Computing and Informatics, Vol. 30, 2011, 137–153

A POINT SET CONNECTION PROBLEM FOR AUTONOMOUS MOBILE ROBOTS IN A GRID*

Adrian Kosowski

Department of Algorithms and System Modeling
Gdańsk University of Technology
80-952 Gdańsk, Poland
e-mail: kosowski@kaims.pl

Ichiro Suzuki

Department of Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee
WI 53201-0784 Milwaukee, USA
e-mail: suzuki@uwm.edu

Paweł yliski

Institute of Informatics
University of Gdańsk
80-952 Gdańsk, Poland
e-mail: zylinski@inf.ug.edu.pl

Manuscript received 8 September 2008; revised 22 October 2010
Communicated by Boleslaw Szymanski

* A preliminary version of this paper appeared in the Proceedings of the International Multiconference on Computer Science and Information Technology (Mrągowo, Poland, 2009), Volume 4, pp. 583–588. I. Suzuki was supported in part by UWM Research Growth Initiative. Work of P. Żyliński was partially done while he was visiting the Lund University under the postdoctoral Visby Programme Scholarship 01224/2007.

Abstract. Consider an orthogonal grid of streets and avenues in a Manhattan-like city populated by stationary sensor modules at some intersections and mobile robots that can serve as relays of information that the modules exchange, where both module-module and module-robot communication is limited to a straight line of sight within the grid. The robots are oblivious and move asynchronously. We present a distributed algorithm that, given the sensor locations as input, moves the robots to suitable locations in the grid so that a connected network of all modules is established. The number of robots that the algorithm uses is worst case optimal.

Keywords: Asynchronous algorithm, autonomous mobile robot, distributed algorithm, connected network, oblivious algorithm, grid

Mathematics Subject Classification 2000: 68W15, 68M14, 68R01

1 INTRODUCTION

Let $\mathbb{Z}$ and $\mathbb{R}$ be the set of integers and the set of reals, respectively. Let G be an infinite grid in the 2D Euclidean space $\mathbb{R}^2$ defined as the union of integer-coordinate points (x, y), $x, y \in \mathbb{Z}$, called *vertices* and unit-length line segments called *edges* connecting "adjacent" vertices located at distance 1 of each other. We view G as an environment of rows and columns in which communication is possible between two points p and q if and only if the line segment $\overline{pq}$ connecting them lies entirely within G. In other words, p and q can communicate with each other if and only if they can "see" each other assuming straight line visibility along rows and columns. A grid-like environment is a natural model for considering limitations on both vision and movement, when discussing motion planning problems in urban spaces.

Given a finite subset P of vertices of G, define its *visibility graph* G_P using P as the vertex set and including edge (p, q) for every pair of vertices $p, q \in P$ that are mutually visible. We refer to each connected components of G_P simply as a *component* of P. P is said to be *connected* if it has exactly one component. Assuming that

1. P represents stationary sensor modules in G that from time to time must communicate with each other, and
2. G contains a number of mobile robots, each represented by a point, that can serve as "relays" for inter-module communication, we discuss the following *connection problem*: Given a finite set P of vertices of G and initial locations of the robots, move the robots so that $R \cup P$ is connected, where R is the set of final locations of the robots.

We present a simple distributed algorithm, to be executed by the robots individually, for solving the connection problem in the CORDA model [22]. The CORDA model uses continuous time $t \geq 0$, and the robots asynchronously and repeatedly

execute an Idle-Look-Idle-Compute-Idle-Move cycle, where the Look and Compute steps are instantaneous while the Idle steps take a finite but unpredictable length of time. In the Move step the robot moves continuously at an unpredictable speed toward the target position computed in the Compute step based on the observation of the environment obtained in the Look step. Usually it is assumed that a robot stops and ends the Move step when it hits an upper bound on the distance it can move in one Move. In this paper we assume that the upper bound is 1 (so a robot can move from one vertex to an adjacent vertex). Note that a robot may be "seen while moving", and a robot may compute its target position based on an observation that may be obsolete because of the Idle step between the Look and Compute steps. We assume that in the Look step a robot obtains a complete description of the current configuration – the locations of the sensors and robots, as well as its own location, all in terms of its local coordinate system (the local coordinate systems of two robots may not agree). Here, we may conceive each robot as being equipped with a radar having an unlimited range for locating objects in G. Finally, we assume that the robots are *oblivious*. An oblivious robot does not have memory to store the events in the past, and hence the target location it computes in the Compute step is a function of what it observes in the Look step immediately preceding it.

We assume that initially the robots occupy distinct vertices, and impose the condition that at any time two or more robots must not create a *multiplicity* by occupying the same location simultaneously. This is based on the observation that, since the robots are oblivious, two robots (whose local coordinate systems agree) may never be separated once they occupy the same location, effectively reducing the number of available robots.

The problem of establishing or maintaining a connected network of a given set of entities has arisen in many areas, and hence there are a number of motivations for considering the connection problem in the setting described above – we shall discuss only three. First, our problem addresses data aggregation, which is a fundamental issue in sensor networks, where data collected by spatially distributed sensor modules are sent to a designated sink by a multi-hop routing algorithm – see [3, 9, 25] for recent surveys of strategies and techniques for the node placement problem in wireless sensor networks. Specifically, our connection problem can be considered as a variant of the dynamic node placement problem, where the network is adaptive and the objective is to maintain the connectivity between sensors via additional relays in a changing environment – see for example [1, 2, 13]. Although we discuss our problem in a static setting, our solution, which involves oblivious robots acting as relays, can be used to handle a dynamic situation in which the set of sensors to be connected may change from time to time, provided that the number of robots is sufficient.

