Deployment of Asynchronous Robotic Sensors in Unknown Orthogonal Environments *

Eduardo Mesa Barrameda[1], Shantanu Das[2], and Nicola Santoro[3]

[1] Universidad de La Habana, Cuba, eduardomesa@matcom.uh.cu
[2] ETH Zurich, Switzerland, shantanu.das@inf.ethz.ch
[3] Carleton University, Canada, santoro@scs.carleton.ca

Abstract. We consider the problem of uniformly dispersing mobile robotic sensors in a simply connected orthogonal space of unknown shape. The mobile sensors are injected into the space from one or more entry points and rely only on sensed local information within a restricted radius. Unlike the existing solution, we allow the sensors to be asynchronous and show how, even in this case, the sensors can uniformly fill the unknown space, avoiding any collisions and without using any explicit communication, endowed with only $O(1)$ bits of persistent memory and $O(1)$ visibility radius. Our protocols are memory- and radius- optimal; in fact, we show that filling is impossible without persistent memory (even if visibility is unlimited); and that it is impossible with less visibility than that used by our algorithms (even if memory is unbounded).

1 Introduction

The Framework: An important problem for wireless sensor systems is the effective deployment of the sensors within the target space S. The deployment must usually satisfy some optimization criteria with respect to the space S (e.g., maximize coverage). In case of static sensors, they are usually deployed by external means, either carefully (e.g., manually installed) or randomly (e.g., dropped by an airplane); in the latter case, the distribution of the sensors may not satisfy the desired optimization criteria.

If the sensing entities are *mobile*, as in the case of mobile sensor networks, vehicular networks, and robotic sensor networks, they are potentially capable to position themselves in appropriate locations without the help of any central coordination or external control. However to achieve such a goal is a rather complex task, and designing localized algorithms for efficient and effective deployment of the mobile sensors is a challenging research issue.

We are interested in a specific instance of the problem, called the *Uniform Dispersal* (or *Filling*) problem, where the sensors have to completely fill an unknown space S entering through one or more designated entry points called *doors*. In the process, the sensors must avoid colliding with each other, and must terminate (i.e., reach a quiescent state) within finite time. The space S is

* Research partially supported by NSERC Canada.

assumed to be simply connected (i.e., without holes), and orthogonal, i.e. polygonal with sides either parallel or perpendicular to one another (e.g., see Figure 1). Orthogonal spaces are interesting because they can be used to model indoor and urban environment.

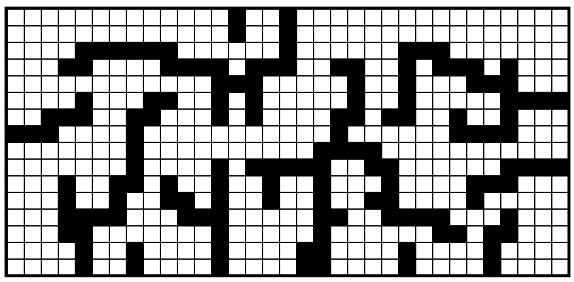

Fig. 1. An orthogonal region to be filled by the sensors.

We wish to study the problem from an algorithmic point of view, focussing on the minimum capabilities required by the sensors in order to effectively complete this task. We consider this problem within the context of robotic sensors networks: the mobile entities rely only on sensed local information within a restricted radius, called *visibility range*; when active they operate in a *sense-compute-move* cycle; and usually they have no explicit means of communication.

Existing Results: The problem of deployment of *mobile sensor networks* has been studied by several authors and continues to be the subject of extensive research; e.g., see [7–10, 13, 16, 22]. Most of the work is focused on the *uniform self-deployment* problem; that is, how to achieve uniform deployment in S (usually assumed to be polygonal) starting from an initial random placement of the sensors in S. The *uniform dispersal* problem, studied here, has been previously investigated by Howard et al. [9]: sensors are deployed one-at-a-time into an unknown environment, and each sensor uses information gathered by previously deployed sensors to determine its deployment location.

