(1)

PROCEEDINGS

THIRTY-FIRST ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

Dilip V. Sarwate
Paul Van Dooren
Conference Co-Chairs

Conference held
September 29 - October 1, 1993
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

Formation and Agreement Problems for Anonymous Mobile Robots*

Ichiro Suzuki † Masafumi Yamashita ‡
† Department of Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee, Milwaukee, WI 53201
‡ Department of Electrical Engineering, Faculty of Engineering
Hiroshima University, Higashi-Hiroshima 724, Japan
Email: suzuki@cs.uwm.edu, mak@se.hiroshima-u.ac.jp

Abstract A system consisting of multiple mobile robots in which the robots can see each other by their eye sensors but are not equipped with any communication system, can be viewed as a distributed system in which the components (i.e., robots) can "communicate" with each other only by means of their moves. We use this system to investigate, through a case study of a number of problems on the formation of geometric figures in the plane, the power and limitations of the distributed control method for mobile robots. In the distributed control method, at every tick of its local clock, each robot observes the positions of all the robots and moves to a new position determined by the given algorithm. The robots are anonymous in the sense that they all execute the same algorithm and they cannot be distinguished by appearances. The local clocks of the robots are not necessarily synchronized, and initially, the robots do not have a common x-y coordinate system. The problems we discuss include (1) converging the robots to a single point, (2) moving the robots to a single point, (3) agreement on a single point, (4) agreement on the unit distance, (5) agreement on direction, and (6) leader election. We develop algorithms for solving some of these problems under various conditions. Some impossibility results are also presented.

1 Introduction

In the last several years, interest in the distributed control method for multiple mobile robots has increased considerably [1, 2, 5, 7]. The main idea of the method is to let each robot execute a simple algorithm and determine its movement adaptively based on the observed movement of other robots, so that the robots as a whole group will achieve the given goal. This approach has been shown to be very promising for the generation of certain patterns and collision avoidance. In the earlier works on distributed robot control, the main emphasis is on the development of heuristic algorithms for various problems and the presentation of simulation results, and in many cases, formal discussions on the correctness and performance of the algorithms are not given [1] [5].

A robot system in which the robots can communicate with each other by radio, such as a system of radio-controlled vehicles or spaceships, can be considered as a distributed system whose communication topology is a complete graph. Therefore, such systems can be analyzed using the standard techniques developed for distributed computing systems (although such analyses are by no means easy). In this paper, we consider a system consisting of multiple

*This work was supported in part by the National Science Foundation under grants CCR-9004346 and IRI 9307506

mobile robots in which the robots can see each other by their eye sensors, but they are not equipped with any communication system. We view this system as a distributed system in which the components (i.e., robots) can "communicate" with each other only by means of their moves. Clearly this system is a suitable model for studying the distributed control method mentioned in the previous paragraph. As the reader might expect from the assumptions we make about the robots below, the study reveals delicate interplay of a number of key concepts of distributed computing, such as synchrony and asynchrony, communication, termination detection, self stabilization, anonymity of processors, and knowledge (in a casual sense).

A basic problem for such a robot system is to design an algorithm such that, if all the robots execute it individually, then the robots as a whole group will eventually form the given geometric figure, such as a circle and a line segment [3, 5, 6]. The main goal of this paper is to present some theoretical results related to this problem. The results presented here provide useful insights that will help us to answer certain fundamental questions, such as whether the given algorithm really solves the given problem and, for that matter, whether the given problem is solvable at all in a strict sense by a distributed algorithm. This work is a step toward the ultimate goal of determining exactly what class of problems are solvable in a distributed manner.

We assume that each robot is a mobile processor having an eye sensor which, at every tick of its local clock, observes the positions of all the robots (including itself) in terms of its own local x-y coordinate system, and moves to a new position determined by the given algorithm. To simplify the discussion, in this paper we assume that a robot can move to its new position instantaneously. This assumption helps us to bring out the fundamental issues of the problem, and still, many of the techniques and results we obtain for this simplified case seem to apply (with some modifications) to the case when the move of a robot is not instantaneous. (We plan to report on the noninstantaneous case in a future paper.)

