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, researchers are trying to design and analyze distributed systems of multiple mobile robots, hoping to gain similar advantages [2, 8].

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, e.g., when deploying a group of mobile robots or a self-arranging array of sensors. Also, by forming a pattern, tasks can be allocated, groups created, leaders elected etc. This problem was considered by many, including [1, 5, 6, 10]. Most of these works address engineering aspects and design behaviors that seem to work well in simulation or in real robot teams.

Suzuki et al [9] 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, what kinds of spatial patterns can they be programmed to create? One of their main results is that 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 happen to be distributed in a symmetrical pattern, and also may happen to be perfectly synchronous forever. Since they all perform the same algorithm, they always have the same view and make the same movements, so the symmetry never breaks. Several other researchers took a similar approach. Notably, Flocchini et al discussed similar questions, using a model which differs mainly w.r.t. the asynchronicity of the agents' actions, and concentrates on oblivious agents [4]. They showed analogous impossibility results for asymmetric patterns. Defago et al [3] discussed the formation of a circle by oblivious agents.

The idea of implicit communication through movement is not new. In fact, it is common in nature. A fine example is that of scouting bees that communicate their findings — the location of nectar — through dance-like motions.

In this work, we deal with similar questions, but assume that the world is a (discrete) *grid*, instead of the continuous plane. We believe that, in real robots, it is reasonable to assume that asynchronous autonomous robots are unlikely to remain synchronous for a long time. By adding this assumption to our model, we are able to give a rather strong possibility result — a simple "generic" algorithm which solves the formation problem for any arbitrary pattern, within a finite expected time. It consists of several "bee dances", through which the agents communicate and reach agreement prior to moving into formation.

Due to limited space, we omitted the formal proofs from most of our claims. Detailed proofs as well as time analysis can be found in [7].

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* (It has no control over this). A sleeping agent does nothing and sees nothing. When an agent wakes up, it sees the locations of all agents, and may move to an adjacent point on the grid (i.e., a 4-neighbor), according to its algorithm. The algorithm's input is the agent's current view of the world and possibly some private internal memory. 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 sleeps 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 observation. 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 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 has no way of knowing if the others have already witnessed the gathering (since the sleeping time of an agent is unbounded).

2.2. Problem definition

Given a *pattern* (a collection of coordinates) $F = (q_1, \ldots, q_n)$ on the grid, the *Formation Problem* is the problem of finding a distributed algorithm, such that from any initial distribution of the agents, they will eventually arrange themselves in the desired pattern F. Note that in [9], "eventually" would mean "in strictly finite time". In our context, it is "in finite expected time", which is a weaker condition. This interpretation is equivalent to Suzuki et al's definition of the *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. Also, a "mirror image" of the desired pattern may be formed, since there is no initial agreement on the handedness (or "chirality") of the coordinate system.

3. Point formation

In order to make 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 $\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i$).

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 $\triangle x = \bar{x} - x_t$. 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 $|\triangle x| < 1/2$ 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.*

The lemma is proven by showing that, in any case, the support of the agents' positions must eventually shrink. The strong asynchronicity assumption is used to deal with the case where all agents reside in two adjacent cells, with $n/2$ agents in each cell. This symmetry breaks as soon as an unequal number of agents wake up in each cell. As there is some probability $\varepsilon > 0$ for this, the expected time until symmetry is broken is $1/\varepsilon$.

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., never moves 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.*

In two or more dimensions, the idea is the same, as shown in Algorithm 3.2 and Figure 1. First, an agent chooses along which dimension (axis) to move, and then moves toward *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.

