International Journal of Foundations of Computer Science

UNIFORM SCATTERING OF AUTONOMOUS MOBILE ROBOTS IN A GRID

LALI BARRIÈRE
Universitat Politècnica de Catalunya[*]

PAOLA FLOCCHINI
University of Ottawa[†]

EDUARDO MESA-BARRAMEDA
Universidad de la Habana [‡]

NICOLA SANTORO
Carleton University[§]

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

We consider the *uniform scattering* problem for a set of autonomous mobile robots deployed in a grid network: starting from an arbitrary placement in the grid, using purely localized computations, the robots must move so to reach in finite time a state of static equilibrium in which they cover uniformly the grid. The theoretical quest is on determining the minimal capabilities needed by the robots to solve the problem.
We prove that uniform scattering is indeed possible even for very weak robots. The proof is constructive. We present a provably correct protocol for uniform self-deployment in a grid. The protocol is fully localized, collision-free, and it makes minimal assumptions; in particular: (1) it does not require any direct or explicit communication between robots; (2) it makes no assumption on robots synchronization or timing, hence the robots can be fully asynchronous in all their actions; (3) it requires only a limited visibility range; (4) it uses at each robot only a constant size memory, hence computationally the robots can be simple Finite-State Machines; (5) it does not need a global localization system but only orientation in the grid (e.g., a compass); (6) it does not require identifiers, hence the robots can be anonymous and totally identical.

[*]Departament de Matemàtica Aplicada IV, Barcelona, Spain. lali@ma4.upc.edu
[†]SITE, Ottawa, Canada. flocchin@site.uottawa.ca
[‡]Facultad de Matemática y Computación, La Habana, Cuba. eduardomesa@matcom.uh.cu
[§]School of Computer Science, Ottawa, Canada. santoro@scs.carleton.ca

1. Introduction

Consider a set of autonomous computational entities, called *robots* or *agents*, located in a grid network. A robot is capable of limited sensing and computational activities, and it can move in the grid from a node to a neighbouring node. The initial positions of the robots in the grid is arbitrary; the goal is to reach a state of static equilibrium in which they are uniformly scattered in the grid. This problem, called *uniform scattering* (or *covering*, or *self-deployment*), occurs in practice, e.g. when robots are randomly deployed in a region but the requirement is for the region to be covered uniformly, so to maximize coverage. A scattering or self-deployment algorithm, the same for all robots, will specify which operations a robot must perform whenever it is active to achieve this goal. An essential requirement is clearly that the robots will reach a state of *static equilibrium*, that is the scattering will be completed within finite time.

How uniform scattering can be efficiently accomplished in a region is the subject of extensive research in several fields. In cooperative mobile robotics and swarm robotics, this question has been posed under the terms *scattering*, *coverage*, and as a special case of *formation* (e.g. [2, 3, 5, 11, 13, 17, 18]). Similar questions have been posed in terms of *self-deployment* in mobile sensor networks and in networks of robotic sensors (e.g., [7, 8, 9, 10, 15, 16]).

The existing protocols differ greatly from each other depending on the assumptions they make. The first and foremost difference is on the nature of the environment, depending on whether the environment is a subset of two-dimensional plane, a setting usually called *continuous* (e.g., [5, 14, 17, 18]), or it is a network or a graph, a setting usually called *discrete* or *graph world* (e.g., [1, 9, 11]). Other major differences exist depending on whether: the robots' actions are synchronized (e.g., [2, 11, 17, 18]) or no timing assumptions exist (e.g., [4, 5, 13]); the robots have persistent memory (e.g., [2, 7, 18]) or are oblivious (e.g., [5, 10]); the robots have the computational power of Turing machines (e.g., [12, 17, 18]) or are simple Finite State machines (e.g., [1, 11]); the visibility/communication range is limited (e.g., [4, 7, 6, 11]) or extends to the entire region (e.g., [5]); exact or approximate uniform covering is reached within finite time (e.g., [4, 5, 11]) or the protocol converges without ever terminating (e.g., [2, 3]); the protocol is generic, i.e., it operates in any space/network (e.g., [5, 12, 18]) or only in specific for classes of regions/graphs (e.g., [2, 3, 6]); etc.

The robots we consider here are autonomous (i.e., without a central control), anonymous (i.e., identical), asynchronous (i.e., their actions take a finite but unpredictable amount of time), randomly dispersed in the environment, without a global localization system, with limited sensing range (called visibility radius), with no communication capabilities, and constant size memory.

Under these conditions, there is no generic protocol capable of a uniform coverage. This fact is hardly surprising considering both the weakness of the robots and the requirement that a generic protocol must work in any environment regardless of its topology or structure. This fact opens the natural question of whether it is possible for these weak robots to self-deploy achieving uniform coverage in spe-

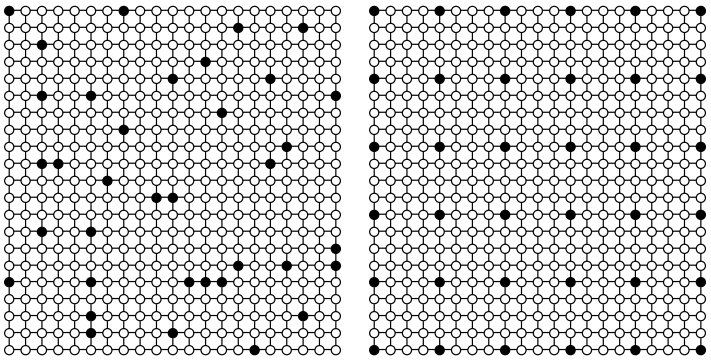

Figure 1: A random initial configuration, and the uniform scattering configuration.

cific environments (e.g., corridors, grids, rims). A solution algorithm has recently been developed for the *line* (e.g., a rectilinear corridor) [2], and several have been designed for the *ring* (e.g., the border of a convex region) [3, 4, 13].

We are interested in the uniform scattering of these weak robots in a *grid*: starting from an initial random placement on the grid, the robots must position themselves in the grid at equal distance, within finite time (see Figure 1). The existing investigations on uniform covering of a grid by weak robots [1, 11] assume that the robots enter the grid one at a time from fixed locations. However, nothing is known, to date, on whether or not the problem is solvable when the robots are, initially, arbitrarily dispersed in the grid.

