Computing and Informatics, Vol. 31, 2012, 349–369

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 Żyliński

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

Communicated by Boleslav Szymanski

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. We assume a weak *fairness* condition that guarantees that every robot executes the cycle infinitely many times. Here, 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 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 $c \geq 2$ 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;
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 having $c \geq 2$ components, 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.

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. $[Z_1] = [Z_2] = [Z_3] = [Z_4]$.

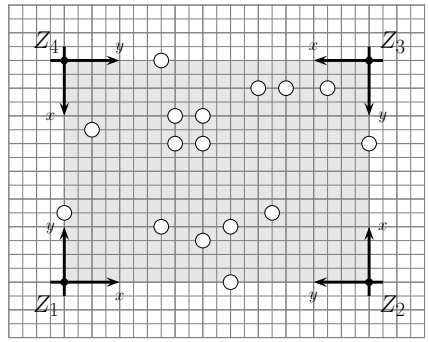

Fig. 1. Coordinate systems Z_1, Z_2, Z_3 and Z_4. Hollow circles represent the points in P.

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. We enforce the following rules in the algorithms. Recall that initially all robots are located at distinct vertices of the grid.

1. In a single Move step, a robot either remains stationary or moves to a vertex adjacent to the vertex it currently occupies.
2. A robot that sees another robot r in the interior of an edge (i.e., r is moving) in a Look step does not move in that cycle. That is, a robot may move in the Move step of a cycle only if it observes in the Look step a configuration in which every robot occupies a vertex.

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 call point (x_j, y_j) the *representative* of C_j with respect to Z_1. We define the *target point* for C_j to be $(x_j, 0)$, and say that C_j *contributes* $(x_j, 0)$ or 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.

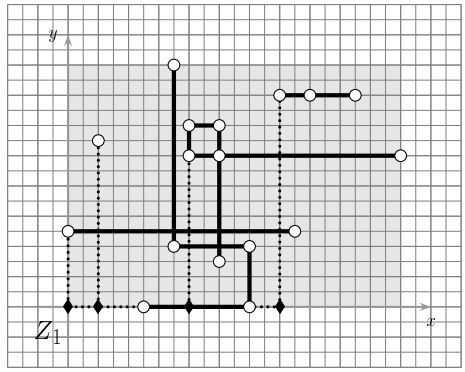

Fig. 2. Target points in Case 1 shown as solid lozenges. For each target point a dotted line indicates the point in P that contributes it. The points of P forming a component are connected by solid line segments.

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 move to the x-axis one by one according to some deterministic strategy, 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. No robot ever moves away from the x-axis while executing Algorithm 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. Some deterministic 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.

In summary, P can be connected in Case 1 using $m \geq c-1$ robots. (End of Case 1.)

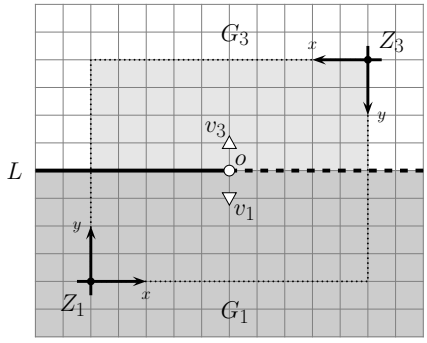

Fig. 3. Division of G into two subgrids G_1 and G_3 in Case 2. Subgrid G_1, shown dark shaded, contains the part of L shown by the solid line but not the part shown by the dashed line.

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.) Let $o = (a, b)$ (in Z_1 and Z_3) be the point equidistant from the origins of Z_1 and Z_3, and L the line through o parallel to the x-axes of Z_1 and Z_3. (Note that o may or may not be in G.) Divide G into two subgrids G_1 and G_3 along L, where the points in $G \cap L$ to the west of o belong to G_1, the points in $G \cap L$ to the east of o belong to G_3, and o does not belong to either subgrid; see Figure 3. 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 use different strategies depending on m.

2.1: Odd $m \geq c-1$.

2.1.1: If there is a robot r at o (this means that o is a vertex), then we move r into G_1 or G_3, and then proceed to 2.1.2. This is done as follows. For $i \in \{1, 3\}$, let $v_i = (a, b-1)$ of Z_i be the vertex in G_i adjacent to o in the direction toward the x-axis of Z_i (see Figure 3).