The robot (or the algorithm it uses to compute the new position) is *oblivious* if the new position is determined only from the positions of the robots observed at that time instant. Otherwise, it is not oblivious, and the new position may depend also on the observations made in the past. The local clocks of the robots are not necessarily synchronized, i.e., they may tick at different rates. It is assumed that initially, the robots do not have a common x-y coordinate system. So the local x-y coordinate systems of two robots may not agree on any of the following: the location of the origin, the unit distance, and the direction of the positive x-axis. Finally, we assume that the robots are anonymous, in the sense that they all use the same algorithm for determining the next position, and they cannot be distinguished by their appearances. This implies, for example, that depending on what movements are allowed, a robot that observes other robots at two consecutive clock ticks may not be able to tell which robot has moved to which position between the two ticks.

Under these assumptions, we first consider the problem of converging all the robots toward a single point. (The process of convergence need not terminate in finite steps.) Note that since the robots do not have a common x-y coordinate system, we cannot simply use an algorithm such as "move toward point $(0,0)$". For this problem, we give an oblivious algorithm and then discuss the subtlety of the problem by showing how certain minor changes in the algorithm or the assumption affect the possibility of achieving the goal. Also, we show that the set of geometric figures realizable by any deterministic algorithm must necessarily include the configuration in which all robots are located at the same position.

Next, we investigate the problem of having the robots agree on a common x-y coordinate system. Clearly, such an agreement can greatly reduce the complexity of subsequent motion coordination algorithms. The problem consists of three subproblems, agreement on the origin, agreement on the unit distance, and agreement on the direction of the positive x-axis. We first consider a related problem of moving two robots to a single point *in finite steps* (such a problem is called a *formation problem*, in contrast to a convergence problem) and show that this problem can be solved by a nonoblivious algorithm but not by any oblivious algorithm. (The corresponding convergence problem can be solved by an oblivious algorithm, as we stated

1. $I := \emptyset$;
2. **repeat**
3. Wait until c_r ticks; Let k be the new value of c_r;
   /* Start Move Action */
4. Let P_r be the multiset of positions of the robots, excluding r, at local time k, given in terms of Z_r;
5. $I := I \cup \{(p,k) | p \in P_r\} \cup \{(p^*,k)\}$, where p^* is the current position of r;
6. Randomly select an element q from $\psi(I)$;
7. Move to q;
   /* End Move Action */
8. **until** forever.

Figure 1: Behavior of robot r.

above.) Since two robots can agree on the origin if they can move to a single point, this result shows that agreement on the origin for two robots is solvable by a nonoblivious algorithm. Moreover, we show that agreement on the origin is solvable by a nonoblivious algorithm even for the case $n > 2$, where n is the number of robots. We also show that agreement on the unit distance is solvable by a nonoblivious algorithm if either $n = 2$, or $n > 2$ and no two robots are "clones" (to be defined later) of each other. On the other hand, we show that agreement on direction is not solvable even if $n = 2$ and the clocks are synchronized. This last result shows that the robots cannot agree on a common x-y coordinate system in general.

Finally, we consider the case in which the robots have a sense of direction (i.e., the direction of the positive x-axis is the same for all robots). For this case, we show that the robots can agree on a common x-y coordinate system and elect a leader, provided that no "clones" exist. We can show that once a unique leader is elected, the robots can be moved to form any geometric figure.

It can be shown that if the local clocks are synchronized, then the robots can easily communicate with each other by means of the distances of their moves, once they agree on the unit distance. Therefore, many problems on synchronized robot systems can be reduced to the corresponding problems on anonymous complete networks, and can be analyzed using the techniques developed, for example, in [8]. Details can be found in the full paper.

We present necessary definitions and basic assumptions in Section 2. Section 3 discusses the problem of converging the robots to a point. Agreement on a common coordinate system is discussed in Section 4. In Section 5 we consider the case in which the robots have a sense of direction. Concluding remarks are found in Section 6. In this abstract, we present only the key ideas. All the details and the missing proofs can be found in the full paper.