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 $d$.
```

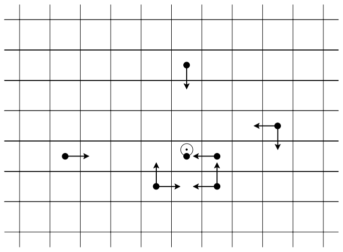

Figure 1. A step in Algorithm 3.2. The tiny circle is *COM*. The arrows signify possible movement choices for each waking agent (dot).

The situation is separable. If we view the projection of the world on the x 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 look as if they are asleep. The only catch is that we must rule out the possibility that an agent never chooses to move along a specific axis. This is proven as follows. Assume that from some point in time there exists an agent which never chooses 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) eventually reaches $\bar{y}$ and stays there. From that point, the agent's only choice would be to move along the x axis. 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. 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, the agents gather at a single point (as described above) and make it their common origin. Then, they perform a series of little "dances", to vote and agree on the direction and orientation of each axis.

4.1. Agreement on an origin

Algorithm 4.1 makes the agents agree on a common origin, as follows. First, the agents gather in a single point, by performing Algorithm 3.2. Second, upon waking up, each agent makes this point its new origin, raises its flag, and waits until all other agents raise their flags as well or leave the origin. Third, once all agents have raised flags, each agent lowers its flag and leaves to its next destination.

Algorithm 4.1 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 4.1. *Algorithm 4.1 will make the agents eventually agree on a common origin, and move out of it.*

The same procedure as in Algorithm 4.1 can be used as a generic "glue" that properly chains any two algorithms A and B in a sequence (e.g., as in Algorithm 6.1), 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; and (3) in the first phase of the second algorithm B, the agents move out of the origin.

4.2. 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 is presented in Algorithm 4.2 and Figure 2.

First, each agent leaves the origin one step in its own positive x direction (i.e., to $(1, 0)$), and waits until the origin is eventually empty. Then, it 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 of its own coordinate system. This symmetry sustains 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. Second, when finally the origin is empty and those quantities are unequal, each agent moves one step further from the origin (i.e., to $(2, 0)$) to

Algorithm 4.2 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 4.1.
```

"acknowledge" the choice, and sets its local x axis to coincide with the axis that contains the majority of the agents. The agent waits there until all agents acknowledged as well. Finally, each agent simply returns to the origin and Algorithm 4.1 is performed to coordinate the next algorithm.

Lemma 4.2. *Let all agents have a common origin and reside in it. Then Algorithm 4.2 will make them agree on the directions of their axes.*

At this stage, all agents' local axes coincide, but they still need to agree on their *orientations* (polarities). They do so by performing additional dances, quite similar to Algorithm 4.2, once for each axis.

5. Agreement on a total ordering

After agreeing on a common coordinate system, the agents perform another "dance" (Algorithm 5.1) 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 is that the agents "broadcast" to 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 total ordering of the agents.

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.

First, 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 others), and waits until all other agents do so as well. Note the carefully chosen path to this destination (line 1), along which only the destination $(3x_0+1, \hat{y_0})$ satisfies both conditions stated in line 2. Thus, the second phase begins only when all agents reach their destinations. Then, 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 (This step does not interfere with the id calculation to be done by other agents). Finally, all agents return to the origin, along paths chosen so that each point along the path satisfies both conditions in line 5 (so that each waking agent will be able to advance to this phase and return to the origin).

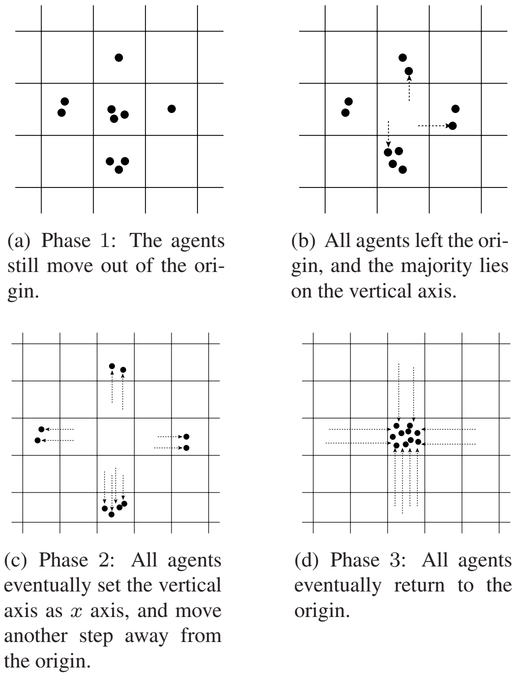
(a) Phase 1: The agents still move out of the origin.
(b) All agents left the origin, and the majority lies on the vertical axis.
(c) Phase 2: All agents eventually set the vertical axis as x axis, and move another step away from the origin.
(d) Phase 3: All agents eventually return to the origin.