1. If $\#G_1 > \#G_3$, then robot r waits at o while the robots in G_1 move within G_1 to make v_1 empty using some deterministic procedure (if it is currently occupied), and then moves to v_1.
2. Symmetrically, if $\#G_1 < \#G_3$ then r moves to v_3 after the robots in G_3 empty v_3 by moving within G_3.

3. If $\#G_1 = \#G_3$, then r waits at o until *both* v_1 and v_3 become empty while the robots in G_1 and the robots in G_3 empty v_1 and v_3 within their respective subgrids, and then moves deterministically to one of them based on its local coordinate system.

Note that in all cases, r leaves o only after all other robots have finished the procedure of emptying v_1 and/or v_3. This (together with the fact that no robot ever moves to o in 2.1.2) ensures that there is no "confusion" among the robots as to whether they are in 2.1.1 or 2.1.2, regardless of the delay in their execution cycles.

2.1.2: Suppose there is no robot at o (o may or may not be a vertex). Note that $\#G_1 \neq \#G_3$ since m is odd. We move enough robots to the subgrid having a larger number of robots and connect P using Algorithm 1 within that subgrid. This is done as follows. Assume $\#G_1 > \#G_3$. (The case $\#G_1 < \#G_3$ is handled similarly.) Let T_1 be the set of target points on Z_1's x-axis, where $|T_1| = c - 1$.

a) If $\#G_1 < c-1$, then a single robot uniquely identified in G_3 moves to G_1 without passing through o, using some deterministic procedure. Since $m \geq c - 1$, eventually we reach 2.1.2 b).

b) If $\#G_1 \geq c-1$, then the robots in G_1 execute Algorithm 1 and cover T_1 without leaving G_1. Observe that since the robots never move away from the x-axis of Z_1 during the execution of Algorithm 1, the conditions $\#G_1 > \#G_3$ and $\#G_1 \geq c-1$ continue to hold (and hence we remain in 2.1.2 b)) until T_1 is covered.

In summary, P can be connected using m robots for any odd $m \geq c - 1$. (End of 2.1.)

2.2: $m = 2k \geq \min\{n - 1, 2c - 2\}$.

2.2.1: If there is a robot r at o, then as in 2.1.1, r moves to G_1 or G_3, whichever has more robots. We then reach 2.2.2.2.

2.2.2: Suppose there is no robot at o. There are two cases.

2.2.2.1 Suppose $\#G_1 = \#G_3 = k$. As in Case 1, let T_1 be the set of target points for the components of P on Z_1's x-axis. Symmetrically, let T_3 be the set of target points for the components of P on Z_3's x-axis. Using Algorithm 1, the robots in G_1 (resp. G_3) cover *some* of the target points in T_1 (or T_3) that are considered "essential".
Specifically, a component of P is said to be *symmetric* if it "looks the same" in Z_1 and Z_3, i.e., the set of coordinates of the points in it is the same in Z_1 and Z_3. Observe that in Case 2, every component C is of one of the following four types. The target point(s) that C contributes is (are) designated as *essential* in some cases, as mentioned below. (See Figure 4.)

type 1: C is symmetric, has a point on both the x-axis of Z_1 and the x-axis of Z_3, and contributes no target point.

type 2: C is symmetric, has no point on either of the two x-axes, and contributes exactly two target points $t_1 \in T_1$ and $t_3 \in T_3$. The x-coordinate of t_1 in Z_1 is the same as that of t_3 in Z_3. Both t_1 and t_3 are designated as essential.

type 3: C is not symmetric, has a point on one of the two x-axes (but not both), and contributes exactly one target point t, in either T_1 or T_3. In Case 2.2.2.1 a) below, t is not designated as essential, while in Case 2.2.2.1 b) below, t becomes essential.

type 4: C is not symmetric, has no point on either of the two x-axes, and contributes exactly two target points $t_1 \in T_1$ and $t_3 \in T_3$. The x-coordinate of t_1 in Z_1 and that of t_3 in Z_3 are different. Whichever of the two having a smaller x-coordinate is designated as essential.