Second, since we assume vision-based communication, our objective can be viewed as providing connectivity between the connected components of the visibility graph of a set of guards, as in [20, 21], where the problem is discussed in the context of computing control points of a navigational path in the presence of obstacles. In our scenario, sensors may be thought of as guards (partially) covering

the streets/avenues of a Manhattan-like city, with blocks of buildings obscuring visibility, where the robots must serve as additional connectors to make the visibility graph of the set of guards connected.

Finally, the connecting problem we discuss can be viewed as a variation of the *formation problem* of geometric patterns for autonomous mobile robots [10, 23, 24] in which the target pattern, usually fixed and given in advance, depends on the sensor module locations given as input to the robots. Problems related to formation include "rendezvous" [4, 7, 11, 17, 18], "spreading" [8] and "partitioning" [12]. See [19] for a survey of some of the results on the subject.

The following theorem summarizes the main result. As we discuss in Section 2, the instances of the connection problem are categorized into three cases, Cases 1, 2 and 3.

Theorem 1. There exists an oblivious algorithm in the CORDA model that, given an arbitrary vertex set P of size n having c components, solves the connection problem for P using m robots in any initial configuration,

1. for any $m \geq c-1$ in Case 1,
2. for any odd $m \geq c-1$ and any even $m \geq \min\{n-1, 2c-2\}$ in Case 2, and
3. for any odd $m \geq c-1$, any $m = 4k+2 \geq \min\{n-1, 2c-2\}$, and any $m = 4k \geq \min\{n-1, 4c-4\}$ in Case 3.

These lower bounds on the number of robots are tight, in the sense that there exist instances (i.e., P together with the robots' initial positions and local coordinate systems) in which the connection problem cannot be solved using fewer robots by any deterministic algorithm.

We prove the theorem in Section 2, and give some concluding remarks in Section 3.

2 AN ALGORITHM FOR THE CONNECTION PROBLEM

Given a set $P = \{p_1, \ldots, p_n\}$ of vertices, define Z_1, Z_2, Z_3 and Z_4 to be the following four coordinate systems, where

1. Z_i, $i = 1, 2, 3, 4$, has all points in P in its first quadrant, with at least one point in P on its x-axis and at least one point in P on its y-axis, and
2. the directions of the positive x-axes of Z_1, Z_2, Z_3 and Z_4 are east, north, west, and south, respectively, of the global coordinate system.

(All coordinate systems we discuss are right-handed.) See Figure 1.

For $i = 1, 2, 3, 4$, let $[Z_i]$ be the description of the coordinates in Z_i of the points in P under some encoding scheme (e.g., $[Z_i]$ lists the coordinates of the points in P in nondecreasing order of their x-coordinates, and in nondecreasing order of their y-coordinates for each x-coordinate). We can then order $[Z_1]$, $[Z_2]$, $[Z_3]$ and $[Z_4]$ lexicographically.

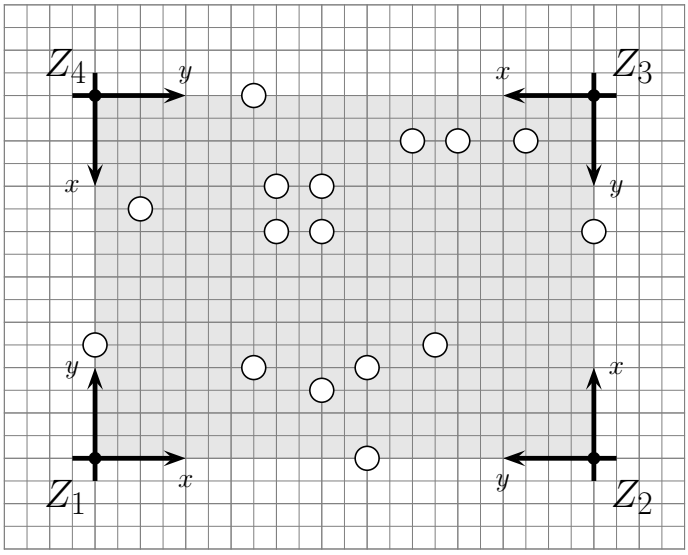

Fig. 1. Coordinate systems Z_1, Z_2, Z_3 and Z_4. Hollow circles represent the points in P.

Lemma 2. One of the following holds.

1. $[Z_1]$, $[Z_2]$, $[Z_3]$ and $[Z_4]$ are all distinct.
2. $[Z_1] = [Z_3] \neq [Z_2] = [Z_4]$.
3. $[Z_1] = [Z_2] = [Z_3] = [Z_4]$.

Proof. If $[Z_1] = [Z_3]$, then P looks the same in Z_1 and Z_3; so it must also look the same in Z_2 and Z_4, and thus $[Z_2] = [Z_4]$. If $[Z_1] = [Z_2]$, then P looks the same in Z_1 and Z_2; so it must also look the same in Z_2 and Z_3, and thus $[Z_2] = [Z_3]$. Continuing this argument, we obtain $[Z_1] = [Z_2] = [Z_3] = [Z_4]$. □

In the following, for each of the three possibilities given in Lemma 2 we present an algorithm for solving the connection problem. Let c be the number of components of P.

Case 1: $[Z_1]$, $[Z_2]$, $[Z_3]$ and $[Z_4]$ are all distinct.