In this paper we prove that indeed uniform coverage by very weak robots is possible in a square grid starting from any arbitrary initial configuration and without any collisions. The proof is constructive: we present a collision-free protocol for uniform scattering in a grid and prove its correctness. The protocol is fully localized and decentralized, and it makes minimal assumptions; in particular: (1) it does not require any direct or explicit communication between robots; (2) it makes no assumption on robots synchronization or timing, hence the robots can be fully asynchronous in all their actions; (3) it requires only a limited visibility range; (4) it uses at each robot only a constant persistent memory, hence computationally the robots can be simple Finite-State Machines; (5) it does not need a global localization system but only orientation in the grid (e.g., a compass); (6) it does not require identifiers, hence the robots can be anonymous and totally identical.

2. Model

The system is composed of a set $\mathcal{S}$ of $K = (k+1)^2$ identical computational entities, called robots, dispersed in a a square grid $\mathcal{G}$ of $N = (n+1) \times (n+1)$ nodes, where $n = k \cdot d$, $d \geq 2$, and $k \geq 2$.

The robots in $\mathcal{S}$ are autonomous, asynchronous, anonymous computational units with sensory and locomotion capabilities. Each robot functions according to an algorithm (the same for all robots) preprogrammed into it, and can move from a

node to any neighbouring node in the grid. They have a local sense of orientation (i.e., each robot has a consistent notion of "up-down" and "left-right", e.g., as provided by a compass) but no access to any global localization system, so in general they do not know their own position in the grid. The common orientation will be modeled by assuming that the edges of the grid are consistently labeled *Up*, *Down*, *Left* and *Right*, and edge labels are visible to the robots. We will indicate by $(0,0)$ the left-most lower corner of $\mathcal{G}$ and by (i,j) the node belonging to column i and row j.

The sensory devices on the robot allow it to have a vision of its immediate surrounding; we assume the robots to have restricted vision of radius $2d$ around it; i.e., a robot sees all nodes at distance (in the Manhattan or city block metric) at most $2d$ from it and the links between them. Even if two robots see each-other, they do not have any explicit means of communicating with each-other.

Each robot performs the following sequence of operations: *Look*: the robot obtains a snapshot of its surroundings within its visibility radius $2d$, to see which nodes are empty and which are occupied and on which links robots are moving; *Compute*: based on the obtained snapshot and the rules of the algorithm, the robot chooses as its next destination either one of the neighboring nodes or the node where it resides; *Move*: the robot moves to the computed destination node. The sequence *Look-Compute-Move* constitutes the unit *cycle*, and each robot reiterates this cycle endlessly.

The robots are fully *asynchronous*; that is, between successive operations, the robot can be idle for a finite but unpredictable amount of time. While *Look* is by definition instantaneous, *Compute* and *Move* may take an unpredictable (but finite) amount of time. Nevertheless, without any loss of generality, it is possible to assume that also these two operation are instantaneous. This is trivially true in the case of *Compute* (e.g., by attributing its duration to idle time *after* the operation); in the case of *Move*, every protocol can be easily modified by adding rules to ensure atomicity of the operation (e.g., the current cycle will be ended if the snapshot shows some robots on the edges). Hence, in the following, we will assume that all operations are instantaneous.

The *uniform scattering* (or self-deployment) problem consists of starting from a random initial placement of the robots in $\mathcal{G}$ and to reach an equilibrium configuration where the robots cover the grid evenly. Since $n = kd$ such an equilibrium configuration consists of nodes $(i \cdot d, j \cdot d)$, with $i, j \in [0, k]$ hosting exactly one robot each (see Fig 1).

In the description of the algorithm for robot s we use the following notation: $\ell_t(s)$ denotes the location of s at time t (i.e., it is a pair $(i,j) \in [0,n]^2$), and $(x,y)_t^+$ denotes coordinates relative to s at time t (i.e. $(x,y)_t^+ = \ell_t(s) + (x,y)$). The algorithm will use only *relative coordinates* because the robots are not aware of their global coordinates; a robot, in fact, can only understand whether it is located inside the grid, on a corner or on a border. Note that, due to orientation it can also distinguish the four borders and corners, and due to visibility it can "see" a border or a corner if at distance smaller than $2d-1$ from it: in particular, if it sees

a border it can also understand its global coordinates.

The *visibility set* of s at time t is the set of the nodes of the grid that are visible by s at time t; that is $V_t(s) = \{\ell_t(s) + (x, y) : |x| + |y| \leq 2d\} \cap [0, n] \times [0, n]$.

Let $V_t^*(s) = \{\ell_t(s) + (x, y) : |x| + |y| \leq 2d,\ x > -2d\} \cap [1, n] \times [0, n]$ indicate the visibility set $V_t(s)$ of s excluding any node that is, or might be, on the left border of the grid (we will need such a set for a phase of the algorithm). Notice that if robot s does not "see" the border, the node $\ell(s) + (-2d, 0)$, which it perceives as $(-2d, 0)^+$, might belong to the border, so the need to exclude it as well.

In the following, when no ambiguity arises, we will omit the subscript t.

3. Uniform Scattering Protocol

The uniform scattering algorithm SCATTER is a set of local rules for the robots. Each robot has a *state* variable (initialized to -1) which determines the set of rules to be followed, which solely depend on its current state, its position, and the positions of the robots within its visibility radius. Recall that if $0 \leq \delta < 2d$, a robot is able to determine whether it is at a distance δ from any of the borders of the grid.

Upon startup, the execution of the algorithm is logically divided in three *phases*.

- *Cleaning:* robots (if any) on the left and bottom borders move leaving those nodes empty (it is performed only by robots in state 0).
- *Collecting:* robots move towards the left-upper corner of the grid (it is performed by robots in states 1 and 2).
- *Deploying:* robots follow a distinguished path on the grid, called *snake-path*, eventually occupying their final positions (it is performed by robots in states 3 and 4. We refer to the first robot to enter this phase the *head* of the snake; however, such a robot does not need to be aware of being the head.

Waking up for the first time (in state -1), each robot executes the following set of rules:

ROBOT s IN STATE -1 (waking up for the first time):
LOOK $V(s)^*$

- If s is at *left or bottom border*, but not on *upper corner* (i.e., $\ell(s) = (0, j)$, with $j < n$, or at $\ell(s) = (i, 0)$, with $i > 0$): s enters state 0 (i.e., it performs *Cleaning*).
- If s is *not in the left nor in the bottom border* (i.e., $\ell(s) = (i, j)$, with $i, j \geq 1$): s enters state 1 (i.e., it performs *Collecting*).
- If s is at *left upper corner*: (i.e., $\ell(s) = (0, n)$): s enters state 3 (i.e., it starts *Deploying*).