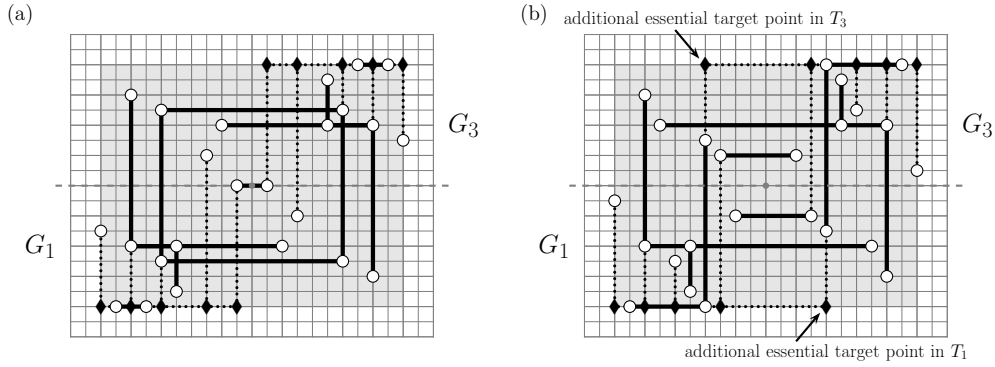

Fig. 4. Essential target points in a) Case 2.2.2.1 a) and b) Case 2.2.2.1 b)

2.2.2.1 a): If there exists at least one symmetric component, then covering all essential target points connects all components. This is because of the following (see Figure 4 a)):

- Every component either contains a point or has an essential target point on one of the two x-axes.
- The points on the two x-axes either belong to one symmetric component or are connected through a symmetric component and its two essential target points.

The number of essential target points is at most $n-1$, because, of the n points of P, at least two points that lie on the x-axis of Z_1 or Z_3 do not contribute any essential target point, and any other point 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. Also, the number of essential target points is at most $|T_1 \cup T_3| = 2c-2$. Thus the above strategy works for any even $m \geq \min\{n-1, 2c-2\}$.

2.2.2.1 b): Suppose no symmetric component exists. Then, since the x-axis of every Z_i contains at least one point of P, there exist exactly two components of type 3, C' having a point on the x-axis of Z_1 and C'' having a point on the x-axis of Z_3. Covering the essential target points that C' and C'' contribute in the description of type 3 connects the points of P on the x-axis of Z_1 and the points of P on the x-axis of Z_3. Thus covering all essential target points, including those that C' and C'' contribute, connects all components (see Figure 4 b)).

Since every component contributes exactly one essential target point, the total number of essential target points is c, and thus the strategy works for any even $m \geq c$.

2.2.2.2: If $\#G_1 \neq \#G_3$, then we proceed as in Case 2.1.2. That is, we move enough robots to the subgrid having more robots and connect P within that subgrid using Algorithm 1. It is sufficient to have $m \geq c-1$ robots to do so.

The number of robots needed is $c-1$ in 2.1, $\min\{n-1, 2c-2\}$ in 2.2.2.1 a), c in 2.2.2.1 b), and $c-1$ in 2.2.2.2. Note that

1. $2c-2 \geq c > c-1$ for $c \geq 2$, and
2. in 2.2.2.1 b) n is even[1] and hence any even $m \geq n-1$ satisfies $m \geq n \geq c$.

Thus P can be connected using m robots for any even $m \geq \min\{n-1, 2c-2\}$. (End of 2.2.)

In summary, P can be connected by m robots for any odd $m \geq c-1$ and any even $m \geq \min\{n-1, 2c-2\}$. (End of Case 2.)

Remark 3. One can easily construct an instance in Case 2 in which $c = n$ and hence $n-1$ robots are necessary. Figure 5 shows an instance in 2.2.2.1 a) in which $n = 18$, $c = 4$ and the positions of the $2c-4 = 4$ robots are symmetric with respect to o. Using 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 sometimes necessary.

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

Algorithm 3 (sketch): Roughly speaking, we do as in Case 2 using all four coordinate systems 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 $|T_1| = |T_2| = |T_3| = |T_4| = c-1$. Next, we define four subgrids G_1, G_2, G_3 and G_4 as shown in Figure 6, where point $o = (a, a)$ (in any Z_i) does not belong to any subgrid. Let $\#G_i$ be the number of robots (in the current configuration) in G_i, $i = 1, 2, 3, 4$.

[1] In Case 2, if n is odd then $o \in P$, which implies the existence of a symmetric component.

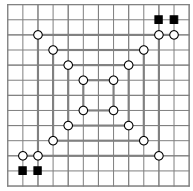

Fig. 5. An instance in which $2c-4$ robots are not enough to connect P in Case 2.2.2.1 a). Here $c=4$ and the black squares are the robots' positions.

