数理解析研究所講究録
第 833 巻 1993 年 142-152

Cooperative Control Algorithms for Anonymous Mobile Robots

鈴木一郎 (Ichiro Suzuki)
Department of Electrical Engineering and Computer Science
University of Wisconsin - Milwaukee
P.O. Box 784, Milwaukee, WI 53201, U.S.A.
suzuki@cvax.cs.uwm.edu

山下雅史 (Masafumi Yamashita)
広島大学 工学部
724 東広島市 鏡山町
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 they 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 method, each robot executes a simple algorithm and determines its movement adaptively based on the observed movement of other robots, so that the robots as a whole group will achieve the given goal. Each robot is a mobile processor with an eye sensor which, at every tick of its local clock, observes the positions of all the robots and moves to a new position. 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, the interest in the distributed control method for multiple mobile robots has increased considerably among the researchers in the U.S. and Japan[1, 2, 3, 4, 6, 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 [3] [6].

A robot system in which the robots can communicate with each other by radio, for example, 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 in the field of distributed computing (although such analyses are by no means easy). In this paper, we consider a system consisting of multiple 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. 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. (The new position need not be distinct from the current position.) 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, and thus the clock of a robot may tick much more frequently than that of some other robots. It is assumed that initially, the robots do not have a common sense of direction and unit distance, and hence the local x-y coordinate system of a robot may not be the same as that of another robot. Thus, the robots do not have a common x-y coordinate system. 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 from 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. 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 that works correctly even if robots are added or deleted from the system a finite number of times. (For simplicity, in this paper we assume that a robot can move to its new position instantaneously.) We 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 letting 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 (instead of converging) two robots to a single point, and show that this problem can be solved by a nonoblivious algorithm but not by any oblivious algorithm. 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 occupy the same position initially. On the other hand, we show that agreement on the direction is not solvable even if $n = 2$. 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 that the robots can elect a leader provided that no two robots occupy the same position initially. Once a unique leader is elected, the robots can be moved to form any geometric figure.

As a final note of this section, we remark that if the local clocks are synchronized, then the robots can easily communicate with each other by means of the distances of their moves. 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].

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 consider robots in 2–dimensional Euclidean space E^2. (Many of the results given in the paper also hold for robots in 3–dimensional Euclidean space.) We fix an x–y coordinate system Z and specify the position of a robot as a pair (x, y) of real numbers using Z. It is assumed, however, that Z is used only for specifying the positions of robots and the robots have no information on Z.

A robot r is modeled as a mobile processor with sufficiently large memory space, an artificial eye sensor, and a local clock c_r. It can see and locate other robots in its visibility range with respect to its local x–y coordinate system Z_r, and also move to any position specified as coordinates of Z_r. The coordinate system Z_r is local to r, in the sense that two robots may have different local coordinate systems. We denote by $c_r(t)$ the clock reading of c_r at real time t. (The robots have no access to real time.) We assume that for any real times t and t', $c_r(t) < c_r(t')$ if $t < t'$, and that for any non-negative integer k, there exists a real time t such that $c_r(t) = k$. The clocks c_r are said to be synchronized if there is a constant integer d such that for any r and t, $t - c_r(t) = d$ holds. Otherwise, they are said to be asynchronous, and many different degrees of asynchrony can be defined. If the local clocks are synchronized and the number of robots never changes, then for simplicity, we may assume that when robot r is initialized, the clock reading of c_r is 0.

Let $\mathcal{I}$ be the set of all possible configurations I (i.e., contents) of the memory of a robot r. Let R be the set of real numbers. An algorithm for r is a function ψ from $\mathcal{I}$ to the set of subsets of R^2, such that $\psi(I)$ is the set of possible next positions of r (given in terms of Z_r) when the memory configuration of r is I. If $\psi(I)$ is a singleton for each I, then ψ is said to be deterministic. Otherwise, r randomly selects from $\psi(I)$ its next position (and moves there), and hence ψ is a randomized algorithm. Note that the identifier r is not an argument of ψ.

Let P_r be the set of the positions of all robots, including r itself, given in terms of Z_r. That is, P_r shows the positions of the robots from the viewpoint of robot r. We naturally assume that r knows its current position with respect to Z_r. Note that r may be able to obtain only a partial information on P_r, since r cannot see those robots that are outside its visibility range. (But we assume that all robots have the same eye sensor.) So we let $viz(P_r) \subseteq P_r$ denote the set of the positions of all robots that are visible from r. For example, if r can see only those objects that are within distance δ, then $viz(P_r)$ contains the positions of those robots that are within distance δ from r. To simplify the discussion, in this paper, we assume that r has unlimited visibility, and hence $viz(P_r) = P_r$. We assume that r observes the positions of other robots every time its local clock reaches an integer value, and that only those positions that have been observed can be used as data for determining the possible next positions of r. This means that the robots never observe, for example, the velocity of other robots.