Figure 2. An illustration of Algorithm 4.2

Lemma 5.1. *Let all agents have a common coordinate system and reside in the origin. Then Algorithm 5.1 will make them agree on a total ordering of the agents.*

6. Formation of an arbitrary pattern

Assuming that the agents have a coordinate system and a total ordering, any given pattern $F = (q_1, \ldots, q_n)$ can be formed, by simply making each agent with id i go to q_i and stay there. So, we solve the formation problem, given an arbitrary initial configuration, by chaining 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 6.1 summarizes the steps.

Algorithm 5.1 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 my id as follows: For each agent's position $(x,y)$ define $x' = \lfloor x/3 \rfloor$. Sort the list of transformed coordinates $(x',y)$ lexicographically. 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 4.1.
```

Algorithm 6.1 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 (3.2, 4.1);
2: Agreement on $x$ direction (4.2);
3: Coordination in the origin (4.1);
4: Agreement on $x$ orientation (4.2 variant);
5: Coordination in the origin (4.1);
6: Agreement on $y$ orientation (4.2 variant);
7: Coordination in the origin (4.1);
8: Agreement on a total ordering — Each agent attains a unique id $i \in (1, \ldots n)$ (5.1);
9: Coordination in the origin (4.1);
10: Formation of the pattern (Each agent $i$ goes to $q_i$).
```

Theorem 6.1. *Algorithm 6.1 solves the formation problem for any arbitrary pattern.*

Note that all stages of the algorithm end in strictly finite time, except for possible meta-stability (due to symmetry) in the point formation (Algorithm 3.2) and the voting (Algorithm 4.2) stages. However, by our assumptions, 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.

7. 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 movements for tele-interaction, and minimal zero-range communication ability. We showed that the intuitive idea of moving the agents towards *COM* indeed makes them gather in a single point. We presented simple "bee dances" which make the agents agree on a common coordinate system and a total ordering, and finally form the desired pattern.

In our opinion, the assumption that the agents cannot remain synchronous for a long time is very realistic when dealing with real autonomous robots. This enables us to resolve the symmetry-breaking problem encountered by other researchers, and present a simple and generic solution to the formation problem. In a forthcoming paper, we will use similar ideas to solve the formation problem in the continuous plane, even with range-limited visibility of the agents.

Open questions for further research include solving the problem *without* any use of 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 and so on.

References

[1] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Trans. on Robotics and Automation*, 14(6):926–939, December 1998.

[2] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, March 1997.

[3] X. Defago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proc. of the 2nd ACM Intl. workshop on Principles of mobile computing*, pages 97–104. ACM Press, 2002.

[4] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Proc. of the 10 Annual Intl. Symposium on Algorithms and Computation (ISAAC '99)*, volume 1741 of LNCS, pages 93–102, 1999.

[5] J. Fredslund and M. J. Mataric. Robot formations using only local sensing and control. In *Proc. of the Intl. Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001)*, pages 308–313, Banff, Alberta, Canada, 2001.

[6] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura. Self-organizing formation algorithm for active elements. In *Proc. of 21st IEEE Symposium on Reliable Distributed Systems*, pages 416–421, 2002.

[7] N. Gordon, I. A. Wagner, and A. M. Bruckstein. Discrete bee dance algorithms for pattern formation on a grid. Technical Report CIS-2003-03, CS Department, Technion IIT, Israel, 2003.

[8] M. Resnick. *Turtles, Termites and Traffic Jams*. MIT Press, 1994.

[9] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[10] J. Wang and G. Beni. Cellular robotic systems: Self-organizing robots and kinetic pattern generation. In *IEEE Intl. Workshop on Intelligent Robots*, pages 139–144, October 1988.