Note that, due to asynchrony, at any point in time robots could be performing actions belonging to different phases. In particular, all robots that wake up for the first time on the left or bottom borders, enter the *Cleaning* phase; robots that are already inside the grid the first time they wake up enter the *Collecting* phase; if a

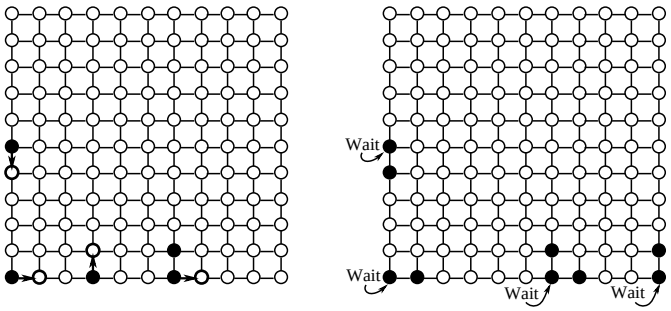

Figure 2: Rules of the *Cleaning* phase.

robot starts the algorithm in the upper left corner, it enters directly the *Deploying* phase. The asynchronous execution of the various phases requires special care in order to insure that robots continue to progress in their phase avoiding collisions and the creation of deadlocks.

3.1. Algorithm SCATTER*: Cleaning*

The *Cleaning* phase is performed only by robots in state 0. The goal is to have the robots move from the left and bottom borders. If moving from the bottom, a robot enters state 1 and starts the *Collecting* phase. The rules are described below (see also Figure 2).

ROBOT s IN STATE 0:
LOOK $V(s)^*$

- If s is at left border but not on the corners (i.e., $\ell(s) = (0, j)$, with $0 < j < n$):
  - If its lower node $((0,-1)^+)$ is empty: s goes down.
  - Otherwise, s waits.
- If s is at left bottom corner (i.e., $\ell(s) = (0, 0)$).
  - If its right node $((1,0)^+)$ is empty: s goes right.
  - Otherwise, s waits.
- If s is at bottom border but not on the left corner (i.e., $\ell(s) = (j, 0)$, with $0 < j \leq n$).
  - If the upper node $((0,1)^+)$ is empty, s goes up, enters state 1 (i.e., starts *Collecting*).
  - If the upper node $((0,1)^+)$ is occupied and the right node $((1,0)^+)$ is empty, s goes right .
  - Otherwise, s waits.

3.2. Algorithm SCATTER*: Collecting*

The *Collecting* phase is performed by robots in states 1 or 2. The goal of the robots is to arrive to the left-upper corner, i.e. node $(0, n)$. To avoid conflicts with robots possibly already in the *Deploying* phase, during the *Collecting* phase a robot should not consider robots that are on the left border; i.e., it limits its observation to $V^*(s)$. With this aim, robots act according to the following rules[a] (see also Figure 3).

ROBOT s IN STATE 1:
LOOK $V^*(s)$
- If s is on the right neighbor of the upper left corner (i.e. $\ell(s) = (1, n)$):

  - If the upper left corner ($(0, n)$) is empty, s goes left and enters state 3 (i.e., starts the *Deploying* phase).
  - Otherwise, s waits and stays in state 1.

- Otherwise (i.e., $\ell(s) = (i, j)$, $i, j > 0$; $\ell(s) \neq (1, n)$):

  - If the upper node ($(0, 1)^+$) is empty, s goes up and stays in state 1.
  - If all the above nodes in its same column (i.e., $(0, y)^+$, with $1 \leq y \leq 2d$), are occupied and the left and bottom-left nodes ($(-1, 0)^+$ and $(-1, -1)^+$) are empty, s goes left and stays in state 1.[a].
  - If within $V(s)^*$ all the following conditions are satisfied, then s goes right and enters state 2.
    - All the nodes not below nor right (i.e., $(x, y)^+$, with $x \leq 0$ and $y \geq 0$), are occupied.
    - If the lower right node (i.e., $(1, -1)^+$) is empty.
    - All the nodes in its same row on the right (i.e., $(x, 0)^+$, with $x > 0$), are empty.
    - At least one of the the above and right nodes (i.e., $(x, y)^+$, with $x > 0$ and $y > 0$), is empty.
    - None of the robots in the above and right nodes at distance strictly less than $2d$ (i.e., $(x, y)^+$, with $x > 0$, $y > 0$ and $x + y < 2d$), can go up.
  - Otherwise, s waits and stay in state 1.

[a]Note that a robot in state 1 never moves to the left border unless it is at node $(1, n)$

As we will show in Section 4, this set of rules allows the robots to be located in a monotone portion of the grid. These rules are shown in Figure 3.

[a]Note that if s is in state 1 then it can not be on the left nor bottom border

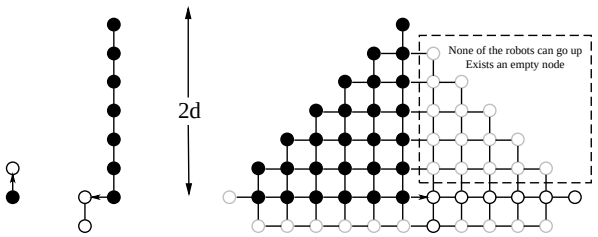

Figure 3: Rules for state 1 of the *Collecting* phase.

ROBOT s IN STATE 2:
LOOK $V^*(s)$
- In all situations[a] (i.e., $\ell(s) = (i, j)$, $i, j > 0$):

- If the upper node ($(0, 1)^+$) is empty, s goes up and enters again state 1.
- If within $V(s)^*$ all the following conditions are satisfied, then s goes right (staying in state 2).
  - All the nodes above and the ones not to the right (i.e., $(x, y)^+$, with $x \leq 0$ and $y > 0$), are occupied.
  - The lower right node (i.e., $(1, -1)^+$) is empty.
  - All the nodes in the same row on the right (i.e., $(x, 0)^+$, with $x > 0$), are empty.
  - At least one among the nodes above and the ones to the right (i.e., $(x, y)^+$, with $x > 0$ and $y > 0$), is empty.
  - None of the robots in the above and right nodes at distance strictly less than $2d$ (i.e., $(x, y)^+$, with $x > 0$, $y > 0$ and $x + y < 2d$), can go up.
- Otherwise, s enters state 1.

