routes in the general case.

10 Conclusions

We have presented a distributed algorithm for the formation of any arbitrary pattern by anonymous homogenous mobile agents on a grid, with no initial agreement on a coordinate system, using only movement and observation for tele-interaction, and minimal anonymous zero-range communication ability. We first showed that the intuitive idea of simply moving the agents towards the collective’s center of mass indeed makes them gather in a single point. Then we used the approach of chaining together several simple procedures to present an algorithm, which makes the agents agree on a coordinate system as well as a total ordering of the agents, and form the desired pattern. Our results do not contradict the impossibility results obtained in [8] (for a continuous plane world model), because we assumed that the agents are strongly asynchronous, hence symmetry will be broken within a finite expected time. In our opinion, this assumption is very realistic when dealing with real autonomous robots, as the probability for such robots to indefinitely remain synchronous is nil.

We extended our solution to work in a hyper-grid of any number of dimensions $k > 1$. We also showed how the agents can form an arbitrary sequence of patterns instead of just one. In a forthcoming paper we will show how the simple ideas expressed in this paper are used to solve the formation problem for
any arbitrary pattern in the *continuous* plane (or any hyper-plane), even with range-limited visibility of the agents.

Open questions for further research include solving the problem

- *without* the use of any zero-range communication ability;
- with range-limited visibility or in the presence of sensory and/or control errors;
- considering other realistic aspects of robotics, such as occlusions, collisions, positioning errors etc.
- with completely *oblivious* (i.e. memoryless) agents.

**References**