The *robotic sensor networks*, studied in this paper, have been and continue to be the object of extensive investigations both from the control and the computing point of view (e.g., [1, 3–6, 12, 14, 20, 21]; see [2, 18] for recent surveys). A crucial difference between robotic sensor networks and traditional wireless sensor networks is in the determination of an entity's neighbours. In robotic sensor networks, the determination of one's neighbours is done by sensing capabilities (e.g., vision): any sensor in the sensing radius is detected even if inactive. On the other hand, in traditional wireless sensor networks, determination of the neighbours is achieved by radio communication; since an inactive sensor does not participate in any communication, the simple activity of determining one's neighbours, to be completed, requires the use of randomization or the presence of sophisticated synchronization and scheduling mechanisms (e.g., [15, 17]). Both

problems, *uniform self-deployment* and *uniform dispersal* have been studied for robotic sensor networks.

The *uniform self-deployment* problem for robotic sensor networks has been studied recently, and localized solution algorithms have been developed when the space S is a *line* (e.g., a rectilinear corridor) [3], and when it is a *ring* (e.g., the boundary of a convex region) [4]. The proposed solutions operate even if the sensors are very weak; indeed they are *anonymous* (i.e., indistinguishable), *oblivious* (i.e., without any recollection of computations and actions performed in the previous activity cycles), *asynchronous* (i.e., when the time between successive activity cycles is finite but unpredictable), and are *communication-free* (i.e., they use no explicit form of communication).

The *uniform dispersal* problem for robotic sensor networks, in which the sensors are injected one-at-a-time into the unknown environment S, has been introduced and investigated by Hsiang et al.[11]. Their results are based on an ingenious follow-the-leader technique where each sensor communicates with the one following it and instructions to move are communicated from predecessor to successor. The sensors are anonymous but they need some *persistent memory* to remember whether or not is a leader[4] and the direction of its movement. Since the algorithm uses only $O(1)$ bits of working memory in total, computationally the sensors can be just *finite-state machines*. In addition to requiring explicit communication, the solution of [11] makes the strong assumption that the sensors operate *synchronously*, which allows perfect coordination between the sensors.

This fact opens a series of interesting questions on the capabilities needed by the sensors to achieve uniform dispersal in orthogonal environments. In other words, how "weak" the sensors can be and still be able to uniformly disperse ? In particular: is persistent memory needed ? can the task be performed if the sensors are asynchronous ? is explicit communication really necessary ? In this paper we consider precisely these questions and provide some definite answers.

Our Results: We identify intrinsic limitations on the type of memory and the visibility radius of needed by asynchronous sensors to solve the uniform dispersal problem for any simply connected orthogonal space whose shape is a priori unknown. We then show that these results are tight; in fact we present protocols and prove that they solve the problem while meeting these limitations, and without using explicit communication.

We first consider the case of a *single door*. We show that oblivious sensors can *not* deterministically solve the problem, even if they have unlimited visibility. We then present (in section 4) an algorithm for solving the problem in the case of single door with asynchronous identical sensors having persistent working memory of only two bits and a visibility radius of just one unit.

For the case of *multiple doors*, we prove that asynchronous sensors can *not* solve the problem if the visibility radius is less than two units, even if they have unbounded memory. On the other hand, even with unbounded visibility and memory, the problem is still unsolvable if the sensors are identical. Thus, we assume that sensors entering the space from different doors have different

[4] There is one leader for each door.

colors (i.e. they are distinguishable). We prove that under this assumption, the problem can be solved with sensors having visibility radius two and constant (persistent) memory. The proof is constructive: we present the radius-optimal and memory optimal distributed algorithm for achieving this (in section 5).

Let us stress that, in our algorithms sensors use only constant memory; hence, they can be simple finite-state machines like in [11]. Unlike those in [11], our algorithms work for the asynchronous case; hence they are robust against occasional stalling of the sensors.

2 Model, Definitions and Properties

2.1 Sensors and Dispersal

The space to be filled by the sensors is a simply connected orthogonal region S that is partitioned into square cells each of size roughly equal to the area occupied by a sensor. Simply connected means that it is possible to reach any cell in the space from any other cell and there are no obstacle surrounded completely by cells belonging to the space.