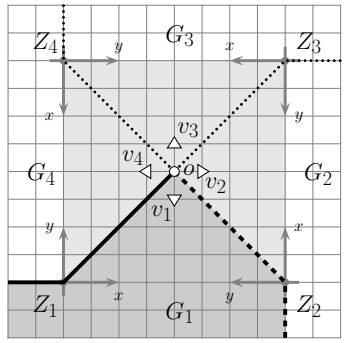

Fig. 6. Division of G into four subgrids G_1, G_2, G_3 and G_4 in Case 3. Subgrid G_1 is shown dark shaded.

We give a brief outline. First, if o is a vertex and there is a robot r at o, then we move it to one of the subgrids G_1, G_2, G_3 and G_4. If o is empty and it is possible to uniquely identify one of the subgrids having the largest number of robots, say G_i, based on $\#G_1$, $\#G_2$, $\#G_3$ and $\#G_4$, then we move enough robots to G_i and connect P within G_i by covering T_i using Algorithm 1. If on the other hand (due to symmetry) it is not possible to uniquely identify a subgrid, then we either connect P (i) in two opposite subgrids (G_1 and G_3, or G_2 and G_4) after moving enough robots there, or (ii) in all four subgrids, again using Algorithm 1 in each of these subgrids. When moving robots between subgrids, care must be taken so that there will be no "confusion" as to which robots are allowed to change subgrids, regardless of the delays in the robots' execution cycles.

Specifically, we connect P using m robots as follows.

3.1: Odd $m \geq c-1$.

3.1.1: Suppose that there is a robot, say r, at o. (This means that o is a vertex.) We first move r into one of the subgrids having the largest number of robots. This is done as follows. For $i=1,2,3,4$, let $v_i=(a,a-1)$ of Z_i be the vertex in G_i adjacent to o in the direction toward the x-axis of Z_i (see Figure 6). Robot r waits at o until vertex v_i becomes empty in *every* subgrid G_i that currently has the largest number of robots. Mean-

while, in each such subgrid G_i the robots move deterministically to make v_i empty without leaving G_i. Once these v_i's become all empty, r moves to one of them breaking ties deterministically based on its coordinate system. We then proceed to 3.1.2.

3.1.2: Suppose there is no robot at o (o may or may not be a vertex). We uniquely identify one of the subgrids having the largest number of robots, as follows: G_i is identified uniquely if and only if it maximizes the sum $\#G_i + \#G_{i+1} + \#G_{i+2}$ over all subgrids having the largest number of robots.[2] (In Case 3 the indices are taken cyclically over $1, 2, 3, 4$.) Now, let G_1 be the subgrid so identified. (The other cases are handled similarly.) Our goal is to move enough robots to G_1 while keeping o empty, and connect P using Algorithm 1 with respect to Z_1, by covering the target points in T_1 on Z_1's x-axis. Note that point $(2a, 0)$ of Z_1, which belongs to G_2, but not to G_1, may or may not be in T_1.

3.1.2.1: $(2a, 0) \notin T_1$.

a) If $\#G_1 < c - 1$ (and thus $\#G_2 + \#G_3 + \#G_4 \geq 1$), then a single robot moves to G_1 from G_2, G_3 or G_4. (We proceed to 3.1.2.1 b) when $\#G_1$ increases to $c - 1$.) This is accomplished as follows, so that G_1 will continue to be uniquely identified:

1. If $\#G_2 > 0$ then a single robot uniquely identified in G_2 moves to G_1;
2. if $\#G_2 = 0$ and $\#G_3 > 0$ then a single robot uniquely identified in G_3 moves to G_2;
3. if $\#G_2 = \#G_3 = 0$ and $\#G_4 > 0$ then a single robot uniquely identified in G_4 moves to G_3.

b) If $\#G_1 \geq c - 1$, then the robots in G_1 execute Algorithm 1 and cover T_1 without leaving G_1.