From the preceding discussion, the behavior of robot r is given as follows. Here, I_0 is a fixed (common) initial configuration of the memory of a robot.

[Behavior of robot r]

```
$I := I_0$;
Wait until the clock reading of $c_r$ becomes an integer $s$;
**repeat**
  /* Start Move Action*/
```

```
    Let $P_r$ be the positions of the robots at local time $s$;
    $I := I \cup \{(p,s) | p \in viz(P_r)\}$;
    Randomly select an element $q$ from $\psi(I)$;
    Move to $q$;
    $s := s + 1$;
    /* End Move Action*/
    Wait until $c_r$ reaches the value $s$;
  until forever.
```

The assignment $I := I \cup \{(p,s)|p \in viz(P_r)\}$ means that robot r stores information $viz(P_r)$ obtained at local time s with tag s, in order to distinguish it from the information obtained at different times. Note that the robots are anonymous in the following sense: (1) the initial configuration I_0 and function ψ are common to all robots, (2) the identifier r of robot r is not an argument of ψ, and (3) $viz(P_r)$ contains only the positions of robots (but not their identities).

For the sake of simplicity, we assume that a robot can move to any position instantaneously. In other words, we assume that the time necessary for executing a move action, a series of instructions between two comments "Start Move Action" and "End move Action", is negligibly small compared with one unit time of each local clock, and that whenever a robot observes, all robots have already reached their destinations and are stationary.

Because of the assumption for local clock c_r, the execution of move actions of robots is fair. Every robot r executes its move action infinitely many times. Finally, note that if synchronized clocks are available, by definition, all robots move simultaneously.

Let π be a predicate over the set of multisets of points which is invariant under any motion (i.e., rotation and parallel transformation) and similarity. In what follows, predicates over multisets we will discuss in this paper are assumed to satisfy the above condition. Let π be such a predicate. We are interested in designing algorithms (functions) ψ such that, for example, the robot obeying ψ converge to points (with respect to coordinate system Z) such that predicate π holds, no matter where the robots are initially distributed. Since robots never can perceive the absolute coordinate system Z, all we can expect for algorithms is to form a figure which is similar to the goal figure. The restriction for predicates was introduced because of this reason.

This framework permits us to add and/or remove robots dynamically. Although we will not state it explicitly, all algorithms we will discuss in this paper correctly work (after small modifications for some algorithms), even if the number of robots changes dynamically. Algorithms having this property can be seen as self-stabilizing algorithms, which solve problems in spite of any transient failures. Finally, throughout this paper, we reserve symbol n for denoting the number of robots.

3 Converging to a Point

The *convergence problem* for a predicate π is the problem of designing an algorithm such that the robots obeying the algorithm converge to points for which π holds, regardless of their initial distribution. Define a predicate π_{point} as $\pi_{point}(p_1, \ldots, p_n) = true$ iff $p_i = p_j$ for any $1 \leq i, j \leq n, 2 \leq n$. 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. The aim of this section is not only to propose algorithms for POINT, but also to discuss relations between additional assumptions we can make and possible algorithms. We also discuss a limitation of deterministic algorithms.

Let P_r be the current positions of robots with respect to Z_r. Although r may not occupy the origin of Z_r, it can be assumed that r is at the origin, without loss of generality.

First of all, let us propose a deterministic algorithm for solving POINT. Let P and p be a set of points and a farthest point in P from the origin, respectively. Define a function ψ_{point} as $\psi_{point}(P) = \{p/2\}$ for any P.[1] Since we assume that a robot r is at the origin of its coordinate system Z_r, r executing ψ_{point} always moves to the midpoint of a farthest robots and itself. Notice that the next position of such r is determined only from the current positions P_r of the robots (with respect to Z_r), regardless of the rest of the history I that r might have observed. (Notice also that P_r does not include time information.) Such an algorithm is said to be *oblivious*. All algorithms we will discuss in this section are oblivious deterministic.

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

Suppose we use point p instead of $p/2$ as the next position. Consider the case of two robots. If their clocks are synchronized, then the two robots simply exchange their positions and never converge to a point. In fact, using an argument similar to that in the proof of Theorem 1, we can prove the following theorem. Let $\psi_{point1}(P) = \{p/b\}$ for any P, where b is a constant real number.