The system is composed of simple entities, called sensors, having sensory and locomotion capabilities. The entities can turn and move in any direction. The sensory devices on the entity allows it to have a vision of its immediate surrounding; we assume the sensors to have restricted vision up to a fixed radius around it[5]. Even if two sensors see each-other, they do not have any explicit means of communicating with each-other. Each sensor functions according to an algorithm preprogrammed into it. The sensors have a $O(1)$ bits of working memory, and they have a local sense of orientation (i.e., each sensor has a consistent notion of "up-down" and "left-right");

If two sensors are in the same cell at the same time then there is a *collision*. The algorithm executed by the sensors must avoid collisions (e.g., to prevent damage to the sensor or its sensory equipment).

The sensors enter the space through special cells called doors [11]. A door is simply a cell in the space which always has a sensor in it. Whenever the sensor in the door moves to a neighboring cell, a new sensor appears instantaneously in the door. A sensor may not distinguish a door cell from an ordinary cell, using its sensory vision.

During each step taken by a sensor, the sensor first looks at its surrounding (up to its visibility radius) and then based on the rules of the algorithm, the sensor either chooses one of the neighboring cells to move to, or decides to remain stationary. Each step is atomic and during one step a sensor can only move to a neighboring cell. However, since the sensors are asynchronous, an arbitrary amount of time may lapse between two steps taken by a sensor.

The problem to be solved is that of *uniform dispersal* (or *filling*) : within finite time, the entire space must be *filled*, i.e., every cell of the space is occupied by a sensor; furthermore the system configuration at that time must be *quiescent*,

[5] A visibility radius of one means that the robot sees all eight neighboring cells.

i.e., no sensor moves thereafter. The goal is to design a protocol P, the same for all sensors, that specifies which operations a sensor must perform whenever it is active, and that will always correctly within a finite time and without collisions lead the system to a quiescent configuration where the entire space is filled.

2.2 Space Representation and Properties

Let $A = A_y \times A_x$ be the smallest rectangular area containing the space S. We consider a partition of the area A into pixels $p_{i,j}$, $1 \leq i \leq r$, $1 \leq j \leq c$ where $r = A_y/q$, $c = A_x/q$ and q is the length of each cell in the space. Thus, some of the pixels (called valid pixels) correspond to the cells in the space while the other pixels represent obstacles. We represent the structure of the space S in the form a graph $G = (N, E)$ defined as follow:

- Each column of the space is partitioned into segments of consecutive valid pixels ended by an obstacle in both extremes and numbered from top to down.
- Each segment is a node of G. We denote by $l_j^k \in N$ the node corresponding to the $k-th$ segment of column j.
- We denote by $d\_p_j^k$ the bottom-most pixel of the segment l_j^k.
- There is an edge $(l_j^k, l_{j'}^{k'}) \in E$ if and only if:
  (a) $j = j' + 1$ or $j = j' - 1$ **and**
  (b) There is a pixel $p_{i,j'} \in l_{j'}^{k'}$ neighbor to $d\_p_j^k$ or there is a pixel $p_{i,j} \in l_j^k$ neighbor to $d\_p_{j'}^{k'}$.

The following propositions are easy to verify.

Proposition 1. *If l_j^k and $l_{j'}^{k'}$ are two distinct nodes of G then there is at most one edge between them.*

Proposition 2. *Two nodes l_j^k and $l_{j'}^{k'}$ have neighboring pixels if and only if there is an edge $(l_j^k, l_{j'}^{k'}) \in E$ between them.*

Proposition 3. *The graph G is connected.*

Proposition 4. *The graph G is acyclic.*

If there is an edge $(l_j^k, l_{j'}^{k'})$ such that the bottom-most pixel $d\_p_j^k = p_{i,j}$ of l_j^k is a neighbor of the pixel $p_{i,j'} \in, l_{j'}^{k'}$, we say that $p_{i,j}$ is the ***entry point*** from l_j^k to $l_{j'}^{k'}$ and $p_{i,j'}$ is the ***entry point*** from $l_{j'}^{k'}$ to l_j^k.

3 Impossibility Results

We first show that the sensors must have some *persistent* memory of the past, for solving the filling problem successfully.