[a] Notice that if s is in state 2 then it can not be on the left nor bottom border.

Notice that state 2 is dedicated to robots going right, which is a non-prioritary movement. As we will see, these movements allow the set of robots to adopt a shape in the grid that facilitates avoiding deadlocks. These rules are shown in Figure 4.

3.3. Algorithm SCATTER*: Deploying*

This phase is executed by robots in state 3 and 4. A robot starts this phase when it arrives to node $(0, n)$ and enters state 3. In this phase the robots move on the snake-path (see Figure 5) and eventually stop in their final positions, that is the nodes $(i \cdot d, j \cdot d)$, with $0 \leq i, j \leq k$, called *final nodes*.

Let k be odd; the snake-path is the path $n_0, n_1, \ldots n_{(K-1)d}$, that starts at $n_0 = (0, n)$, ends at $n_{(K-1)d} = (d, n)$ and passes through every final node as shown in Figure 5. We will denote the nodes in the snake-path as $n_{i \cdot d}$, for the i-th final node, and $n_{i \cdot d + j}$, $1 \leq j < d$, for the j-th node in the path between $n_{i \cdot d}$ and $n_{(i+1)d}$. By a

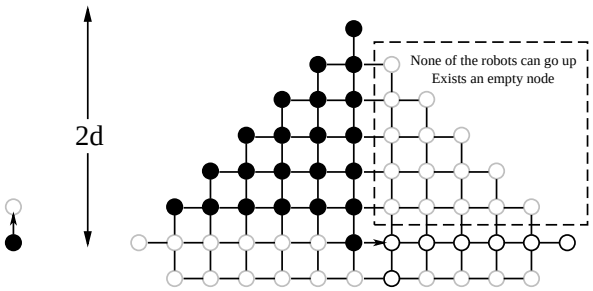

Figure 4: Rules for state 2 of the *Collecting* phase.

slight modification of this path and, consequently, of the algorithm, the snake-path can be defined for k even.

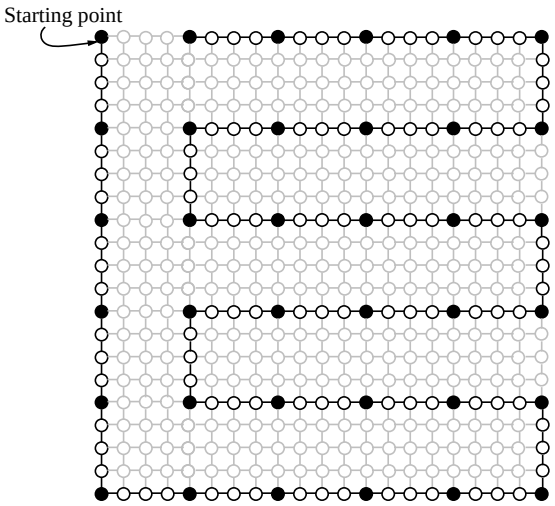

Figure 5: The snake-path for $N = 21^2$, $K = 36$,$k = 5$ $d = 4$.

Because of asynchrony, complications may arise. For example, if a robot in the *Deploying* phase enters in contact with robots that are still in the *Collecting* phase, the robot has to wait before continuing on the snake-path. Special care has to be taken to avoid deadlocks and guarantee progress.

Rules for state 3. A robot s enters state 3 when it reaches the left-upper corner, i.e. node $(0, n)$. In this state, s follows the left border moving only if it sees above another robot at distance lower than d and if the lower node is empty. If the robot is in the left upper corner it only moves if the right node is occupied, insuring in this way that there is at least one node in the *Collecting* phase. When the last node enters the *Deploying* phase it will stay at the upper left corner and eventually the other nodes will position themselves at distance d from each other. When in state

3, a robot s is following the left border of the grid, i.e., its location can be indicated by $\ell(s) = n_i$, for $0 \leq i < kd$. It enters state 4 when it reaches the bottom-left corner. See Figure 6 for an illustration of the rules.

ROBOT s IN STATE 3:
LOOK $V(s)$

- If s is at upper left corner (i.e., $\ell(s) = n_0 = (0, n)$):
  - If the right node $((1, n))$ is occupied and the lower node is empty then s moves down.
  - Otherwise s waits.
- If s is at left border but not on the corners (i.e., $\ell(s) = n_j$, for $0 < j < kd$):
  - If the next node on the snake-path (i.e., n_{j+1}) is occupied then s waits.
  - If all the $d - 1$ prior nodes on the snake-path (i.e., nodes from $n_{j-(d-1)}$ to n_{j-1})[a] are empty, then s waits.
  - Otherwise, s goes down. If s reaches node $(0, 0)$, it enters state 4.

[a] If $j < (d - 1)$ robot s only sees the path from n_0 to n_{j-1}

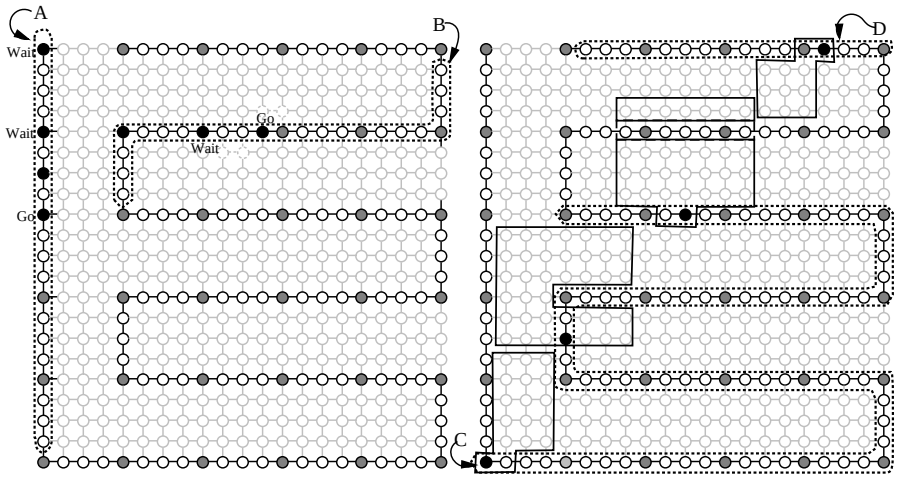

Figure 6: Rules for states 3 and 4. Gray nodes are final nodes; black nodes are hosting a robot. Robots in A are in state 3; robots in B are in state 4 and only look at the snake-path to decide whether to go on or wait; robots located in C and D are in state 4 and can continue only if they are alone in the marked areas.