2 Definitions and Basic Assumptions

We briefly formalize the problem described in Section 1.

A robot r is a mobile processor with sufficiently large memory space, an artificial eye sensor, a local x-y coordinate system Z_r, and a local clock c_r. The behavior of r is given in Figure 1. Local clock c_r is an integer counter whose contents, initially 0, continue to increase by 1 (i.e., the clock "ticks") at infinitely many (unknown) real time instants. We assume that at real time 0, r is at the origin $(0,0)$ of Z_r, executes line 1 instantaneously, and waits (in line 3) for its local clock c_r to tick and reach the value 1. The robots have no access to real time. The symbol I, called the *configuration* of the memory of r, is a multiset[1] containing elements of the form

[1] For simplicity, we assume that a robot is a point, and hence two or more robots can occupy the same position

1. (p, k), where p is the position of a robot other than r that r observed at local time k, and

2. $\langle p, k \rangle$, where p is the position of r at local time k.

Here, p is given in terms of Z_r. The function ψ (line 6) is called an *algorithm* for r, where for each configuration I, $\psi(I)$ is the set of possible next positions of r (given in terms of Z_r) when r "knows" I. If $\psi(I)$ is a singleton for all I, then ψ is said to be *deterministic*. Otherwise, r randomly selects its next position from $\psi(I)$ (line 6), and hence ψ is a *randomized* algorithm. The fact that I contains elements of the form (p, k) indicates that r always knows its current position in terms of Z_r. Also, note that I contains only the *positions* of the robots that r has observed. So r never observes, for example, the velocity of other robots. Note that the robots are anonymous in the following sense: (1) the initial configuration $\emptyset$ and function ψ are common to all the robots, (2) the identifier "r" of robot r is not an argument of ψ, and (3) P_r contains only the positions of the robots (but not their identities).

We assume that different robots may have different coordinate systems, i.e., $Z_r \neq Z_s$ for some robots r and s. Local clocks of two robots are said to be *synchronized* if they always tick simultaneously; otherwise, they are *asynchronous*. Unless otherwise stated, we assume that the local clocks are not necessarily synchronized. Therefore, even though all the robots execute line 1 at real time 0, they may not execute line 4 for the first time simultaneously. This means that the robots may not be able to obtain a consistent view of their initial distribution. In fact, we will see that the difficulty of obtaining a consistent view of the system is a source of some of the technical problems in designing correct algorithms. For example, if all the robots know that they can observe their initial distribution simultaneously, then they can adopt the center of gravity of their distribution as the common origin, and the minimum distance between any two robots as the common unit distance. So the robots can move to a point on the circumference of the unit circle centered at the origin, and form an approximation of a circle.

Robot r is said to be *active* when it executes lines 4–7; otherwise, it is *inactive*. By the definition of c_r, r becomes active infinitely many times. As we stated in Section 1, we assume that a robot can move to any position instantaneously, and thus the time it takes to execute lines 4–7 is negligibly small.

Using this framework, it is possible to discuss the situation in which some robots are added and/or removed from the system dynamically. This can be done by assuming that a robot becomes visible (or invisible) when it is added (or removed) from the system. Under this assumption, all the algorithms we present in this paper can easily be modified to work correctly even if the number of robots changes a finite number of times. (See the full paper for details.) Algorithms having this property can be viewed as self-stabilizing algorithms, since they solve the given problem in the presence of transient failures. This is an advantage of the distributed control method. In the centralized method, the entire system can crash if the robot controlling all other robots becomes faulty (and is removed).