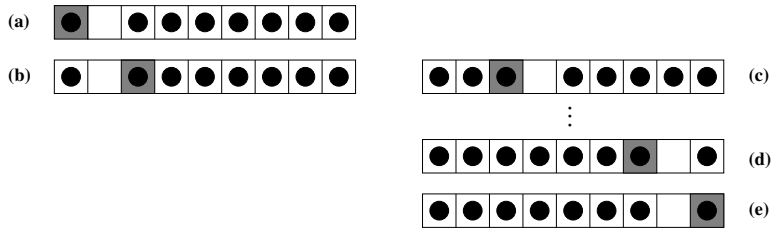

Fig. 2. (a) and (b) are indistinguishable configurations. (c) to (d) are allowable configurations; (e) is an unavoidable configuration.

Theorem 1. *The filling problem can not be solved by oblivious sensors, even if they have unbounded visibility. This result holds even if there is only a single door.*

Proof. Consider the space consisting of a single line of $n = 2m+1$ pixels of which one of them is a door. By contradiction, let P be a correct filling protocol. Since the sensors have no memory of past, each step taken by a sensor depends only on the current configuration (i.e. which cells are filled and which are empty). We can represent each empty cell by 0 and each filled cell by 1; the door would be represented by D; however note that it is not distinguishable from a filled cell. A configuration can thus represented by the sequence $< d_1...d_n >$ of the the values of the cells left-to-right. If algorithm P is correct then the penultimate configuration (i.e., the final configuration before the space is completely filled), must have exactly one empty cell and this cell should be adjacent to the door. So, if the door is the leftmost cell then the only possible final configuration is $< D011....11111 >$. Notice that this is indistinguishable from the configuration $< 10D11....1111 >$ and the algorithm must make the same move in both cases. In the former situation, the leftmost robot (from the door) must move to the right, but the same move will leave the space unfilled in the latter scenario. So the configuration $< 10D11....1111 >$ must be avoided by the algorithm; this implies that the only correct penultimate configuration when the door is the third cell is $< 11D01....1111 >$. Extending this argument inductively, the only correct penultimate configuration when the door is the $2i+1$-th cell ($0 \leq i < m$), is the one where $d_{2i+1} = D$, $d_{2(i+1)} = 0$, and all other d_j's are 1. Hence the only correct penultimate configuration when the door is the $2(m-1)+1$-th cell, must be $< 11111....1D01 >$. Notice that this configuration is indistinguishable from $< 11111....110D >$ which thus must be avoided by the algorithm, However this is the only possible penultimate configuration when the door is the rightmost cell. A contradiction.

Theorem 2. *In the case of multiple doors, it is impossible for asynchronous sensors to solve the filling problem avoiding collisions, if the visibility radius is less than two. The result holds even if the sensors have distinct visible identities and each sensor has an unbounded amount of memory.*

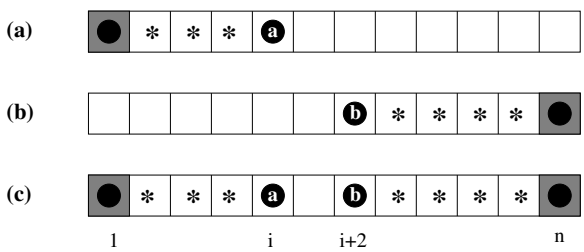

Fig. 3. A single line of cells with doors at one or both ends.

Proof. (Sketch) By contradiction, let P be a correct filling protocol for asynchronous sensors in simply-connected orthogonal spaces with multiple doors. Consider the space consisting of a single line of pixels, with a door at each end. Since the sensors at one door do not know of the existence of the other door, P must force the sensor initially at the left door (sensor 'a') to move towards the right end of the line (as in the case when there is no right door: figure 3(a)). Similarly, the protocol must force the sensor initially at the right door (sensor 'b') to move into the corridor towards the left end of the line (figure 3(b)).Thus, if both doors were present the sensors 'a' and 'b' must move towards each other. It is possible for an adversary to schedule the activations of the sensors (i.e., to choose the finite delays between successive activity cycles) so as to generate the situation where both 'a' and 'b' are about to enter the same empty cell that lies between them (see figure 3(c)). Since the visibility is less than two, they do not see each other but only the empty cell. By scheduling both sensors to move at the same time, the adversary generates a collision during the execution of P; this contradicts the assumption that P is correct and thus collision-free. Notice that even if the sensors remember a complete history of their past moves and have distinct visible IDs, the collision can not be avoided.