Rules for state 4. A robot s in this state is following the snake-path, from the left-bottom corner. That is, its location is $\ell(s) = n_i$, for $kd \leq i \leq (K - 1)d$.

A robot s in the *Deploying* phase might see in the rows above some robots still performing the *Collecting* phase (the presence of these robots can be detected because of their "wrong" positions). In this case s waits. If s does not see any robot

out of the snake-path but there are no robots in the $d-1$ preceding nodes or the next node is occupied, then it waits. Otherwise s moves forward on the snake-path. When the last node enters the *Deploying* phase it will stay at the upper left corner and eventually all other nodes will stop at their final position.

ROBOT s IN STATE 4: LOOK $V(s)$

- In all situations (i.e., $\ell(s) = n_j$, for $kd \leq j \leq (K-1)d$):
  - If all the $d-1$ prior nodes in the snake-path (i.e., from node $n_{j-(d-1)}$ to n_{j-1}) are empty, then s waits.
  - If the next node in the snake-path (n_{j+1}) is occupied then s waits.
- If s is not on the two last row of the snake-path(i.e., $\ell(s) = n_j$, for $kd \leq j \leq (K-2k-1)d$):
  - If s detects a robot z not in the snake-path, in the next $d+1$ rows above and at most $d-1$ columns farther, then s waits (for example, robots in C in Figure 6 apply this rule).
- If s is in one of the last $kd-d+1$ nodes of the snake-path (i.e., $\ell(s) = n_j$, for $(K-k)d \leq j < (K-1)d$):
  - If s detects a robot z at left in the $d-1$ rows below (i.e, at least one of the nodes $(-i,-j)^+$, with $1 \leq i, j \leq d-1$ is occupied), then s waits (for example, robots in D in Figure 6 apply this rule).
- Otherwise, s goes forward on the snake-path.

4. Correctness

In this Section we show the correctness of algorithm SCATTER: we prove that both collisions between robots and deadlocks are avoided, and also that any execution of the algorithm eventually ends in a uniform scattering configuration.

First we start with two observations.

Note that, according to the rules for state 4, a robot s in state 4, which is not on the last two rows of the snake-path, never moves if it detects a robot z, not in the snake-path, in the $d+1$ rows above it. Thus:

Observation 4.1 *If s is in state 4, and it is not on the $d-1$ last rows of the snake-path, then every robot below s is in the* Deploying *phase.*

Note also that, if s is in one of the last $kd-d+1$ nodes of the snake-path then, according to the rules for state 4, s will not move if it detects a robot z at left in the $d-1$ rows below it. Thus, with this fact and observation 4.1 we have that:

Observation 4.2 *If a robot s in the* Collecting *phase is blocked by a robot in the* Deploying *phase, then it is at node* $(1,0)$.

Using the above observations we can derive the following theorem.

Theorem 1 *During the execution of algorithm* SCATTER*, no two robots can collide.*

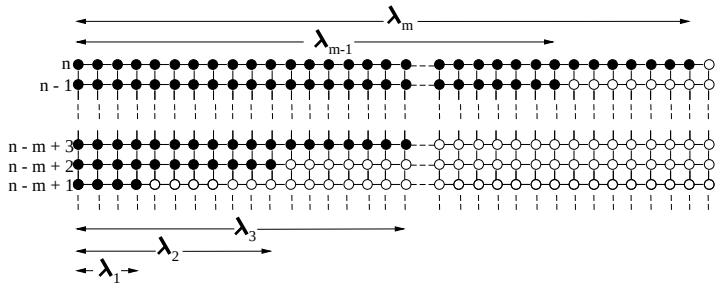

Figure 7: A m-stable configuration, with no complete rows.

We now define, for a given set of robots, a particular configuration in the grid which we call a *stable configuration*. This definition will be used later to prove that algorithm SCATTER cannot stop because of deadlock.

Definition 1 (Stable configuration) *Let $S \subseteq \mathcal{S}$ be a set of robots executing algorithm* SCATTER. *We say that the set S is in an m-*stable configuration *with t complete rows (possibly $t = 0$) if:*

- *all robots of S are located in the upper m rows;*
- *all robots located in the upper m rows are in S;*
- *for every $j = 1, \ldots, m - t$, the robots of S located in row $n - m + j$ occupy consecutive nodes $(1, n - m + j), \ldots, (\lambda_j, n - m + j)$;*
- *the $m-t$ positive integers $\lambda_1, \ldots, \lambda_{m-t}$ satisfy $\lambda_1 \geq 1$ and, $kd > \lambda_j \geq \lambda_{j-1}+2d-1$, for every $j = 2, \ldots, m - t$;*
- *$\lambda_j = kd$, for every $j = m - t + 1, \ldots, m$.*

We say that S is in an stable configuration *if it is a m-stable configuration, for some m. (See Figure 7.)*

Notice that, for a set of robots S to be in a stable configuration, we ask the robots to be located at the upper rows of the grid, occupying consecutive nodes and every row hosting at least $2d - 1$ nodes more than the previous one. In fact, this last condition cannot be fulfilled if the row is complete. This is reflected in the last t rows, which host kd robots each (i.e., every node is occupied).

The correctness of the algorithm follows from a sequence of lemmas.

In the following Lemma it is shown that, if no robot in the *Collecting* phase enters in the *Deploying* phase for a sufficient amount of time then they will eventually reach a stable configuration.

Lemma 1 *Let S be the set of robots in the* Collecting *phase. Then eventually they will reach a stable configuration, provided that no robot in S enters the* Deploying *phase.*

Proof. Assume that for a large enough number of steps, robots in S cannot enter the *Deploying* phase. Let h in row $n-m-1$ be the head of the snake. By observation 4.1 we know that every robot in rows below $n - m - 1$ is in the *Deploying* phase. Moreover, we know by observation 4.2 that no robot in the *Collecting* phase could

be blocked by a robot in the *Deploying* phase which is not at node $(0,0)$. Thus, eventually the set of robots S will be isolated in the last m upper rows.