Algorithm 1 (sketch): Suppose $[Z_1]$ is the "smallest" among $[Z_1]$, $[Z_2]$, $[Z_3]$ and $[Z_4]$ in the ordering defined above. We then use Z_1 to define a set T_1 of $c-1$ "target points" on its x-axis such that placing a robot at every target point solves the connection problem for P. (All other cases are handled similarly, using Z_2, Z_3 or Z_4 instead of Z_1.) All references to a coordinate system in the following refer to Z_1. Let $C_1, C_2, \ldots, C_c$ be the components of P. Since at least one point in P lies on the x-axis, exactly one component has a point on the x-axis. For

each component C_j that does not have a point on the x-axis, let (x_j, y_j) be the point in C_j having the smallest y-coordinate among those having the smallest x-coordinate. We define the *target point* for C_j to be $(x_j, 0)$, and say that point (x_j, y_j) *contributes* $(x_j, 0)$ (in the sense that $(x_j, 0)$ is a projection of (x_j, y_j) onto the x-axis of Z_1). Let T_1 be the set of target points for all such C_j, where $|T_1| = c - 1$. Placing one robot at each point in T_1 (or "covering T_1") solves the connection problem for P. See Figure 2.

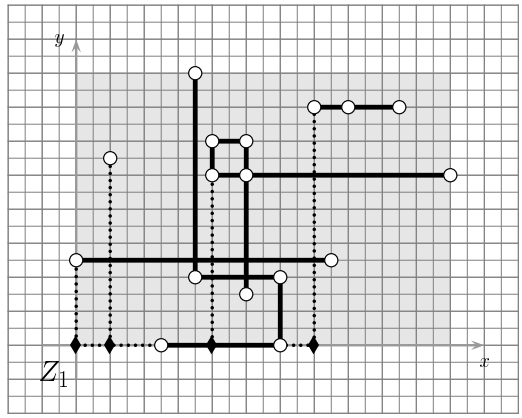

Fig. 2. Target points in Case 1 shown as solid lozenges. For each target point a dotted line indicates the point in P that contributes it. The points of P forming a component are connected by solid line segments.

The overall strategy of the robots (based on the computation described above) is as follows. Assume that there are at least $c - 1$ robots. The robots attempt to move to the x-axis, until $c - 1$ or more robots lie on the x-axis. Then the robots on the x-axis move on the x-axis and cover T_1.

It is not hard to see that an oblivious algorithm can be constructed in the CORDA model that accomplishes the above without creating multiplicities. Here is an outline. Suppose that there are fewer than $c - 1$ robots on the x-axis. Once a state is reached in which every robot occupies some vertex, some fixed strategy chooses, from among those robots not on the axis and having no robot between themselves and the axis, a unique robot that now moves across one edge toward the axis. (The "next" position for the chosen robot is the vertex adjacent to its current position toward the axis.) The strategy may first force some robots currently on the axis to move on the axis to "make room" for the incoming robot. Clearly such a strategy can be designed so that at any time, exactly one robot is allowed to move to an adjacent empty vertex and hence, no multiplicities will be created. (Of course, a more elaborate strategy can be constructed that, in certain situations, moves multiple robots toward the axis concurrently without the risk of creating multiplicities.) Once a state is reached

in which there are at least $c-1$ robots on the x-axis, the robots on the axis move on the axis so that every target point in T_1 will be occupied by one robot. This can easily be done by fixing some strategy that assigns the robots to the target points.

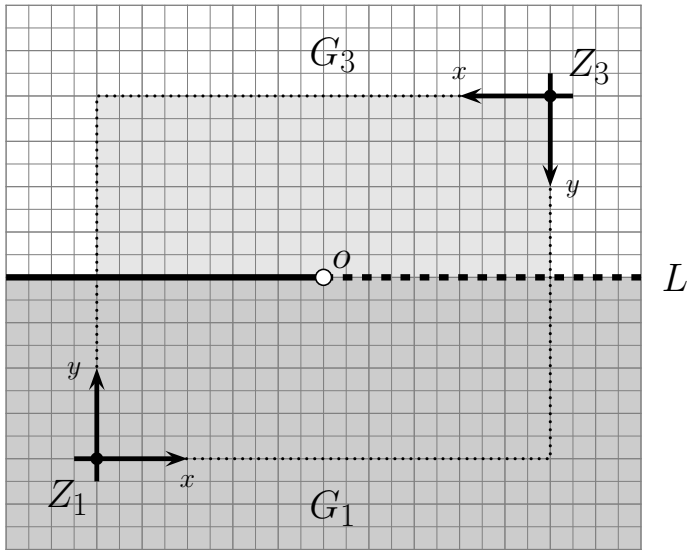

Fig. 3. Division of G into two subgrids G_1 and G_3 in Case 2. Subgrid G_1, shown dark shaded, contains the part of L shown by the solid line but not the part shown by the dotted line.

Case 2: $[Z_1] = [Z_3] \neq [Z_2] = [Z_4]$.

Algorithm 2 (sketch): Suppose $[Z_1] = [Z_3] < [Z_2] = [Z_4]$. We then use Z_1 and Z_3. (The other case is handled similarly using Z_2 and Z_4.) Divide G into two subgrids G_1 and G_3 along line L parallel to and equidistant from the x-axes of Z_1 and Z_3. See Figure 3. Let o be the point on L equidistant from the origins of Z_1 and Z_3. The points on L to the west of o belong to G_1, and the points on L to the east of o belong to G_3. (Point o does not belong to either subgrid. Note that if o is a vertex, then there may be a robot there.) Let T_1 be the set of target points for P computed as in Case 1 using Z_1. Similarly, let T_3 be the set of target points for P computed using Z_3. All target points in T_1 are on the x-axis of Z_1, all target points in T_3 are on the x-axis of Z_3, and $|T_1| = |T_3| = c-1$. Let $\#G_1$ and $\#G_3$ be the numbers of robots (in the current configuration) in G_1 and G_3, respectively. We connect P using the following strategy.