Theorem 3. *If sensors entering from distinct doors are indistinguishable from each other, then there is no collision-free solution to the filling problem for multiple doors. The result holds irrespective of the visibility range of the sensors.*

Proof. Consider the space S consisting of a single column of $n-2$ cells ($n > 2$) and two extra cells adjacent to the column—one on the left and one on the right (see Figure 4). We consider three cases: (i) The cell on the left of the column is the only door, (ii) The cell on the right is the only door, and (iii) Both the cells to the left and the right of the column are doors. As before we consider the penultimate configuration reached by any correct filling algorithm for each of these cases, shown in Figure 4(a), (b) and (c) respectively. Notice that there is only one cell adjacent to the door in each case and thus this cell must be empty in the penultimate configuration. For cases (i) and (iii), let 'a' be the sensor currently at the door on the left. For cases (ii) and (iii), let 'b' be the sensor currently at the door on the right. It is possible for an adversary to schedule the

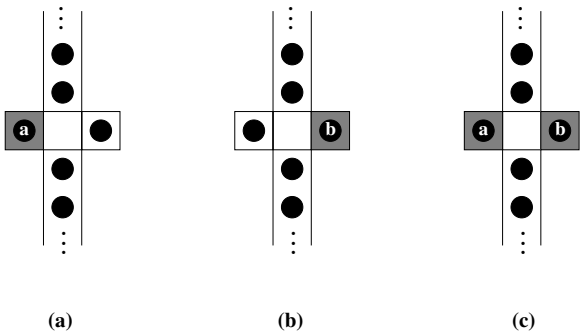

Fig. 4. (a) Single door on the left (b) Single door on the right (c) Indistinguishable sensors entering the column from two doors

activation of the sensors in such a way that both 'a' and 'b' become active for the first time only after reaching the penultimate configuration. Thus, neither sensor has any past history and any decision taken by sensor 'a' (or sensor 'b') would depend only on the current configuration. Notice that the current configuration in the three cases are indistinguishable from each other. So, sensor 'a' must take the same decision for case (i) and (iii). Similarly, sensor 'b' must take the same decision in case (ii) and (iii). If one of the sensors decides not to move to the empty cell, then the space remains unfilled in at least one of scenarios. On the other hand, if both decide to move, the adversary can force a collision for case (iii), by scheduling them to move at the same time.

In Section 5 we show that if the sensors have visibility radius at least two and sensors coming from distinct doors are distinguishable then there is a solution to the filling problem for any connected space with any number of doors.

4 Filling Algorithm: Single Door

In this section we consider the case when is only one door through which the sensors enter the space. We show that visibility radius of one and a constant amount of memory for each sensor, is sufficient in this case. Each sensor just needs one bit of memory, to remember its last location, so that it never backtracks. The idea of the algorithm (SINGLE) is to move the robots along the paths in G, starting from the node containing the door. Since the sensor can see the eight neighboring pixels, it can determine when it has reached an *entry point*.

Following the rules of algorithm SINGLE, any path of consecutive pixels in the space on which a sensor is allowed to travel is called a *valid path*. Notice that any valid path corresponds to some path in G.

Theorem 4. *Algorithm SINGLE solves the filling problem for any space of size n and a single door, without any collisions and using n sensors each having a constant amount of memory and a visibility radius of one.*

Algorithm 1 SINGLE

Meta-Rule:

– A sensor never backtracks.