Corollary 1 [2] *Function ψ_{point1} correctly solves problem POINT iff $b > 1$.*

As another possible candidate for a POINT algorithm, one may choose function $\psi_{point2}(P_r)$ $= q$, where q is a point which is uniquely determined from the positions of robots, independently of how they look from the view P_r of r, such as the center of gravity. It is easy to see that by using an argument similar to that in the proof of Theorem 1, one can show that this new algorithm solves problem POINT, in this particular case of the center of gravity. Note that if the local clocks are synchronized, then it can solve problem POINT very efficiently since all robots simultaneously move to the center of gravity in the first move. However, it is not clear whether or not ψ_{point2} can solve POINT more efficiently than ψ_{point} in general using more information on the robot positions.

Function $\psi_{point3}(P) = q/2$, where q is a nearest point in P from the origin (except the points at the origin), seems to be a better algorithm than ψ_{point}, since finding a nearest robot should be easier than finding a farthest one. Note that ψ_{point3} always moves a robot to the midpoint of the nearest robot (except those at the same point) and itself. Unfortunately, ψ_{point3} cannot solve POINT. To see this, consider the case of four robots r_1, r_2, r_3 and r_4 initially distributed in such a way that r_1 and r_2 (resp. r_3 and r_4) occupy close positions. Suppose that none of them move at the same time. Then, robots r_1 and r_2 (resp. r_3 and r_4) converge to a point, which is unachievable within a finite time. Hence, ψ_{point3} cannot solve 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}.

[1]Since an algorithm is a function from $\mathcal{I}$ to the set of subsets of R^2, strictly speaking, ψ_{point} must be defined as $\psi_{point}(I) = \{p/2\}$ for any I. Here, p is a farthest point in P from the origin, and $P = \{p| \ (p,s) \in I\}$, where s is the integer such that (q,t) does not exist in I for any q and $t(> s)$.

[2]The corollary is partially suggested by Saito[5].

Theorem 2 *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 2 uses the fact that "clones" remain indistinguishable forever, where two robots r and s are *clones* of each other if they share a coordinate system in common (i.e., $Z_r = Z_s$), their local clocks always show the same value (i.e., $c_r = c_s$) after they are initialized, and both their initial positions and the times at which they are initialized, respectively, are the same. In what follows, we assume that no clones exist.

4 Agreeing on a Coordinate System

In the previous section, we investigated oblivious algorithms for POINT. Although oblivious algorithms are useful (because they are easy to understand and to analyze), they have also apparent drawbacks caused by the fact (actually the definition) that the robots cannot communicate with each other, since some common concepts (e.g., how large the unit distance is, which direction north means, and so on) are inevitable for communication, but remembering the concepts requires memory. This section investigates the problem of obtaining a common coordinate system. Assuming that robots share a coordinate system in common, roughly speaking, robots can be named in terms of their coordinates, and a special leader robot can be elected (if it is necessary) who controls all other robots. The fact is, however, that agreeing on a coordinate system is not possible in general. The problem of agreeing on a coordinate system consists of the following three agreement problems; agreeing on the origin, agreeing on the direction, and agreeing on the unit distance. We, therefore, examine which ones of these are difficult. Through the investigation, we will learn what kind of knowledge is essential to have a common coordinate system. We also explain a limitation of oblivious algorithms.

4.1 Agreeing on the Origin

Recall that an algorithm ψ is said to solve POINT if the robots obeying ψ converge to a point from any initial distribution. This definition does not require that the process finishes in finite steps. In fact, algorithm ψ_{point}, for example, does not assure that the robots will move to a single point in finite steps. In this subsection we first consider the problem of truly moving two robots to a single point in finite steps. We call this problem RENDEZVOUS. We have the following negative result.

Theorem 3 *There is no oblivious algorithm to solve problem RENDEZVOUS.*

On the other hand, we can show that there is an algorithm $\psi_{rendezvous}$ which is not oblivious for solving RENDEZVOUS. Instead of describing $\psi_{rendezvous}$ using mathematical notations, we explain the behavior of a robot executing it.

Algorithm $\psi_{rendezvous}$: When a robot r wakes up for the first time, if it finds out that the two robots reside at the same point, then there is nothing to do. Otherwise, if they occupy different points, it transforms its coordinate systems Z_r in such a way that (1) its position (with respect to Z_r) becomes its origin $\mathbf{0}_r$ and (2) the position (with respect to Z_r) of the other robot s becomes $(0, a)$ for some real number $a > 0$. Then it moves one unit distance (with respect to Z_r) towards its right to $(1, 0)$.