2.1: If $\#G_1 > \#G_3$, then the robots execute Algorithm 1 with respect to Z_1 and cover T_1. (Since in this case the robots never move away from the x-axis

of Z_1, the condition $\#G_1 > \#G_3$ continues to hold during the execution of Algorithm 1.) The strategy works for any $m \geq c - 1$.

2.2: If $\#G_1 < \#G_3$, then the robots execute Algorithm 1 with respect to Z_3 and cover T_3. The strategy works for any $m \geq c - 1$.

2.3: If $\#G_1 = \#G_3$ and there is a robot at o, then first the robot at o moves to one of its adjacent empty vertices (and hence, into G_1 or G_3). This move is deterministic, based on the robot's local coordinate system. (If all vertices adjacent to o are occupied, then some other robots will have to move first to "make room" for the robot at o to move. This can easily be done while maintaining $\#G_1 = \#G_3$ using some deterministic strategy. We omit the details.) Once the condition $\#G_1 \neq \#G_3$ is satisfied, we proceed as in 2.1 or 2.2. The strategy works for any odd $m \geq c - 1$.

2.4: If $\#G_1 = \#G_3$ and there is no robot at o, then the robots in each subgrid cover the "essential" target points in their subgrid, as described below.

Recall from Case 1 that each connected component C_j of P that does not have a point on the x-axis of G_1 or G_3 contributes two target points, one in T_1 on the x-axis of Z_1, and another in T_3 on the x-axis of Z_3. If the coordinates of these target points are identical, then C_j is said to be *symmetric*; otherwise C_j is *asymmetric*. Note that if C_j is symmetric, then its target point $(x_i, 0)$ (in Z_1 and in Z_3) satisfies $x_i \leq a$, where $o = (a, b)$ in Z_1 and Z_3. See Figure 4.

We assume in the following that at least one point in P lies on the x-axis of Z_1 (and Z_3) at coordinates $(x, 0)$ with $x \leq a$. The case in which this condition is not satisfied can be handled in a symmetric manner with respect to the vertical line through o.

2.4(a): If there exists at least one symmetric component, then the robots in G_1 (resp. G_3) cover those target points $(x_i, 0)$ in T_1 (resp. T_3) such that $x_i \leq a$. See Figure 4(a). We call these target points *essential*. Covering the essential target points in T_1 and T_3 ensures that all components are connected, in part via some symmetric component.

By the assumption on the existence of a point $(x, 0)$, $x \leq a$, on the x-axis of Z_1 and Z_3, there exist at least two points that lie on the x-axis of Z_1 or Z_3 *and* do not contribute any essential target point. Any other point – there are at most $n-2$ of them – contributes at most one essential target point, except, if there is a point at o that forms a component by itself, then it contributes two essential target points, one in T_1 and another in T_3. Thus the total number of essential target points in this case is at most $n-1$. Since clearly the total number of essential target points is at most $2c-2$, the strategy works for any even $m \geq \min\{n-1, 2c-2\}$.

2.4(b): If all components are asymmetric, then in addition to the essential target points identified in 2.4(a), we may designate one additional target point in T_1 and in T_3 as essential; if there is at least one target point $(x, 0)$ in T_1 (and T_3) such that $x > a$, then the one among these having the