Fix an x-y coordinate system Z. (The robots have no information on Z.) Let π be a predicate over the set of multisets of points (given in terms of Z) that is invariant under any motion (i.e., rotation and parallel transformation) and uniform scaling. For example, π might be true iff the given points are on the circumference of a circle or on a line segment. For such π, we consider two types of problems, the *convergence problem* and the *formation problem*. In the convergence problem, the goal is to design an algorithm ψ such that, if all the robots execute the instructions using ψ, then the positions of the robots converge to a multiset of points (with respect to the absolute coordinate system Z) satisfying π, regardless of the initial distribution of the robots. The formation problem is similar, except that the robots must reach some points satisfying π in finite steps. Since the robots have no knowledge of the coordinate system Z, all we can expect is to have the robots converge to or form a figure which is similar to the given

simultaneously. If robot r observes at local time k that both robots r_1 and r_2 are at position p, then I should contain (p, k) for each of r_1 and r_2

goal figure. The restriction on π stated above was introduced for this reason. All predicates we discuss in the following are assumed to satisfy this condition.

3 Converging to a Point

Define a predicate π_{point} by $\pi_{point}(p_1, \ldots, p_n) = true$ iff $p_i = p_j$ for any $1 \leq i, j \leq n$, where $n \geq 2$ is the number of robots. In this section we discuss the convergence problem for π_{point}, which we call POINT. That is, POINT is the problem of converging all the robots to a single point. We present three correct oblivious algorithms and one incorrect algorithm. All of these are deterministic algorithms. We also discuss a limitation of deterministic algorithms.

Our first algorithm, ψ_{point}, simply moves each robot r to the midpoint of its current position and the position of a robot that is currently furthest from r. Formally, let p^* and P_r be the current position of r and the multiset of positions of all other robots at local time k, respectively, as in lines 3 and 4 of Figure 1. Since ψ_{point} is oblivious, "$\psi_{point}(I)$" can simply be written as "$\psi_{point}(P_r, p^*)$." Then $\psi_{point}(P_r, p^*) = \{(p^* + p)/2\}$, where p is any of the points in P_r furthest from p^*.

Theorem 1 *Function ψ_{point} correctly solves problem POINT.*

Outline of Proof For any real time t, let $S(t)$ be the set of positions of the robots at t. Then it suffices to show that $CH(S(t))$ converges to a single point as t goes to infinity, where CH denotes the convex hull. Suppose this is not the case. Since $CH(S(t)) \supseteq CH(S(t'))$ for any real times t and t' such that $t < t'$, $CH(S(t))$ must then converge to a convex polygon C. This implies that for any small real number $\delta > 0$, there exists some real time t_0 such that for any $t > t_0$ and any corner v of C, there is at least one robot in the δ-neighborhood of v (denoted δ_v). If δ is chosen sufficiently small, then any such inactive robot must eventually (become active and) leave δ_v, since the robots that are furthest from it are located near the δ-neighborhood of some other corners of C, that are more than 3δ away from v. So, for $CH(S(t))$ to converge to C, some of the robots, say r, must enter δ_v repeatedly. However, since δ is chosen sufficiently small, this is possible only if there already is a robot in δ_v when r chooses a robot furthest from itself. Clearly this is impossible. □

Let us modify ψ_{point}, and use point p (the point in P_r furthest from p^*) instead of $(p^* + p)/2$ as the next position of r. That is, robot r moves to the position of a robot furthest from itself. This function does not solve POINT, since of there are only two robots and their clocks are synchronized, then they simply continue to exchange their positions. In fact, using an argument similar to that in the proof of Theorem 1, we can prove the following theorem. Let $\psi_{point1}(P_r, p^*) = \{bp^* + (1-b)p\}$, where b is a constant real number.

Theorem 2 [2] *Function ψ_{point1} correctly solves problem POINT iff $0 < b < 1$.*

Another possible algorithm for POINT is function $\psi_{point2}(P_r, p^*) = \{g\}$, where g is the gravity center of p^* and the points in P_r. (We can prove its correctness using an argument similar to that in the proof of Theorem 1.) Note that if the local clocks are synchronized, then it can solve POINT in one step, since all robots simultaneously move to g. However, if the clocks are not synchronized, it is not clear whether or not ψ_{point2} can solve POINT more quickly than ψ_{point} in general.