3.1.2.2: $(2a, 0) \in T_1$.

a) If $\#G_1 < c - 2$ (and thus $\#G_2 + \#G_3 + \#G_4 \geq 2$), then a single robot moves to G_1 from G_2, G_3 or G_4, as in 3.1.2.1 a).

b) If $\#G_1 \geq c - 2$, $\#G_2 + \#G_3 + \#G_4 \geq 1$ and $(2a, 0)$ is empty, then a single robot moves to $(2a, 0)$ from G_2, G_3 or G_4, using a strategy similar to that used in 3.1.2.1 a).

c) If $\#G_2 + \#G_3 + \#G_4 = 0$ (and thus $\#G_1 = m \geq c - 1$ and $(2a, 0) \in G_2$ is empty), then a single robot in G_1 moves to $(2a, 0)$. ($\#G_1 > \#G_2$ continues to hold after the move, because $(2a, 0) \in T_1$ implies $c \geq 4$ in Case 3.)

[2] It is easy to show that if $\#G_1 + \#G_2 + \#G_3 + \#G_4$ is odd, then the "sum-of-three" condition uniquely identifies a subgrid among those having the largest number of robots. We omit the details. Note that if we reach 3.1.2 from 3.1.1, then there is a unique subgrid having the largest number of robots.

d) If none of the above applies, i.e., if $\#G_1 \geq c-2$ and $(2a, 0)$ is occupied, then the robots in G_1 cover the target points in T_1 *except* $(2a, 0)$, by executing Algorithm 1 without leaving G_1.

Thus P is connected by m robots for any odd $m \geq c-1$. (End of 3.1)

3.2: $m = 4k+2 \geq \min\{n-1, 2c-2\}$.

3.2.1: If there is a robot, say r, at $o = (a, a)$, then r moves to the unique subgrid G_i that maximizes the sum $\#G_i + \#G_{i+1} + \#G_{i+2}$ among the subgrids having the largest number of robots.[3] (This may require the robots in G_i to first move deterministically within G_i to make vertex v_i empty, as in 3.1.1.)

Let us assume without loss of generality that G_1 is the subgrid uniquely identified above into which r moves and discuss which of the four cases of 3.2.2 applies after the move. Since G_1 has more robots than any other subgrid after the move, 3.2.2 a) and 3.2.2 c) do not apply. 3.2.2 b) does not apply either, because that would imply $\#G_1 = \#G_3$ and $\#G_1 + \#G_2 + \#G_3 > \#G_3 + \#G_4 + \#G_1$ before the move, and hence, $\#G_2 > \#G_4$ after the move. Therefore, the configuration resulting from the move falls under 3.2.2 d).

3.2.2: Suppose there is no robot at o, and assume without loss of generality that $\#G_1 \geq \#G_2, \#G_3, \#G_4$.[4]

a) If $c-1 > \#G_1 = \#G_3 > \#G_2 = \#G_4 > 0$, then a robot, say r_2, uniquely identified in G_2 moves to G_1, and a robot, say r_4, uniquely identified in G_4 moves to G_3. Note that if both r_2 and r_4 observe the current configuration in 3.2.2 a) and, for instance, both complete the respective moves before any other robot executes the Look step, then we arrive at a configuration in 3.2.2 a) again (or 3.2.2 c)) with one more robots than before in both G_1 and G_3. However, due to the delay in the robots' execution cycles, it is possible that only one of r_2 and r_4 observes the current configuration in 3.2.2 a) and completes the required move (yielding a configuration in 3.2.2 b) given below), or both observe it in 3.2.2 a) but one of them, say r_4, starts the move long after the other robot r_2 has finished the move and hence some robots observe a configuration in 3.2.2 b) in which r_4 has already "decided" to move to G_3. For this reason, although in 3.2.2 b) one subgrid contains more robots than any other subgrid and hence it appears as if we could move enough robots to that subgrid and connect P there, we must instead "restore" 3.2.2 a) (or arrive at 3.2.2 c)) by allowing only one robot (in fact, robot r_4 in the above scenario) to move from G_4 to G_3.

[3] Since $\#G_1 + \#G_2 + \#G_3 + \#G_4$ is odd, the condition uniquely identifies a subgrid. See the previous footnote.