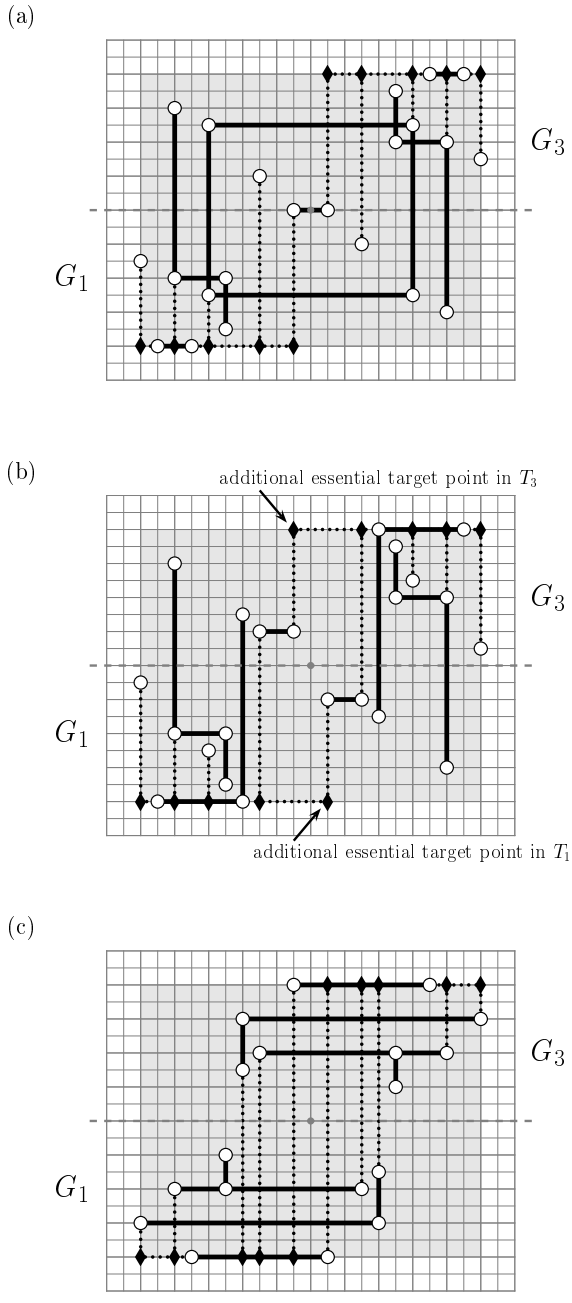

Fig. 4. Essential target points. (a) Case 2.4 (a), (b) Case 2.4 (b) with additional essential target points, and (c) Case 2.4 (b) without additional essential target points.

smallest x coordinate, say $(a', 0)$, is considered essential as well. (If such a target point does not exist, then no additional target point is designated as essential.) See Figure 4(b)(c). Covering these essential target points in T_1 and T_3 ensures that all components are connected, possibly via the components having a target point at $(a', 0)$ in Z_1 or Z_3.
Note that if there was a point at o, then the component containing o would be symmetric. Thus $o \notin P$, n is even, and there can be only two points (namely, those that project to $(a', 0)$ in Z_1 or Z_3) that contribute two essential target points. Thus the total number of essential target points is at most n, and the strategy works for any even $m \geq \min\{n, 2c-2\}$. (Since $n-1$ is odd, P can be connected using $m = n-1$ robots as in Cases 2.1, 2.2 and 2.3.)

In summary, P can be connected using m robots for any odd $m \geq c-1$ and any even $m \geq \min\{n-1, 2c-2\}$.

Remark 3. One can easily construct an instance in Case 2 in which $c = n-1$ and hence $n-1$ robots are necessary. Figure 5(a) shows an instance in Case 2.4(a) in which $n = 18$, $c = 4$ and the positions of the $2c-4 = 4$ robots are symmetric with respect to o. Under any deterministic algorithm, starting from this configuration the robots may always move symmetrically with respect to o and fail to connect all components. Thus if the number of robots is even, then $2c-2$ robots are necessary. Figure 5(b) shows an instance in Case 2.4(b) in which, if the number of robots is even, then $2c-2$ robots are necessary.

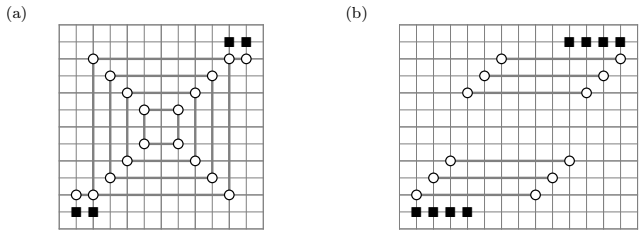

Fig. 5. Instances in which $2c-4$ robots are not enough to connect P in (a) Case 2.4 (a) with $c = 4$, and (b) Case 2.4 (b) with $c = 6$. The black squares are the robots' positions.

Case 3: $[Z_1] = [Z_2] = [Z_3] = [Z_4]$.

Algorithm 3 (sketch): Briefly, we do as in Case 2 using Z_1, Z_2, Z_3 and Z_4. Given P, we compute the set of target points T_1, T_2, T_3 and T_4, in terms of Z_1, Z_2, Z_3 and Z_4, respectively, where $|T_1| = |T_2| = |T_3| = |T_4| = c-1$. Define four subgrids G_1, G_2, G_3 and G_4 as shown in Figure 6 (point o does not belong to any subgrid), and let $\#G_i$ be the numbers of robots (in the current configuration) in G_i, $i = 1, 2, 3, 4$. We connect P using the following strategy.

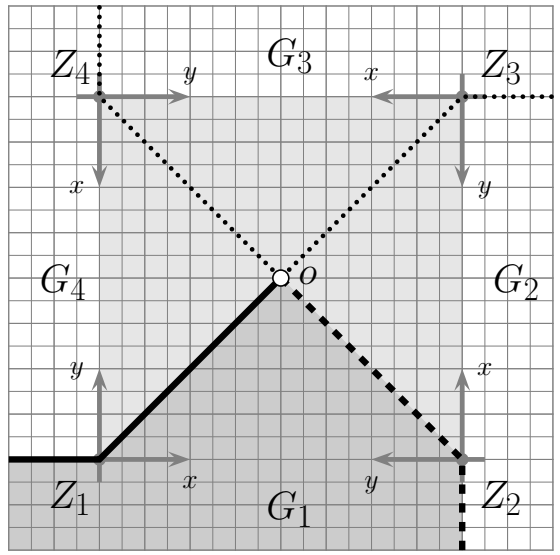

Fig. 6. Division of G into four subgrids G_1, G_2, G_3 and G_4 in Case 3. Subgrid G_1 is shown dark shaded.

3.1: If $\#G_1 > \#G_2, \#G_3, \#G_4$, then first move all robots to G_1 without invalidating the condition using some deterministic strategy and then cover T_1 using all robots, without allowing any robot to leave G_1 except when $(2a, 0)$ of Z_1 is a target point and a robot moves there. (We omit the details of this trivial operation.) Cases in which all robots are moved to other subgrids are defined symmetrically. The strategy works for any $m \geq c - 1$.