Function $\psi_{point3}(P_r, p^*) = \{(p^* + q)/2\}$, where q ($\neq p^*$) is a point in P_r closest to p^*, moves robot r to the midpoint of r and a robot nearest to r. This function may seem to be a better algorithm than ψ_{point}, since finding a nearest robot can be easier than finding a furthest one. Unfortunately, ψ_{point3} does not solve POINT. To see this, consider the case of four robots

[2] The theorem is partially suggested by Saito[4].

r_1, r_2, r_3 and r_4 in which initially, r_1 and r_2 (or r_3 and r_4) are close to each other but r_1 and r_3 are far apart. Then, r_1 and r_2 (or r_3 and r_4) converge to a point, but since this process never terminates in finite steps, the four robots never converge to a point.

The next theorem states that any problem solvable by a deterministic algorithm is actually solvable by an algorithm for POINT, such as ψ_{point}.

Theorem 3 *Let π be a predicate which is invariant under any motion and similarity. Then there is a deterministic algorithm for solving the convergence problem for π if and only if for any P, $\pi_{point}(P)$ implies $\pi(P)$.*

The proof of Theorem 3 uses the fact that "clones" remain indistinguishable forever, where two robots r and s are *clones* of each other if they have the same coordinate system (and thus by assumption, the same initial position), and their local clocks are synchronized. In the following, we assume that no clones exist.

4 Agreement on a Coordinate System

In this section we investigate the problem of obtaining a common coordinate system, where the goal is to let the robots agree on the origin, unit distance, and direction. If the robots have a common coordinate system, then a special leader robot can be elected (if necessary) that controls all other robots. It turns out, however, that agreeing on a coordinate system is not possible in general. Specifically, we show that, while it is possible for the robots to agree on the origin and unit distance, agreeing on direction is not possible in general. We also discuss a limitation of oblivious algorithms.

4.1 Agreement on the Origin

We call the problem of agreeing on the origin ORIGIN. We first consider the formation problem for *two* robots, called MEET, for predicate π_{point} introduced in Section 3. The goal of MEET is thus to move two robots to a single point *in finite steps*. Recall that function ψ_{point} of Section 3 solves the corresponding convergence problem for any number of robots. We have:

Theorem 4 *There is no oblivious algorithm to solve problem MEET.*

Outline of Proof Consider robots r and s. Suppose that there is an oblivious algorithm ψ that solves MEET. We first observe that, regardless of Z_r and Z_s, there must be positions p and q (given in terms of Z) of robots r and s, respectively, such that, from that configuration, ψ moves exactly one of r and s to the position of the other. (If this is not true, then by changing the rate of their local clocks carefully and using the fact that ψ is oblivious, we can obtain an infinite sequence of moves that never brings the robots to a single point.) Now, consider an initial configuration in which, in terms of Z, r is at $(0,0)$, s is at $(1,1)$, $Z_r = Z$, and Z_s is obtained from Z_r by translating its origin to $(1,1)$ and then rotating it about $(1,1)$ for 180 degrees. Then the situation looks identical to both r and s. Thus even though ψ may not be deterministic, all future configurations can look identical to r and s, if their clocks are synchronized and they always move in the same (symmetric) manner. So it is possible that they never reach a configuration in which ψ moves exactly one of r and s to the position of the other. This is a contradiction. □

On the other hand, we can show that there is a non-oblivious algorithm, called ψ_{meet}, for solving MEET. Instead of describing ψ_{meet} formally, we explain the behavior of a robot that executes it.

Algorithm ψ_{meet}: When robot r becomes active for the first time, it tests whether the other robot s is at the same position as itself. If so, then r does not do anything. Otherwise, r rotates

its coordinate system Z_r about the origin so that s is on the positive y-axis of Z_r, say at $(0, a)$ (By assumption, r is at $(0,0)$ of Z_r.) Then it moves in the positive x direction of Z_r, over any nonzero distance. It then continues to move in the same direction whenever it becomes active, until it observes that the position of s has changed twice.