It then continues to move one unit distance in the same direction on the X–axis, whenever it wakes up until it observes that the position of s changes for the second time.

If the trajectory of the first change of s's position which r observes is parallel with r's X–axis, then it moves to $(0, a/2)$; otherwise, it moves to the intersection of r's X–axis and the line which includes the trajectory of s. □

Theorem 4 *Function $\psi_{rendezvous}$ correctly solves problem RENDEZVOUS.*

Note that in $\psi_{rendezvous}$, both robots know the point at which they will meet before actually meeting each other there. (This is not true for the algorithms in the previous section.) Thus $\psi_{rendezvous}$ *almost* solves the problem of agreeing on the origin, which we call ORIGIN, for two robots. Here, we say "almost," since if initially the two robots are at the same point p, then neither of them ever moves, and hence neither of them will ever be convinced that the other robot has already woken up and recognized this situation. Therefore, the robots will never be convinced that they have agreed on p. To remove this small flaw, we may modify the instruction for this case as follows: If a robot finds that it occupies the same point as the other robot when it wakes up at local time t for the first time, then it moves to $(1, t)$ with respect to its coordinate system, and restarts the algorithm $\psi_{rendezvous}$. Since we assume that clones do not exist, it is easy to see that the two robots never move to the same point infinitely many times. So we obtain:

Theorem 5 *Problem ORIGIN is solvable, provided $n = 2$.* □

Now, we propose an algorithm ψ_{origin} for ORIGIN for the case $n > 2$. Let P be the set of the initial positions of the robots. The algorithm assumes that the robots are located at distinct points. So we first transform any given initial distribution (in which two or more robots may occupy the same point) into one in which this condition is satisfied. This can be done by an algorithm $\psi_{scatter}$ given below. Although we execute $\psi_{scatter}$ before ψ_{origin} as preprocessing, confusion between the two algorithms never occurs. Algorithm $\psi_{scatter}$ is an extension of the scattering process of $\psi_{rendezvous}$ we explained immediately before Theorem 5.

Algorithm $\psi_{scatter}$: Let P_r be the current positions of robots r observes. Hence, r always resides at its origin. When r wakes up, if no other robots are located at the origin, then it stays there until all robots are located at distinct points. If there is at least one other robot located at the origin, then it moves to $(a/4, a/4t)$ (with respect to Z_r) and repeats this procedure from the beginning. Here $a > 0$ is the minimum distance between two positions in P_r, and t is the current local time. □

It is easy to check that in finite time, each robot occupies a distinct point, since no clones exist.

Another 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 point between t_1 and t_2. We overcome this problem by imposing a bound on the maximum distance that any robot can move while another robot sleeps, so that the robot at point p at time t_1 must be at the point closest to p at time t_2. Specifically, let $a_r > 0$

be the minimum distance between two robots, when robot r wakes up for the first time. Then we allow r to move at most distance $a_r\epsilon/2^k$ in the k–th move. Here, $\epsilon > 0$ is a (small) constant chosen by the algorithm. This restriction assures the above property, since r remains inside the $a_r\epsilon$–neighborhood of its initial position. Then any robot can correctly recognize which robot has moved to which position even if it sleeps for a long time period. Of course, this technique cannot be used in an algorithm in which a robot needs to move over a long distance.

Algorithm ψ_{origin}: Let P be the initial distribution of robots, and let $CH(P)$ be the convex hull of P. It is assumed that the parameter ϵ mentioned above is chosen in such a way that the farthest neighbor relation among vertex points of $CH(P)$ does not change by their moves. When a robot r wakes up for the first time, it decides a_r. Next, it moves towards a farthest robot s from r, if r is located at a vertex point of $CH(P)$; otherwise, it moves along the line perpendicular to the line connecting r and s. Let d_r be the direction r moves.

It then continues to move keeping the direction d_r unchanged, until it observes that the position of each robot changes at least twice.

Then, r knows the trajectory ℓ_s of s's first move for each robot s. We identify the trajectory with the line including it, and let $L = \{\ell_s\}$. Further let E be the envelop polygon of L.[3] Now, it chooses the center of gravity of the vertex points which define E as the (common) origin. □

Theorem 6 *Algorithm ψ_{origin} solves ORIGIN, provided $n > 2$.*

Hence, we can conclude that ORIGIN is solvable.

Corollary 2 *Problem ORIGIN is solvable.* □

4.2 Agreeing on the Unit Distance

We call the problem of agreeing on the unit distance UNITDIST. First, using an algorithm similar to $\psi_{rendezvous}$, we can show the next theorem.