3.2: If (i) $\#G_1 = \#G_2 > \#G_3, \#G_4$, (ii) $\#G_1 = \#G_2 = \#G_3 > \#G_4$, or (iii) $\#G_1 = \#G_3 > \#G_2 > \#G_4$, then first we move a robot in G_2 to G_1 using some deterministic strategy and then proceed as in 3.1. (As in 2.3, some robots in G_1 may have to move first to "make room" for a robot in G_2 to enter G_1. This can easily be done using some deterministic strategy.) All cases symmetric to this are handled in a similar manner. The strategy works for any $m \geq c - 1$.

3.3: If $\#G_1 = \#G_3 > \#G_2 = \#G_4$ and there is a robot at o, then first we move the robot at o to G_1 or G_3. This move is deterministic, based on the robot's local coordinate system. (Again, if the vertices adjacent to o in G_1 and G_3 are occupied, then some robots in G_1 and G_3 will have to move first to "make room" for the robot at o to move. This can easily be done using some deterministic strategy.) Once the condition $\#G_1 \neq \#G_3$ is satisfied, we proceed as in 3.1. All cases symmetric to this are handled in a similar manner. The strategy works for any odd $m \geq c - 1$.

3.4: If $\#G_1 = \#G_2 = \#G_3 = \#G_4$ and there is a robot at o, then first move the robot at o to one of the subgrids deterministically, and then proceed as in 3.1. This is similar to 3.3. The strategy works for any $m = 4k + 1 \geq c - 1$.

3.5: If $\#G_1 = \#G_3 > \#G_2 = \#G_4$ and there is no robot at o, then without allowing any robot to occupy o, first we move the robots in G_2 to G_1 and those in G_4 to G_3. (If the condition for 3.1 holds while the robots are changing subgrids, then we proceed as in 3.1.) Once a configuration in which $\#G_1 = \#G_3$ and $\#G_2 = \#G_4 = 0$ is reached, we connect P by covering T_1 and T_3 as in 2.4, without allowing the robots to leave G_1 or G_3 except when they move to a target point at $(2a, 0)$. Note that the robots can identify Z_1 and Z_3 (instead of Z_2 and Z_4) as the coordinate systems to use, based on their distribution in the four subgrids. All cases symmetric to this are handled in a similar manner. By the analysis given in 2.4, the strategy works for any even $m \geq \min\{n - 1, 2c - 2\}$.

3.6: If $\#G_1 = \#G_2 = \#G_3 = \#G_4$ and there is no robot at o, then the robots in each subgrid cover certain "essential" target points in their subgrid, as in 2.4. For the purpose here we call a component of P *symmetric* if its four target points in T_i, $i = 1, 2, 3, 4$, have identical coordinates in the respective coordinate systems Z_i; otherwise it is called *asymmetric*. Let $o = (a, a)$ (in any Z_i). As in 2.4, assume that at least one point in P lies on the x-axis of Z_i, $i = 1, 2, 3, 4$, at coordinates $(x, 0)$ with $x \leq a$. There are two cases.

3.6(a): If there exists at least one symmetric component, then the robots in G_i cover those target points $(x, 0)$ in T_i such that $x \leq a$ and either (i) $(x, 0)$ is contributed by a point in P that belongs to a symmetric component, or (ii) $0 < y < a$ holds for some point (x, y) in P. See Figure 7(a). We call such target points *essential*. We leave it to the reader to verify that covering the essential target points in all T_i, $i = 1, 2, 3, 4$ ensures that all components are connected, in part via some symmetric component. Again, a simple counting argument shows that the total number of essential target points is at most $n - 1$. (By the assumption on the existence of a point $(x, 0)$, $x \leq a$, on the x-axis of Z_i, $i = 1, 2, 3, 4$, there exist at least four points that lie on an x-axis *and* do not contribute any essential target point. Any other point – there are at most $n - 4$ of them – contributes at most one essential target point, except, if there is a point at o that forms a component by itself, then it contributes four essential target points.) Since the total number of essential target points is at most $4c - 4$, the strategy works for any $m = 4k \geq \min\{n - 1, 4c - 4\}$.

3.6(b): If all components are asymmetric, then in addition to the essential target points identified in 3.6(a), we designate $(0, 0)$ of Z_i, $i = 1, 2, 3, 4$, as an additional essential target point, provided that $(0, 0)$ is not yet

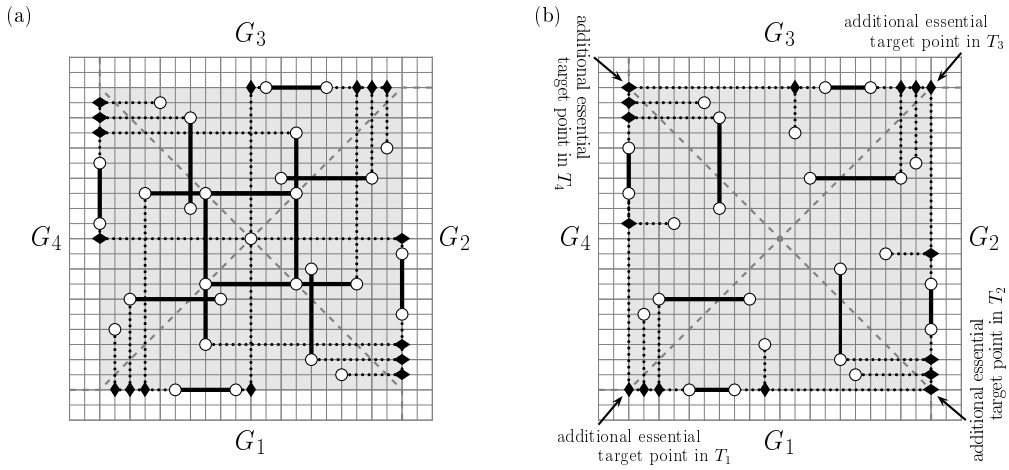

Fig. 7. Essential target points in (a) Case 3.6 (a) and (b) Case 3.6 (b)