By the rules for states 1 and 2, if a robot s decides to go right and enter state 2, either it will go up in a few steps or a faster robot will come from below, occupying an empty node in the same row as s or in a row above. After that, s will go back to state 1. Since robots never go down, after a finite number of steps in the execution of algorithm SCATTER, no robot enters state 2 again.

We can therefore assume that we have reached a situation in which none of the robots enter state 2 again. By an analogous reasoning, if all robots can only go left or up, then, after some steps, they all will be blocked.

We now show that, if none of the robots can move, then they are in a stable configuration.

If none of the robots can go up, nor left, the robots in S are located in the upper m' rows, for some $m' < m$. In every row, they occupy consecutive nodes. Moreover, if λ_j is the number of occupied nodes in row $n-m'+j$, then $\lambda_1 \geq 1$, and $\lambda_j \geq \lambda_{j-1}$ for $j = 2, \ldots, m'$.

Let us concentrate now in robot s_j, with $j < m'$, located at node $\ell(s_j) = (\lambda_j, n - m' + j)$. We know that s_j cannot enter state 2 and go right. We can distinguish the following three cases:

- if $\lambda_j = kd$, then $\lambda_i = kd$, for $j \leq i \leq m'$;
- if $\lambda_j \leq kd - (2d-1)$, then all $2d-1$ nodes $(\lambda_j + 1, n - m' + j + 1)$, $(\lambda_j + 2, n - m' + j + 1)$, ..., $(\lambda_j + 2d - 1, n - m' + j + 1)$ are occupied (otherwise, s could go right, by rules for states 1 and 2), and $\lambda_{j+1} \geq \lambda_j + 2d - 1$;
- if $\lambda_j > kd - (2d-1)$, then all $kd - j$ nodes $(\lambda_j + 1, n - m' + j + 1)$, $(\lambda_j + 2, n - m' + j + 1), \ldots (kd, n - m' + j + 1)$ are occupied (otherwise, s could go right, by rules for states 1 and 2), and $\lambda_{j+1} = kd$.

So, S satisfies the conditions of Definition 1, and thus it is an m'-stable configuration. This completes the proof. □

In the next Lemma we give a lower bound on the number of robots in a stable configuration. Notice that the condition on m does not necessarily imply that there are no complete rows, since we are computing a lower bound.

Lemma 2 *Let S be a set of robots in an m-stable configuration. If $1+(2d-1)(m-1) \leq kd$, then*

$$|S| \geq m + \frac{2d-1}{2}(m^2 - m)$$

Proof. The number of robots in the m-stable configuration is, by definition, $|S| = \lambda_1 + \lambda_2 + \cdots + \lambda_m$ with $\lambda_1 \geq 1$, $\lambda_j \geq \lambda_{j-1} + 2d - 1$, for $j = 2, \ldots, m - t$. If $1 + (2d-1)(m-1) \leq kd$, then $\lambda_j \geq 1 + (j-1)(2d-1)$, for every $j = 1, \ldots, m$. Therefore $|S| \geq \sum_{j=1}^{m} 1+(j-1)(2d-1) = m+(2d-1)\sum_{i=1}^{m-1} i = m+\frac{2d-1}{2}(m^2-m)$. □

To prove that there is no deadlock, we first introduce some lemmas.

Lemma 3 *No robot can be blocked in the* Cleaning *phase.*

Proof. Notice that robots in state 0 (i.e., in the *Cleaning* phase) could only wait for robots in state 1 located at some nodes $(i, 1)$, i.e., in row 1. Moreover, robots in state 1 are waiting for robots in state 0 only if there are robots at some nodes $(j, 0)$ (i.e., in row 0). By construction however, robots in row 1 will eventually move up, leaving room for robots in state 0 to go up as well and enter state 1. □

Lemma 4 *Let s be the first robot, on the snake-path from the node n_0, which is in the* Deploying *phase and blocked by a robot in the* Collecting *phase. Let s be located in node $n_i = (x, y)$. Eventually one of the two conditions will occur: 1) all the nodes on the snake-path in rows below s (i.e., nodes $n_j = (x', y'); y' < y$) are occupied by robots in the* Deploying *phase or 2) all the robots are in the* Deploying *phase.*

Proof. Suppose that there is a case when at least one node in the *Deploying* phase is blocked by a robot in the *Collecting* phase. By contradiction suppose that some nodes behind it, in the snake-path and in a lower row, remains empty. It is possible that for some steps none of the robots in the *Collecting* phase enter the *Deploying* phase; however, we know by lemma 1, that they eventually reach a stable configuration. Let s, in node n_i, be the first robot on the snake-path from node n_0, which is blocked by a robot in the *Collecting* phase. Let $n_j; 0 \leq j < i$ be the first empty node on the snake-path in a lower row. If $j = 0$ then the robot in node $(1, n)$ could move entering the *Deploying* phase thus reaching a contradiction.

On the other hand, if $j > 0$, we can assure that eventually there will be at least one robot in the path $\Gamma := \{n_0, n_1, \ldots n_{j-1}\}$, in the *Deploying* phase. Let s' be the robot in path Γ which is closest (in Γ) to the empty node n_j. We next show that robot s' will eventually arrive to n_j. We can distinguish two cases:

- If $1 \leq j \leq kd$, then robot s' is in state 3. Rules for state 3 allow s to go down and to arrive to node n_j.
- If $j > kd$, then robot s' is in state 4.

Notice that robots in the *Collecting* phase never move down. Moreover, robot s has already passed through nodes $n_0, n_1, \ldots n_{j-1}, n_j$. This allows us to assure that no robot in the *Collecting* phase is neither entering Γ, nor in the $d + 1$ rows above n_j. Applying the rules for state 4, s' is able to follow the snake-path and to arrive to node n_j. In both cases, robot s' could move to n_j leaving a new empty node in the same row or in the row below. So, the robots in Γ will move successively until n_0 is left empty. At this moment, a robot in the *Collecting* phase will move to n_0 entering the *Deploying* phase thus reaching a contradiction. □

As a consequence of the previous lemmas we have that:

Lemma 5 *If there is no robot in the* Deploying *phase blocked by a robot in the* Collecting *phase nor robot in the* Collecting *phase blocked by a robot in the* Deploying *phase then Deadlock is not possible.*

We are ready to prove that algorithm SCATTER never stops in a deadlock situation.