Rules: A sensor r in pixel $p_{i,j}$ executes the following rules:

if ($p_{i+1,j}$ is empty) **then**
  r moves to $p_{i+1,j}$.
else if ($p_{i-1,j}$ is empty) **then**
  r moves to $p_{i-1,j}$.
else if (($p_{i,j-1}$ is empty) $\wedge$
    (($p_{i-1,j-1}$ is **obstacle**) $\vee$ ($p_{i-1,j}$ is **obstacle**))) **then**
  r moves to $p_{i,j-1}$.
else if (($p_{i,j+1}$ is empty) $\wedge$
    (($p_{i-1,j+1}$ is **obstacle**) $\vee$ ($p_{i-1,j}$ is **obstacle**))) **then**
  r moves to $p_{i,j+1}$.
else
  r does not move.
end if

5 Filling Algorithm: Multiple Doors

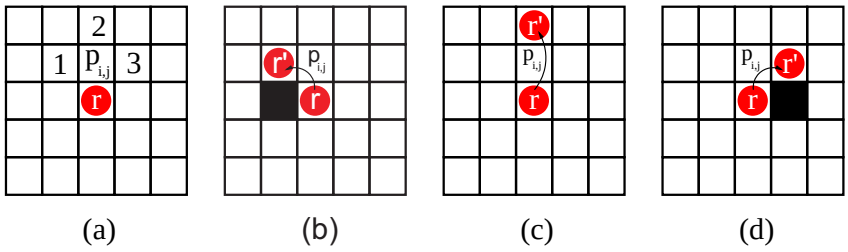

Fig. 5. Sensor r needs to check cells 1,2, and 3. The sensor (of the same color) that last visited $p_{i,j}$ must be in one of these locations.

If there are multiple doors, then we know that the sensors must have a visibility radius of at least two and they should not be indistinguishable. We assume sensors coming from different doors have different colors and each sensor has visibility radius of two. Our algorithm for this case, uses the following restriction on the movement of the sensors.

Meta-Rule "A sensor may not move until its previous position is occupied".

The idea of our algorithm (called algorithm MULTIPLE) is sketched here. Sensors coming from different doors (i.e. sensors of different colors) follow distinct paths in G and these paths do not intersect. In other words, we ensure that

the cells visited by sensors of color c_i are occupied by sensors of the same color (and never by sensors of any other color). Thus, a sensor before moving to a pixel $p_{i,j}$ needs to determine if this pixel was visited by sensors of another color. We show a sensor r of color c_r can always determine if the next pixel $p_{i,j}$ (in one of its valid paths) was visited by sensors of the same color (see Figure 5). In that case, the sensor r moves to the pixel $p_{i,j}$. Otherwise there may be two cases: (i) either pixel $p_{i,j}$ was visited by sensors of another color or (ii) pixel $p_{i,j}$ was never visited before. In the first case, sensor r does not move to pixel $p_{i,j}$ and searches for alternate paths. In the second case, sensor r needs to take a decision based on whether there are other sensors waiting to move into pixel $p_{i,j}$. In case there are two or more sensors in cells neighboring an unvisited pixel $p_{i,j}$, the sensors are assigned priorities[6] based whether they are coming from the left, right, top or bottom (in that order). The sensor with the highest priority (among those sensors for whom $p_{i,j}$ is a valid move) moves to pixel $p_{i,j}$. Notice that it may not be always possible for a sensor r to determine if its neighbor pixel $p_{i,j}$ is unvisited or has been visited by a sensor of another color (see Figure 6). If that is the case, then sensor r simply waits until the situation changes (so that it is able to take a decision). We shall show that such waiting never results in a deadlock, as the sensor with the highest priority to move is always able to decide, without waiting.

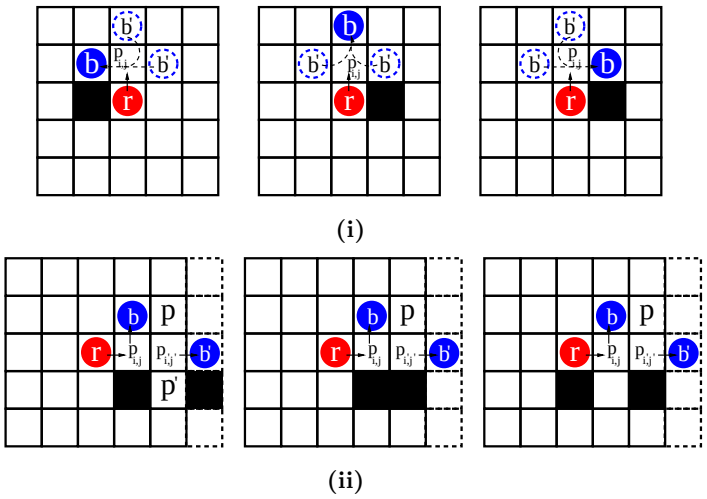

Fig. 6. (i) To determine if $p_{i,j}$ was visited by sensor b, sensor r should be able to see both b and its successor b. (ii) If b' has moved out of the visibility range of r then there must be a sensor of the same color in cell P (or P').