identified as an essential target point. See Figure 7(b). We leave it to the reader to verify that covering the essential target points including $(0, 0)$ in all T_i, $i = 1, 2, 3, 4$ ensures that all components are connected. The total number of essential target points is at most n. (The counting argument is similar to that in 3.6(a), except that no point exists at o and there can be four additional essential target points.) Thus the strategy works for any $m = 4k \geq \min\{n, 4c - 4\}$.

In summary, P can be connected using m robots for any odd $m \geq c - 1$, any $m = 4k + 2 \geq \min\{n - 1, 2c - 2\}$, and any $m = 4k \geq \min\{n - 1, 4c - 4\}$.

Remark 4. Figure 8 shows instances that demonstrate the tightness of the bounds obtained for 3.5, 3.6(a) and 3.6(b).

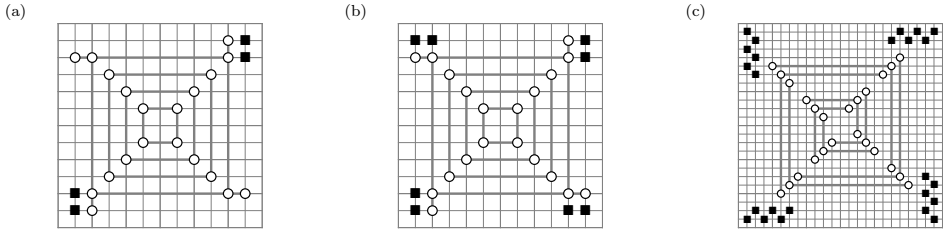

Fig. 8. (a) Case 3.5 in which $2c - 4$ robots are not enough to connect P having $c = 4$. (b) Case 3.6 (a) in which $4c - 8$ robots are not enough to connect P having $c = 4$. (c) Case 3.6 (b) in which $4c - 8$ robots are not enough to connect P having $c = 6$.

Theorem 1 follows from the discussion given above. Note that the three algorithms can be combined into one, because the robots can always decide which of the three cases applies, and obviously, such an algorithm can be constructed in the CORDA model.

3 CONCLUDING REMARKS

We presented a distributed algorithm for connecting a given set of grid vertices under straight-line visibility using a number of autonomous mobile robots that can function as relays. The number of robots required for an input set P critically depends on the type of symmetry of P. It is worth pointing out that if the robots' local coordinate systems do not agree on the orientation, then creating a multiplicity may be unavoidable when solving the connection problem. See Figure 9 for such an example.

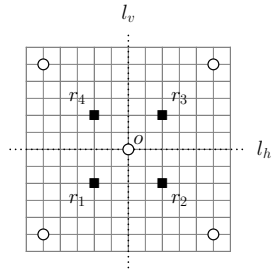

Fig. 9. To connect the two components, a robot must move to either row l_h or column l_v. If the local coordinate systems of r_1 and r_3 are right-handed and those of r_2 and r_4 are left-handed, then it is possible that r_1 and r_3 always move symmetrically with respect to o, r_2 and r_4 always move symmetrically with respect to o, and r_1 and r_2 always move symmetrically with respect to l_v. This means that a multiplicity can be created when a robot reaches l_h or l_v.

For future study, the connection problem can be considered in the 2D plane under a suitable assumption on the module's communication capabilities. For instance, the plane may contain polygonal obstacles that block visibility and two modules can communicate with each other if and only if they see each other within a certain distance (for a relevant variant in wireless sensor networks, see for example [5, 6]). One can also consider the problem of constructing a "fault-tolerant" network, where the objective is to establish $k \geq 2$ disjoint paths between any two components. In wireless sensor networks, a variation of this problem has been studied in both static and dynamic settings [1, 14, 15, 16].

Acknowledgment

We would like to thank the anonymous referees for their insightful comments and constructive suggestions, which have helped us improve our paper.

REFERENCES

[1] Abbasi, A. A.—Younis, M. F.—Akkaya, K.: Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks. IEEE Transactions on Parallel and Distributed Systems, Vol. 20, 2009, No. 9, pp. 1366–1379.

[2] Akkaya, K.—Younis, M. F.: Coverage and Latency Aware Actor Placement Mechanisms in WSANs. International Journal of Sensor Networks, Vol. 3, 2008, No. 3, pp. 152–164.

[3] Akyildiz, I. F.—Su, W.—Sankarasubramaniam, Y.—Cayirci, E.: Wireless Sensor Networks: A Survey. Computer Networks, Vol. 38, 2002, No. 4, pp. 393–422.

[4] Ando, H.—Oasa, Y.—Suzuki, I.—Yamashita, M.: A Distributed Memoryless Point Convergence Algorithm for Mobile Robots With Limited Visibility. IEEE Transactions on Robotics and Automation, Vol. 15, 1999, No. 5, pp. 818–828.

[5] Călinescu, G.—Tongngam, S.: Relay Nodes in Wireless Sensor Networks. WASA 2008, Lecture Notes in Computer Science, Vol. 5258, 2008, pp. 286–297.