Theorem 7 *UNITDIST is solvable, provided $n = 2$.*

For the case $n > 2$, consider algorithm $\psi_{unitdist}$ given below. Again, it is assumed that we use $\psi_{scatter}$ before executing $\psi_{unitdist}$ so that no two robots are located at the same point in P.

Algorithm $\psi_{unitdist}$: Let P be the initial distribution of robots, and let $CH(P)$ be the convex hull of P. It is assumed that the parameter ϵ is chosen in such a way that the set of vertex points of the convex hull of the current positions of robots does not change by their move. When a robot r wakes up for the first time, it decides a_r. Then it moves towards a farthest robot s from r, if r is not a robot located at a vertex point of $CH(P)$. If r is a robot located at a vertex point v of $CH(P)$, then r moves along the line perpendicular to the bisector of the angle at v. Let d_r be the direction r moves.

[3]The envelope polygon of a set L of lines is the polygon consisting of the finite length segments that bound the infinite faces of the arrangement defined by L.

It then continues to move keeping the direction d_r unchanged, until it observes that the position of each robot changes at least twice.

Then, r knows the trajectory ℓ_s of s's first move for each robot s. We identify the trajectory with the line including it, and let $L = \{\ell_s\}$. Let E be the envelop polygon of L. Now, it chooses the shortest edge in E as the (common) unit distance. □

Theorem 8 *Algorithm $\psi_{unitdist}$ solves UNITDIST, provided $n > 2$.*

Hence, we can conclude that UNITDIST is solvable.

Corollary 3 *Problem UNITDIST is solvable.* □

4.3 Agreeing on Direction

This subsection shows that the third agreement problem of agreeing on direction is unsolvable in general. Let us call this problem DIRECTION.

Theorem 9 *There is no algorithm for solving DIRECTION, even in the case of $n = 2$.*

It is worth emphasizing that DIRECTION is unsolvable even if the local clocks are synchronized. On the other hand, the other two agreement problems are trivially solvable if the local clocks are synchronized.

5 Robots with a Sense of Direction

In this section, first we show that if the robots have a common sense of direction, then they can agree on a common coordinate system. As before, we assume that no two robots occupy the same position in the initial distribution. It is easy to verify that the following algorithm achieves this goal.

Algorithm $\psi_{coordinate}$: When a robot r wakes up, it moves one unit distance (with respect to Z_r) to the east, and repeatedly does so until it observes that the position of each robot has changed at least twice. Then r knows the trajectories ℓ_s of s's first move for each robot s, and since ℓ_s's are parallel to each other, r chooses the minimum distance > 0 between any two trajectories as the common unit distance, and the northern–most line as the common X–axis.

Next, r moves one unit distance to the north, and repeatedly does so, until it again observes that the position of each robot has changed at least twice. Then, r knows the trajectories h_s of s's first northbound move for each robot s, and it chooses the eastern–most line as the common Y–axis. □

Once the robots agree on a common coordinate system, they can elect a leader using any suitable lexicographic ordering of their initial positions. (Recall that we may assume that no two robots have the same initial position.) Here, a robot whose initial position is (a, b) can communicate the values a and b to other robots by moving along line $y = b$ first and then along line $x = a$. This idea is similar to that in $\psi_{coordinate}$.

Let π be any predicate. The *formation problem* for π is the problem of designing an algorithm such that the robots obeying the algorithm will eventually reach some points for which π holds in finite steps. For example, RENDEZVOUS is the formation problem for π_{point} for $n = 2$. By definition, if the formation problem for a predicate is solvable, so is the convergence problem for the same predicate. Since the leader can compute the final positions of all the robots that satisfy any given predicate and then communicate them to the other robots using its movement by a technique similar to the one used in $\psi_{coordinate}$, we obtain the following theorem.

Theorem 10 *For any predicate π, the formation problem for π is solvable, if robots have a sense of direction.*

Corollary 4 *The problems of forming a circle and a line segment are both solvable, if robots have a sense of direction.*

6 Concluding Remarks

We regarded a robot system as a distributed system in which components can communicate with 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 a similar investigation on randomized algorithms and for the 3-dimensional case. The results will be reported in a future paper.

References

[1] R. Brooks, "Micro-brains for micro-brawn: Autonomous microrobots," *Proc. IEEE Micro Robotics and Teleoperators Workshop*, Hyannis, MA, November 1987.

[2] A. Flynn, "Gnat robots (and how they will change robotics)," *Proc. IEEE Micro Robotics and Teleoperators Workshop*, Hyannis, MA, November 1987.

[3] 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.

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

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

[6] 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.

[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.