Now, r knows line ℓ that contains the trajectory of s. (Note that by symmetry, ℓ is the x-axis of s's coordinate system Z_s.) If ℓ is parallel to the x-axis of Z_r, then r moves to $(0, a/2)$. Otherwise, r moves to the intersection of ℓ and the x-axis of Z_r. □

Theorem 5 *Function ψ_{meet} correctly solves problem MEET.*

Outline of Proof The key observation is the following: When one of the robots, say r, observes that the position of s has changed twice, s must have already observed that r's position has changed at least once, and thus s knows where the x-axis of Z_r is. Similarly, s will know that r knows where the x-axis of Z_s is. Now, it is easy to show that the trajectory of r (i.e., the x-axis of Z_r) and the trajectory of s (i.e., the x-axis of Z_s) are parallel iff r and s become active for the first time simultaneously. Also, if they become active for the first time simultaneously, then each robot has seen the other robot when it was at its initial position. So if the two trajectories are parallel, then they move to the midpoint of their initial positions. (They know where that point is.) Otherwise they move to the intersection of the two axes. □

Note that in ψ_{meet}, both robots know the position where they meet before reaching there (This is not true for the algorithms in the previous section.) Thus ψ_{meet} solves ORIGIN for two robots, except in the case when the robots have the same initial position. In this case, neither of the robots ever moves, and thus they will never know whether they have reached an agreement. We cope with this problem by using additional instructions before ψ_{meet}: Each time r becomes active and finds that the other robot s is at the same position as itself, it moves over distance 1 in the positive x-direction of Z_r. (Robot s moves in a similar manner, using its local clock c_s and coordinate system Z_s.) Then, using the assumption that clones do not exist, we can show that eventually, r and s will occupy different positions. Then each of them starts executing ψ_{meet} when it becomes active again, and eventually they both move to the same position (and agree on the origin). So ORIGIN is solvable for $n = 2$.

Now, we describe an algorithm ψ_{origin} for ORIGIN for the case $n > 2$. Before we apply ψ_{origin}, we use another algorithm $\psi_{scatter}$ to transform the given initial distribution of the robots into one in which (1) no two robots occupy the same position, and (2) the robots are not located on a single line segment. Algorithm $\psi_{scatter}$ consists of two parts. Part 1 moves all the robots to distinct positions, and is similar to, but more complex than, the scattering process we used for ψ_{meet}. (We can prove its correctness using the assumption that no clones exist.) Part 2 assures that the robots do not form a single line segment.

Algorithm $\psi_{scatter}$:

Part 1: Suppose that robot r becomes active and finds that not all robots occupy distinct positions. (Otherwise, Part 1 is over.) Let P_r be the positions of the robots that r observes, given in terms of Z_r. (1) If no other robot is located at the current position of r, then r does not move, until it observes that all robots occupy distinct positions. (2) On the other hand, if $m \geq 1$ other robots are located at the current position of r, then r finds the distance $a_r > 0$ to its nearest neighbor (excluding those at the current position of r), and computes the value $b_r = a_r f(a_r)/2$, where for $x \geq 0$, $f(x) = 1 - 1/2^x$ is a monotonically increasing function in the range $[0, 1)$. Then r moves over distance $b_r/2$ in the positive x-direction. After that, each time r becomes active and finds that there are still m other robots at the same position as itself, r moves over distance $b_r/2^k$ in the same direction, if this is the k-th move ($k = 2, 3, \ldots$). When r eventually becomes active and finds that there are fewer than m other robots at the same position as itself, r repeats this entire procedure from the beginning of (2), using a new value of a_r (and resetting k). This process is repeated each time r finds that fewer robots are located

at the same position as itself, until no other robot is found to occupy the same position as r. Then r waits, without moving, until all robots occupy distinct positions.