[6] Cheng, X.—Du, D.-Z.—Wang, L.—Xu, B.: Relay Sensor Placement in Wireless Sensor Networks. Wireless Networks, Vol. 14, 2008, No. 3, pp. 347–355.

[7] Cohen, R.—Peleg, D.: Convergence of Autonomous Mobile Robots With Inaccurate Sensors and Movements. SIAM Journal on Computing, Vol. 38, 2008, No. 1, pp. 276–302.

[8] Cohen, R.—Peleg, D.: Local Spreading Algorithms for Autonomous Robot Systems. Theoretical Computer Science, Vol. 399, 2008, No. 1-2, pp. 71–82.

[9] Culler, D.—Estrin, D.—Srivastava, M.: Overview of Sensor Networks. IEEE Computer, Vol. 37, 2004, No. 8, pp. 41–49.

[10] Défago, X.—Souissi, S.: Non-Uniform Circle Formation Algorithm for Oblivious Mobile Robots With Convergence Toward Uniformity. Theoretical Computer Science, Vol. 396, 2008, No. 1-3, pp. 97–112.

[11] Dessmark, A.—Fraigniaud, P.—Kowalski, D. R.—Pelc, A.: Deterministic Rendezvous in Graphs. Algorithmica, Vol. 46, 2006, No. 1, pp. 69–96.

[12] Efrima, A.—Peleg, D.: Distributed Algorithms for Partitioning a Swarm of Autonomous Mobile Robots. SIROCCO 2007, Lecture Notes in Computer Science, Vol. 4474, 2007, pp. 180–194.

[13] English, J.—Wiacek, M.—Younis, M.: CORE: Coordinated Relocation of Sink Nodes in Wireless Sensor Networks. Proceedings of the 23rd Biennial Symposium on Communications, 2006, pp. 320–323.

[14] Han, X.—Cao, X.—Lloyd, E. L.—Shen, Ch.-Ch.: Fault-Tolerant Relay Nodes Placement in Heterogeneous Wireless Sensor Networks. IEEE Transactions on Mobile Computing, Vol. 9, 2010, No. 5, pp. 643–656.

[15] Hao, B.—Tang, J.—Xue, G.: Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Formulation and Approximation. Proceedings of the Workshop on High Performance Switching and Routing, 2004, pp. 246–250.

[16] Kashyap, A.—Shayman, M.: Relay Placement and Movement Control for Realization of Fault-Tolerant Ad-Hoc Networks. Proceedings of the 41st Annual Conference on Information Sciences and Systems, 2007, pp. 783–788.

[17] Kranakis, E.—Krizanc, D.—Markou, E.: Mobile Agent Rendezvous in a Synchronous Torus. LATIN '06, Lecture Notes in Computer Science, Vol. 3887, 2006, pp. 653–664.

[18] Kranakis, E.—Krizanc, D.—Santoro, N.—Sawchuk, C.: Mobile Agent Rendezvous in the Ring. Proceedings of the 23rd International Conference on Distributed Computing Systems, ICDCS '03, 2003, pp. 592–599.

[19] Kranakis, E.—Krizanc, D.—Rajsbaum, S.: Mobile Agent Rendezvous: A Survey. SIROCCO '06, Lecture Notes in Computer Science, Vol. 4056, 2006, pp. 1–9.

[20] Lulu, L.—Elnagar E.: Efficient and Complete Coverage of 2D Environments by Connectivity Graphs for Motion Planning Algorithms. International Journal of Information Science and Engineering, Vol. 22, 2006, pp. 1355–1366.

[21] Lulu, L.—Elnagar E.: An Art Gallery-Based Approach: Roadmap Construction and Path Planning in Global Environments. International Journal of Robotics and Automation, Vol. 22 , No. 4, pp. 329–339.

[22] Prencipe, G.: On the Feasibility of Gathering by Autonomous Mobile Robots. SIROCCO '05, Lecture Notes in Computer Science, Vol. 3499, 2005, pp. 246–261.

[23] Sugihara, S.—Suzuki, I.: Distributed Algorithms for Formation of Geometric Patterns With Many Mobile Robots. Journal of Robotic Systems, Vol. 13, 1996, No. 3, pp. 127–139.

[24] Suzuki, I.—Yamashita, M.: Distributed Anonymous Mobile Robots – Formation of Geometric Patterns. SIAM Journal on Computing, Vol. 28, 1999, No. 4, pp. 1347–1363.

[25] Younis, M.— Akkaya, K.: Strategies and Techniques for Node Placement in Wireless Sensor Networks. Ad Hoc Networks, No. 6, 2008, pp. 621–655.

Adrian Kosowski received his Ph. D. degree in computer science in 2007 from the Gdańsk University of Technology, Poland, where he has since been working as an Assistant Professor. He is currently a researcher at the INRIA Bordeaux Sud-Ouest center in France. His scientific interests include combinatorial optimization problems in graph theory and distributed computing.

Ichiro Suzuki received his D. E. degree in information and computer sciences from Osaka University, Japan, in 1983. He is currently a Professor of computer science at the University of Wisconsin–Milwaukee. He has held visiting positions at Osaka University and Kyushu University. His research interests include distributed/concurrent systems, computational geometry, and computational robotics. He is a member of the Association for Computing Machinery.

Paweł Żyliński received his Ph. D. degree in mathematics from the University of Gdańsk, Poland, in 2004. He is currently an Assistant Professor at the University of Gdańsk. His research interests include graph theory and computational geometry.