[6] This is the only way to avoid collision as well as ensure progress of the algorithm.

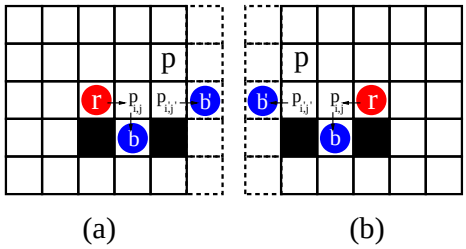

Fig. 7. If b' has moved out of the visibility range of r then there must be a sensor of the same color in the cell marked P.

The formal description of algorithm MULTIPLE is given in Algorithm 2. The functions isValidUp(), isValidDown(), isValidLeft() and isValidRight() express how the sensor makes a decision if it can move or not to an specific neighboring pixel. We use the following notations in the algorithm:

- $r.Color$ is the color of the door sensor r came from.
- If a pixel $p_{i,j}$ is occupied by some sensor r, then $(p_{i,j}).Color = r.Color$. Otherwise $(p_{i,j}).Color = None$.

Algorithm 2 MULTIPLE

Meta-Rules:

- A sensor never backtracks.
- A sensor does not move if its previous position is empty.

Rules: A sensor r in a pixel $p_{i,j}$ of l_j^k executes the following:

if isValidUp(r) **then**
  r moves to $p_{i+1,j}$.
else if isValidDown(r) **then**
  r moves to $p_{i-1,j}$.
else if isValidLeft(r) **then**
  r moves to $p_{i,j-1}$.
else if isValidRight(r) **then**
  r moves to $p_{i,j+1}$.
else
  r does not move.
end if

Proposition 5. *The following properties hold during the execution of the algorithm MULTIPLE.*

(i) A sensor r can always determine if a neighboring pixel $p_{i,j}$ was visited by sensors from the same door.

(ii) The sensor r having the highest priority to move into a pixel $p_{i,j}$ can always determine if the pixel $p_{i,j}$ is unvisited or not.

(iii) Two sensors from different doors never visit the same pixel (i.e. no intersections).

(iv) Two sensors are never in the same pixel at the same time (i.e. no collisions).

Proposition 6. *The algorithm MULTIPLE terminates in a finite time.*

Based on the facts that there are no collisions and the sensors never re-visit the same pixel, we can prove this proposition in the same way as for the previous algorithm.

Proposition 7. *On termination of algorithm MULTIPLE, there are no empty pixels in the space.*

Finally, we have the following result regarding the correctness of our algorithm.

Theorem 5. *The algorithm MULTIPLE completely fills any connected space, without collisions, even when the sensors enter from multiple doors (assuming they have distinct colors). The algorithm requires n sensors each having visibility radius two and a constant amount of memory.*

Algorithm 3 Function IsValidDown($<$ *sensor* $>$)

```
bool IsValidDown( r ){
  if p_{i-1,j} is occupied then
    return false
  else if ( (p_{i-2,j} is empty) ∧
          ( (p_{i-1,j-1} is empty) ∨ ((p_{i-2,j-1} is not obstacle) ∨
                                      (p_{i-2,j} is not obstacle )) ) ∧
          ( (p_{i-1,j+1} is empty) ∨ ((p_{i,j+1} is not obstacle) ∨
                                      (p_{i-2,j} is not obstacle))) ) then
    return true
  else if ( ((p_{i-2,j}).Color = r.Color) ∨
          ( ((p_{i-2,j-1} is obstacle) ∨ (p_{i-2,j} is obstacle) ) ∧
            ((p_{i-1,j-1}).Color = r.Color)) ∨
           ( ((p_{i-2,j+1} is obstacle) ∨ (p_{i-2,j} is obstacle) ) ∧
            ((p_{i-1,j+1}).Color = r.Color)) ) then
    return true
  else
    return false
  end if
}
```