[4] Here, G_1 can be any of the subgrids having the largest number of robots.

b) If $c-1 \geq \#G_1 = \#G_3 + 1$, $\#G_3 > \#G_4$, and $\#G_4 = \#G_2 + 1$, then one robot in G_4 moves to G_3. The robot in G_4 that moves to G_3 is the robot (called r_4 above) that moves from G_4 to G_3 in 3.2.2 a). As explained above, this rule ensures that the resulting configuration satisfies $\#G_1 = \#G_3 > \#G_2 = \#G_4$ and falls under 3.2.2 a) or 3.2.2 c), regardless of the delays in the robots' execution cycles.

c) If $\#G_1 = \#G_3 \geq c-1$ and $\#G_2 = \#G_4$, then as in 2.2.2.1, for both $i = 1$ and 3 the robots in G_i cover the essential target points in T_i on the x-axis of Z_i by executing Algorithm 1 without leaving G_i. We remark that when we apply 2.2.2.1 in Case 3, for $i = 1$ and 3, $(2a, 0)$ of Z_i is not an essential target point[5] and hence the robots in G_i need not leave G_i.

d) If none of 3.2.2 a), b) and c) applies, then there exists a unique subgrid G_i that maximizes the sum $\#G_i + \#G_{i+1} + \#G_{i+2}$ among the subgrids having the largest number of robots.[6] The robots move to this G_i as necessary and connect P using Algorithm 1, as in 3.1.2. We point out that once the robots start executing Algrithm 1, none of 3.2.2 a), b) and c) will apply.

The number of robots needed is $\min\{n-1, 2c-2\}$ in 3.2.2 c) (as in 2.2), and $c-1$ in 3.2.2 d) (as in 3.1.2). Thus P is connected by m robots for any $m = 4k+2 \geq \min\{n-1, 2c-2\}$. (End of 3.2.)

3.3: $m = 4k \geq \min\{n-1, 4c-4\}$.

3.3.1: If there is a robot at o, then we proceed as in 3.2.1 (and then as in 3.2.2 d), and eventually as in 3.1.2). This works for any $m = 4k \geq c-1$.

3.3.2: Suppose there is no robot at o.

3.3.2.1 If $\#G_1 = \#G_2 = \#G_3 = \#G_4 = k$, then the robots in each subgrid cover certain "essential" target points in their subgrid as described below.

A component of P is said to be *symmetric* if it "looks the same" in Z_1, Z_2, Z_3 and Z_4, i.e., the set of coordinates of the points in it is the same in Z_1, Z_2, Z_3 and Z_4. Observe that in Case 3 every component C is of one of the following six types. The target point(s) that C contributes is/are designated as *essential* in some cases, as mentioned below. (See Figure 7.) As in Case 3.2.2 c), for $i = 1$, 2, 3 and 4,

[5] In 2.2.2.1, vertex $(2a, 0)$ of Z_1 can be an essential target point only if there is a non-symmetric component C of type 3 that entirely lies on the x-axis of Z_2 and that includes vertex $(0, 0)$ of Z_3. In Case 3, however, there exist four copies of C, all connected via $(0, 0)$ of Z_1, Z_2, Z_3 and Z_4. Thus C does not entirely lie on the x-axis of Z_2. This is a contradiction, and hence such C cannot exist.

[6] Again, we omit the proof of the fact that if $\#G_1 + \#G_2 + \#G_3 + \#G_4 = 4k+2$ and none of 3.2.2 a), b) and c) applies, then the condition uniquely identifies a subgrid.

$(2, a)$ of Z_i never becomes an essential target point in T_i, and thus the robots in G_i do not need to leave G_i.

type 1: C is symmetric, has a point on the x-axis of each of Z_i, $i = 1, 2, 3, 4$, and contributes no target point.

type 2: C is symmetric, has no point on any of the four x-axes, and contributes exactly four target points $t_1 \in T_1$, $t_2 \in T_2$, $t_3 \in T_3$ and $t_4 \in T_4$, all having the same x-coordinate in their respective coordinate systems Z_1, Z_2, Z_3 and Z_4. All four are designated as essential.

type 3: C is not symmetric, has a point on the x-axis of Z_i and the x-axis of Z_{i+2} for some i (but not on the other two x-axes), and contributes exactly two target points, $t_{i+1} = (0, 0) \in T_{i+1}$ and $t_{i+3} = (0, 0) \in T_{i+3}$. In 3.3.2.1 a) below, t_{i+1} and t_{i+3} are not designated as essential, while in 3.3.2.1 b), t_{i+1} and t_{i+3} become essential.

type 4: C is not symmetric, has a point on the x-axis of only one Z_i, and contributes exactly three target points, $t_{i+1} = (0, 0) \in T_{i+1}$, $t_{i+2} \in T_{i+2}$ and $t_{i+3} \in T_{i+3}$. In 3.3.2.1 a) below, none of these points is designated as essential, while in 3.3.2.1 b), t_{i+1} becomes essential.