Part 2: (At this moment, all the robots occupy distinct positions.) Suppose that robot r becomes active and finds that the robots are located on a single line segment. (Otherwise Part 2 is over.) If r is located at an endpoint of the segment, then it does not move. If r is not at an endpoint, then r moves over any distance in the direction perpendicular to the segment. (As soon as some robot does this, the robots no longer form a single line segment, and of course, all the robots occupy distinct positions.) □

One technical difficulty is that in general, a robot cannot determine, given the positions of the robots observed at two time instants t_1 and t_2, which robot has moved to which position between t_1 and t_2. We overcome this problem by imposing a bound on the maximum distance that any robot can move while other robots are inactive, so that the robot at position p at time t_1 must be at the position closest to p at time t_2. Specifically, when robot r becomes active for the first time, it *memorizes the distance* $a_r > 0$ *to its* nearest neighbor. Then r moves at most distance $a_r\epsilon/2^k$ in the k-th move, where $0 < \epsilon \leq 1/2$ is a constant chosen by the algorithm. This restriction assures that r remains in the interior of the $a_r\epsilon$-neighborhood of its initial position. Since the interiors of such neighborhoods of two robots located at different positions do not intersect, any robot can correctly know the position of r, even after it remains inactive for a long time. In Part 2 of $\psi_{scatter}$, we used a similar technique to prevent two robots located at different positions from moving to the same position.

Algorithm ψ_{origin} is given next. Each robot first executes algorithm $\psi_{scatter}$, and then starts executing ψ_{origin} as soon as it finds that (1) no two robots occupy the same position, and (2) the robots are not located on a single line segment. So in the following description of ψ_{origin}, we assume that these two conditions are already satisfied.

Algorithm ψ_{origin}: Let P_r be the positions of the robots that r observes when it becomes active for the first time (after finishing $\psi_{scatter}$). Robot r finds the distance a_r to its nearest neighbor. If r is not at a corner of the convex hull $CH(P_r)$ of the points in P_r, then r memorizes the current position p of its nearest neighbor, and moves towards p each time it becomes active, staying in the interior of the $(a_r/2)$-neighborhood of its initial position. (See the explanation in the preceding paragraph.) Suppose that r is at a corner of $CH(P_r)$, and let a, b, c and d be consecutive corners in clockwise order, where r is at b. Then r memorizes the direction that is away from a along the line containing $\overline{ab}$, and moves in that direction each time it becomes active, staying in the interior of the $a_r\epsilon$-neighborhood of b for some $\epsilon \leq 1/2$. Here, ϵ is chosen so that the $a_r\epsilon$-neighborhood does not intersect the line containing $\overline{cd}$. (This assures that the robot s at corner c remains to be at a corner of the convex hulls of the new positions of the robots even after r and s move.)

Robot r continues to move as described above, until it observes that the position of each robot has changed at least twice. Then, r knows line ℓ_s containing the trajectory of s for each robot s at a corner of the initial convex hull. Since the convex hull of the initial positions of the robots is a k-sided polygon for some $k \geq 3$, the lines $\{\ell_s\}$ determine a unique, smallest convex polygon Q that contains the initial convex hull. So r chooses the center of gravity of the corners of Q as the (common) origin. □

We can show that using ψ_{origin}, every robot eventually knows the lines $\{\ell_s\}$ correctly. (The trajectories of the robots that are not at a corner of the convex hull are not used to obtain the common origin. We must still move such robots, however, in order for other robots r to know that they have become active sufficiently many times and observed r's movement.) So Algorithm ψ_{origin} solves ORIGIN for $n > 2$.

The following theorem summarizes the discussion given above.

Theorem 6 *ORIGIN is solvable for any* $n \geq 2$.

4.2 Agreement on the Unit Distance

We call the problem of agreeing on the unit distance UNITDIST. For the case $n = 2$, we can show that in ψ_{meet}, each of robots r and s eventually finds the initial position of the other. Then they can use the distance between their initial positions as the unit distance. For the case $n > 2$, the robots can choose, as the unit distance, the length of a shortest side of the k-sided convex polygon Q that they obtained using algorithm ψ_{origin} in Subsection 4.1. So we have:

Theorem 7 *UNITDIST is solvable for any $n > 2$.*

4.3 Agreement on Direction

Now we show that the third problem of agreeing on direction is unsolvable in general. Let us call this problem DIRECTION.