Theorem 2 *No deadlocks will occur during any execution of algorithm* SCATTER.

Proof. By Lemma 3, after some steps in the execution of algorithm SCATTER, all robots are in the *Collecting* phase or in the *Deploying* phase. Assume by contradiction that there is a deadlock; then, by Lemma 5, we know that there is at least one robot in the *Deploying* phase blocked by a robot in the *Collecting* phase. Let s be the first robot, in the *Deploying* phase, in the snake-path blocked by a robot in the *Collecting* phase.

By Lemma 4, after some steps all the nodes in the snake-path in a row lower than s are occupied by robots in the *Deploying* phase. Robot t in node n_0 will wait, provided that s is waiting. Therefore, by Lemma 1, after some steps, the set S of robots in the *Collecting* phase will be in a m''-stable configuration, for some m'' (see gray robots in Figure 8).

Let $m'' = m + m'$ such that m satisfies $1 + (2d-1)(m-1) \leq kd < 1 + (2d-1)m$. By definition 1, we know that the upper m' rows contain kd robots each. Note that robots in the *Deploying* phase can only occupy nodes on the snake path. This means that, for a deadlock to occur when robots are in the *Collecting* phase we would need a higher number of robots if they are in an m''-stable configuration than if they are in an m-stable configuration. Thus we can assume that the robots in the *Collecting* phase are in an m-stable configuration with $1 + (2d-1)(m-1) \leq kd$. This will allow us to apply Lemma 2 for a lower bound on the number of robots in the stable configuration.

We now show that there is a contradiction, based on the fact that there are always at least $d+1$ empty rows between robots in the *Collecting* phase (the gray robots in Figure 8) and robot s. First of all we know that:

- There are $(k+1)d$ robots in the *Deploying* phase in the d left-most columns. In fact, the snake path contains $kd+1$ nodes on the left border and only one node, the bottom-most, on each of the next $d-1$ columns, and by Lemma 4 all these nodes have been occupied.
- By Lemma 2, the number of robots $S(m)$ in the stable configuration satisfies $S(m) \geq m + \frac{2d-1}{2}(m^2 - m)$.
- The number of rows below s in the *Deploying* phase is $kd - m - d$. By Lemma 4, the number of robots in the *Deploying* phase is thus $S(d) \geq (k+1)d + \left\lceil \frac{kd-m-d}{d} \right\rceil kd$.

The total number of robots used is $S(t) = S(m) + S(d)$. To obtain a contradiction we have to prove that the number of used robots exceeds the number of robots in the space, that is, $S(t) \geq (k+1)^2$. The lower bound on $S(m)$ implies that:

$$\begin{aligned} S(t) &\geq m + \frac{2d-1}{2}(m^2 - m) + (k+1)d + \left\lceil \frac{kd-m-d}{d} \right\rceil kd \\ &\geq m + \frac{2d-1}{2}(m^2 - m) + (k+1)d + k(kd-m-d) \\ &= (k+1)d + k^2 d - kd + \left(1 - \frac{2d-1}{2} - k\right) m + \frac{2d-1}{2} m^2 \end{aligned}$$

$$= \quad k^2 d + d + \frac{-2d-2k+3}{2} m + \frac{2d-1}{2} m^2$$

Let us subtract $(k+1)^2$ to this amount and let us denote the result by $f(m)$:

$$f(m) = (k^2+1)(d-1) - 2k + \frac{-2d-2k+3}{2} m + \frac{2d-1}{2} m^2$$

We will show that $f(m) > 0$.

First, we show that $f(m)$ is increasing for $m > m^* = \frac{2d+2k-3}{2(2d-1)}$. In fact $f'(m) = (2d-1)m + \frac{-2d-2k+3}{2}$. Now, $f'(m^*) = 0$ if and only if $m^* = \frac{2d+2k-3}{2(2d-1)}$. Since $f''(m) = (2d-1) > 0$ we have that for $m > m^*$ the function $f(m)$ is strictly increasing.

We know that $m > d(\frac{k}{2} - 1)$, otherwise the number of rows below s would be greater or equal than $\frac{k}{2}d$. The reason is that if we had $s \geq \frac{k}{2}d$, the snake path could room the $(k+1)^2$ robots behind s because $d \geq 2$, which is not possible if there is a deadlock.

Notice also that $(\frac{k}{2} - 1)d - \frac{2d+2k-3}{2(2d-1)} = \frac{d^2(2k-4) - k(d+2) + 3}{2(2d-1)} > 0$ if $k \geq 4$. This is true because $d^2 \geq (d+2)$ for $d \geq 2$; and $(2k-4) \geq k$ for $k \geq 4$. Furthermore, substituting $k = 3$ we obtain $(d-3)(2d+3) + 6 > 0$ which is true for $d \geq 3$. Therefore, $f(m') \geq f((\frac{k}{2} - 1)d)$ for any $m' \geq d(\frac{k}{2} - 1)$ if $k \geq 4$, or $k = 3$ and $d \geq 3$.

Now we show that $f((\frac{k}{2} - 1)d) > 0$.

$$f((\frac{k}{2} - 1)d)$$

$$= (k^2+1)(d-1) - 2k + (\frac{-2d-2k+3}{2})(\frac{(k-2)d}{2}) + (\frac{2d-1}{2})(\frac{(k-2)d}{2})^2$$

$$= \frac{k^2(2d^3 - d^2 + 4d - 8) - 2k(4d^3 - 7d + 8) + 8d^3 + 4d^2 - 4d - 8}{8}$$

Let us denote the numerator as the function $g(k) = k^2 a - 2kb + c$, where $a = 2d^3 - d^2 + 4d - 8 = ((d-2)(2d^2+3d+10)+12)$, $b = 4d^3 - 7d + 8 = ((d-2)(4d^2+8d+9)+26)$ and $c = 8d^3 + 4d^2 - 4d - 8 = 8(d^3 - 1) + 4d(d-1)$. Notice that, since $d \geq 2$, the three terms a, b and c are positive. Thus, $g(k)$ is a parabola opening up and its vertex is in $k = \frac{b}{a}$.

$$g(\frac{b}{a}) = \frac{ac - b^2}{a} = \frac{76d^4 - 124d^3 - 89d^2 + 112d}{a}$$
$$= \frac{d((d-2)((d-1)(76d+104)+71)+46)}{a}$$

Since $d \geq 2$, then $g(\frac{b}{a}) > 0$ and thus $f((\frac{k}{2} - 1)d) > 0$.