Observe that a component of type 3 and a component of type 4 cannot exist simultaneously. The total number of additional essential target points contributed by the components of type 3 or type 4 in 3.3.2.1 b) is four.

type 5: C is not symmetric, has no point on any of the four x-axes, and contributes exactly four target points $t_1 \in T_1$, $t_2 \in T_2$, $t_3 \in T_3$ and $t_4 \in T_4$, where t_1 and t_3 have the same x-coordinate (in their respective coordinate systems Z_1 and Z_3), and t_2 and t_4 have the same x-coordinate (in their respective coordinate systems Z_2 and Z_4) that is different from the x-coordinate of t_1 and t_3. The two target points among the four (either t_1 and t_3, or t_2 and t_4) that have the smaller x-coordinates are designated as essential.

type 6: C is not symmetric, has no point on any of the four x-axes, and contributes exactly four target points $t_1 \in T_1$, $t_2 \in T_2$, $t_3 \in T_3$ and $t_4 \in T_4$, all having distinct x-coordinates in their respective coordinate systems. The one among t_1, t_2, t_3 and t_4 having the smallest x-coordinate is designated as essential.

3.3.2.1 a): If there is at least one symmetric component, then covering all essential target points connects all components. This follows from the following two facts (see Figure 7 a)).

- Every component either contains a point or contributes an essential target point on one of the four x-axes.

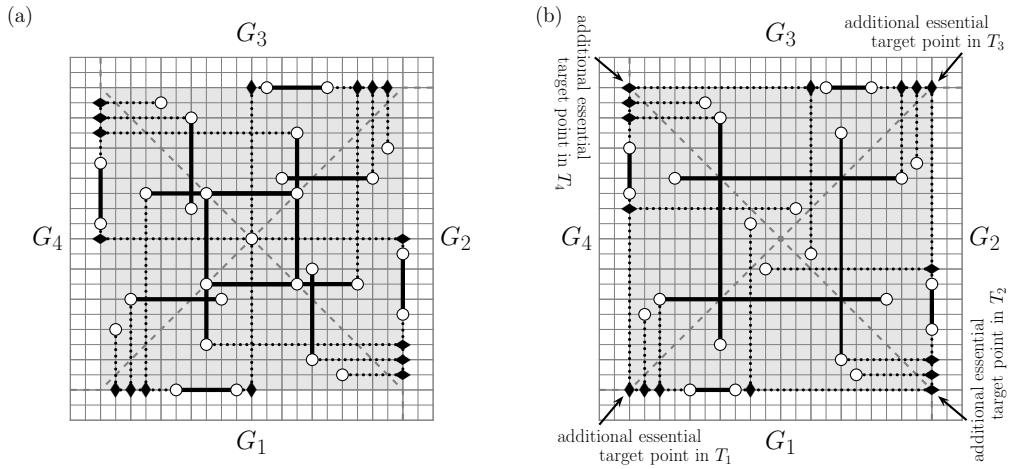

Fig. 7. Essential target points in a) 3.3.2.1 a) and b) 3.3.2.1 b)

- The points on the four x-axes either belong to one symmetric component or are connected through a symmetric component and its four essential target points.

Again, a simple counting argument shows that the total number of essential target points is at most $n-1$: At least four of the n points in P contribute no essential target point because they lie on one of the four x-axes, and any other point 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, one in each T_i, $i = 1, 2, 3, 4$. Also, the total number of essential target points is at most $4c - 4$. Thus the strategy works for any $m = 4k \geq \min\{n-1, 4c-4\}$.

3.3.2.1 b): If no symmetric component exists, then we cover all essential target points, including the four additional ones designated for the type 3 or 4 components above (see Figure 7 b)). Covering the four additional essential target points connects, for each $i = 1, 2, 3, 4$, the component having a point on the x-axis of Z_i and the component having a point on the x-axis of Z_{i+1}. All other components are connected to one of these (four) components via their essential target points. Thus all components are connected.

The total number of essential target points is at most $2c$ because every component contributes at most two essential points, and at most n because each point contributes at most one target point. Thus the strategy works for any $m = 4k \geq \min\{n, 2c\}$.