Theorem 8 *There is no algorithm for solving DIRECTION, even if $n = 2$ and the local clocks are synchronized.*

Outline of Proof Consider two robots r and s such that (1) Z_s is obtained from Z_r by rotating it for 180 degrees about the origin, and (2) the local clocks of r and s are synchronized. Then it is possible that r and s move in the same (symmetric) manner all the time, and thus when r decides the direction of its positive x axis, s chooses the opposite direction for its positive x axis. □

Note that as we stated in Section 2, the other two agreement problems are trivially solvable if the local clocks are synchronized.

5 Robots with a Sense of Direction

We say that the robots have a sense of direction if they agree on the direction of the positive x axis. The main result of this section is the following:

Theorem 9 *If the robots have a sense of direction, then they can agree on a common coordinate system, and elect a leader.*

Although the first claim follows immediately from the discussions in Subsections 4.1 and 4.2, in the full paper we show that the following algorithm, called $\psi_{coordinate}$, solves *both* the coordinate agreement problem and the leader election problem, if the robots have a sense of direction. As before, it is assumed that we first use $\psi_{scatter}$ so that no two robots will occupy the same location.

Algorithm $\psi_{coordinate}$: We only give an outline. (We use "east" to denote the positive x direction, etc.) The idea is to choose the horizontal line though the northern most robots as the common x axis, and the vertical line though the eastern most robots as the common y axis. To achieve this goal, (1) each eastern-most robot that is not northern most continue to move north within a "small" neighborhood of its initial position (as explained in Section 4), and (2) all other robots continue to move west. (If there is a robot that is both eastern-most and northern most, then it first moves to a point to the south of any robot that is eastern-most, and then does (1).) Then eventually, every robot knows the trajectories of all other robots, and chooses the northern-most (or eastern most) trajectory as the common x- (or y-) axis. The unit distance is then chosen to be the minimum distance > 0 between any two westbound (i.e., horizontal) trajectories. The leader is the northern most robot among the eastern most robots. □

Since the leader can compute the final positions of all the robots that satisfy any given predicate and "guide" them to their respective final positions, we obtain the following theorem. Details will be found in the full paper.

Theorem 10 *If the robots have a sense of direction, then for any predicate π, the convergence and formation problems for π is solvable.*

In particular, we have

Corollary 1 *If the robots have a sense of direction, then the problems of forming a circle and a line segment are solvable.*

6 Concluding Remarks

We viewed a group of mobile robots as a distributed system in which the components can communicate with each other only by means of their moves, and investigated the possibility and impossibility of solving some of the problems related to the formation of geometric figures in the plane. Our study indicates that the assumptions we make on the knowledge and capabilities of the robots can affect the difficulty of solving the given problem in a subtle way. We are currently conducting similar investigations on (1) randomized algorithms, (2) the case in which the motion of a robot is not instantaneous, and (3) the 3-dimensional case. The results will be reported in a future paper.

References

[1] K. Fujimura, "Model of reactive planning for multiple mobile agents," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, June 1991, pp. 1503–1509.

[2] T. Fukuda and S. Nakagawa, "Approach to the dynamically reconfigurable robot systems," *Journal of Intelligent and Robotics Systems 1*, 1988, pp. 55–72.

[3] S. Hirota, *private communication*, 1992.

[4] Y. Saito, *private communication*, 1992.

[5] K. Sugihara and I. Suzuki, "Distributed motion coordination of multiple mobile robots," *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, Pennsylvania, September 1990, pp. 138–143.

[6] O. Tanaka, *private communication*, 1992.

[7] J. Wang and G. Beni, "Cellular robotic systems: Self organizing robots and kinetic pattern generation," *Proc. of the 1988 IEEE International Workshop on Intelligent Robots and Systems*, Tokyo, Japan, 1988, pp. 139–144.

[8] Yamashita, M., and Kameda, T., "Computing on anonymous networks," *Proc. 7th ACM Symposium on Principles of Distributed Computing*, 1988, 117–130.