This implies that, when *i)* $k = 3$ and $d \geq 3$, or *ii)* $k \geq 4$, for deadlock to exist there must be more than $(k+1)^2$ robots. Since there are only $(k+1)^2$ robots a deadlock is not possible in these cases. Finally, it is easy to verify (by simple but tedious calculations) that deadlock cannot occur in the special cases of *i)* $k = 3$ and $d = 2$, *ii)* $k = 2$.

Therefore, no deadlocks can occur during any execution of algorithm SCATTER.
□

Hence, by Theorems 1 and 2 the correctness of the possibility result follows.

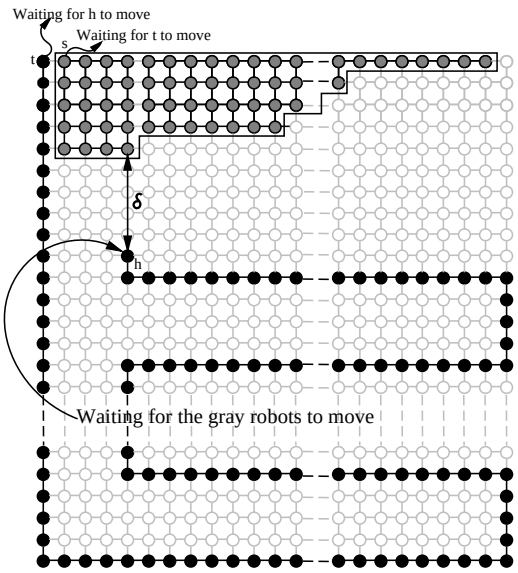

Figure 8: There is deadlock, if $\delta \leq d+1$.

Theorem 3 *Any execution of algorithm* SCATTER *terminates within finite time with a uniform scattering.*

Proof. By theorem 2, deadlock is avoided. Thus, eventually every robot will enter state 3. When the last robot s_0 enters state 3 reaching node $n_0 = (0, n)$ by the rules for state 3 it will remain at node $(0, n)$. The first robot s_1 in front of s will move until be at distance d from s_0. At that moment s_1 will be at node n_d. In this way the robot n_i will stop at distance d from $n_{(i-1)}$ its successor reaching a final node. This process ends when the head reaches the node $n_{(K-1)d}$ and every robot is in a final node. □

5. Concluding Remarks and Open Problems

We have proved that the problem of uniform scattering in a grid is indeed solvable without collisions even for very weak robots. The results of this paper have been proven for square grids. They however hold also in rectangular grids, provided that the direction of the largest dimension is known.

Although the focus of this work has been on the computability rather than the complexity aspects of the problem, it is easy to verify that the proposed solution requires no more than $O(N/d)$ time units in the worst case. An interesting open research question is whether this cost is optimal, or a more efficient solution strategy can be designed.

An important research direction involves the study of the problem in more complex orthogonal spaces (e.g., with holes). It appears doubtful that these cases could be solved without increasing the capabilities of the robots. In particular, the computational power and performance capabilities of weak robots that are however

provided with explicit communication seems to be a promising direction for future research on the uniform deployment problem.

Acknowledgements

The authors would like to thank the anonymous referees for their comments and observations, which have helped improving the presentation of the results. This work has been supported in part by NSERC Discovery grants, and by Dr. Flocchini's University Research Chair.

References

1. S. Das, E. Mesa Barrameda, and N. Santoro. Deployment of asynchronous robotic sensors in unknown orthogonal environments *Proceedings of the 4th International Workshop on Algorithmic Aspects of Wireless Sensor Networks* (ALGOSENSOR'08), 125-140, 2008
2. R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science* 399 (1-2): 71-82, 2008.
3. X. Défago, and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science* 396 (1-3): 97-112, 2008.
4. P. Flocchini, G. Prencipe, and N. Santoro. Self-deployment algorithms for mobile sensors on a ring. *Theoretical Computer Science* 402 (1): 67-80, 2008.
5. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science* 407 (1-3): 412-447, 2008.
6. A. Ganguli, J. Cortes, and F. Bullo. Visibility-based multi-agent deployment in orthogonal environments *Proceedings of the American Control Conference*, 3426-3431, 2007.
7. N. Heo and P. K. Varshney. A distributed self spreading algorithm for mobile wireless sensor networks. *Proceedings of the IEEE Wireless Communication and Networking Conference*, volume 3, 1597–1602, 2003.
8. N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 35 (1): 78–92, 2005.
9. A. Howard, M. J. Mataric, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots* 13 (2): 113–126, 2002.
10. A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems* (DARS'02), 299–308, 2002.
11. T.-R. Hsiang, E. Arkin, M. A. Bender, S. Fekete, and J. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. *Proceedings of the 5th Workshop on Algorithmic Foundations of Robotics* (WAFR), 77–94, 2002.
12. M. Kasuya, N. Ito, N. Inuzuka, and K. Wada. A pattern formation algorithm for a set of autonomous distributed robots with agreement on orientation along one axis. *Systems and Computers in Japan* 37 (10): 747-757, 2006,
13. B. Katreniak. Biangular circle formation by asynchronous mobile robots. *Proceedings of the 12th International Colloquium on Structural Information and Com-*

munication Complexity (SIROCCO'05), 185–199, 2005.

14. Y. Ikemoto, Y. Hasegawa, T. Fukuda, and K. Matsuda. Gradual spatial pattern formation of homogeneous robot group. *Information Sciences* 171 (4): 431-445, 2005
15. X. Li and N. Santoro. An integrated self-deployment and coverage maintenance scheme for mobile sensor networks. In *Proceedings of the 2nd International Conference on Mobile Ad-Hoc and Sensors Networks* (MSN'06), 847-860, 2006.
16. S. Poduri and G.S. Sukhatme. Constrained coverage for mobile sensor networks. In *Proceedings of the IEEE International Conference on Robotic and Automation*, 165–173, 2004.
17. K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotics Systems* 13: 127–139, 1996.
18. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing* 28 (4): 1347–1363, 1999.