3.3.2.2 If $\#G_1$, $\#G_2$, $\#G_3$ and $\#G_4$ are not all equal, then we proceed as in 3.2.2.[7] This works for any $m = 4k \geq \min\{n-1, 2c-2\}$.

The number of robots needed is $c-1$ in 3.3.1, $\min\{n-1, 4c-4\}$ in 3.3.2.1 a), $\min\{n, 2c\}$ in 3.3.2.1 b), and $\min\{n-1, 2c-2\}$ in 3.3.2.2. Note that

1. $4c-4 \geq 2c > 2c-2 > c-1$ for $c \geq 2$, and
2. in 3.3.2.1 b) n is a multiple of 4[8] and hence any $m = 4k \geq n-1$ satisfies $m \geq n$.

Thus P can be connected using m robots for any $m = 4k \geq \min\{n-1, 4c-4\}$. (End of 3.3.)

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 $m = 4k \geq \min\{n-1, 4c-4\}$. (End of Case 3.)

Remark 4. Figure 8 shows instances that demonstrate the tightness of the bounds obtained for 3.2, 3.3.2.1 a) and 3.3.2.1 b); notice that in Case 3, $n = 0, 1 \bmod 4$.

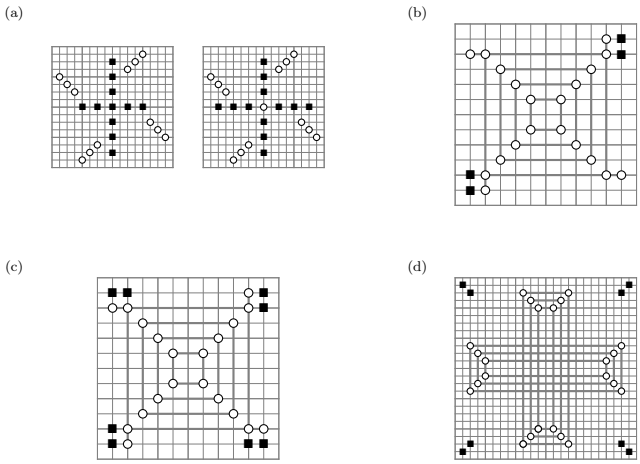

Fig. 8. a) Case 3, in which $c = n$, and so $n-1$ robots are required. b) Case 3.2 in which $2c-4$ robots are not enough to connect P having $c = 4$. c) Case 3.3.2.1(a) in which $4c-8$ robots are not enough to connect P having $c = 4$. d) Case 3.3.2.1(b) in which $2c-4$ robots are not enough to connect P having $c = 6$; notice that c is even, and so $2c-2 = 2 \bmod 4$.

[7] Again, we omit the proof of the fact that if $\#G_1 + \#G_2 + \#G_3 + \#G_4 = 4k$ and none of 3.2.2 a), b) and c) applies, then the condition of 3.2.2 d) uniquely identifies a subgrid.

[8] In Case 3, if n is not a multiple of 4, then $o \in P$, which implies the existence of a symmetric component.

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.

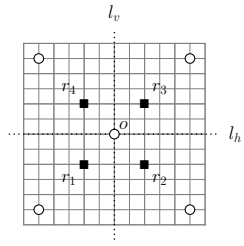

Fig. 9. To connect the two components, a robot must move to either row l_h or column l_v. If the local coordinate systems of r_1 and r_3 are right-handed and those of r_2 and r_4 are left-handed, then it is possible that r_1 and r_3 always move symmetrically with respect to o, r_2 and r_4 always move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

Acknowledgments

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

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. A. Kosowski was partially supported by Polish Ministry Grant N206 491738. 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.

Reasearch partially supported by the Polish National Research Center (DEC-2011/02/A/ST6/00201).

REFERENCES

[1] Abbasi, A. A.—Younis, M.—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.: Coverage and Latency Aware Actor Placement Mechanisms in Wireless Sensor and Actor Networks. 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, A.: Efficient and Complete Coverage of 2D Environments by Connectivity Graphs for Motion Planning Algorithms. Journal of Information Science and Engineering, Vol. 22, 2006, No. 6, pp. 1355–1366.

[21] Lulu, L.—Elnagar, A.: An Art Gallery-Based Approach: Roadmap Construction and Path Planning in Global Environments. International Journal of Robotics and Automation, Vol. 22, 2007, 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. The Journal of Ad Hoc Networks, Vol. 6, 2008, No. 4, 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.