Algorithm 4 Function IsValidUp($< sensor >$)

```
bool IsValidUp( r ){
  if p_{i+1,j} is occupied then
    return false
  else if ( (p_{i+2,j} is empty) ∧
            ( (p_{i+1,j-1} is empty) ∨ (p_{i,j-1} is not obstacle) ) ∧
            ( (p_{i+1,j+1} is empty) ∨ (p_{i,j+1} is not obstacle) ) ) then
    return true
  else if ( ((p_{i+2,j}).Color = r.Color) ∨
            ((p_{i,j-1} is obstacle) ∧ ((p_{i+1,j-1}).Color = r.Color)) ∨
              ((p_{i,j+1} is obstacle) ∧ ((p_{i+1,j+1}).Color = r.Color)) ) then
    return true
  else if ( ((p_{i,j-1} is obstacle) ∧ (p_{i+1,j-1} is occupied)) ∨
              ((p_{i,j+1} is obstacle) ∧ (p_{i+1,j+1} is occupied)) ) then
    return false
  else if ( ( (p_{i,j-1} is obstacle) ∧
                (((p_{i+1,j-1}).Color = (p_{i+2,j}).Color) ∨
                 ((p_{i+2,j-1}).Color = (p_{i+2,j}).Color) ∨
                 ((p_{i+1,j-2}).Color = (p_{i+2,j}).Color))) ∨
              ( (p_{i,j+1} is obstacle) ∧
                (((p_{i+1,j+1}).Color = (p_{i+2,j}).Color) ∨
                 ((p_{i+2,j+1}).Color = (p_{i+2,j}).Color) ∨
                 ((p_{i+1,j+2}).Color = (p_{i+2,j}).Color))) ) then
    return false
  else
    return true
  end if
}
```

Algorithm 5 Function IsValidLeft($< sensor >$)

```
bool IsValidLeft( r ){
  if ( (p_{i,j-1} is occupied) ∨
       ( (p_{i-1,j-1} is not obstacle) ∧ (p_{i-1,j} is not obstacle) ) )  then
    return  false
  else if ( (p_{i+1,j-1} is empty) ∧ (p_{i-1,j-1} is empty) ∧
            ( (p_{i,j-2} is empty) ∨
              ( (p_{i-1,j-2} is not obstacle) ∧ (p_{i-1,j-1} is not obstacle) ) ) ) then
    return  true
  else if ( ((p_{i+1,j-1}).Color = r.Color) ∨ ((p_{i-1,j-1}).Color = r.Color) ∨
            ( ((p_{i,j-2}).Color = r.Color) ∧
              ( (p_{i-1,j-2} is obstacle) ∨ (p_{i-1,j-1} is obstacle) ) ) ) then
    return  true
  else if ( (p_{i,j-2} is occupied) ∧
            ( (p_{i-1,j-2} is obstacle) ∨ (p_{i-1,j-1} is obstacle) ) ) ) then
    return  false
  else if ( ( ((p_{i+1,j-1}).Color = (p_{i+1,j-2}).Color) ∧
              ( (p_{i-1,j-2} is obstacle) ∨ (p_{i-1,j-1} is obstacle) ) ) ∨
            ( ((p_{i+1,j-1}).Color = (p_{i-1,j-2}).Color) ∧
              ( (p_{i-1,j-1} is obstacle) ∨ (p_{i-2,j-2} is obstacle) ∨
                (p_{i-2,j-1} is obstacle) ) ) ∨
            ((p_{i+1,j-1}).Color = (p_{i-1,j-1}).Color) ∨
            ((p_{i+1,j-1}).Color = (p_{i-2,j-1}).Color) ∨
            ( ((p_{i-1,j-1}).Color = (p_{i+1,j-2}).Color) ∧
              ( (p_{i-1,j-2} is obstacle) ∨ (p_{i,j-2} is obstacle) ) ) ∨
            ((p_{i-1,j-1}).Color = (p_{i+2,j-1}).Color) ) ) then
    return  false
  else
    return  true
  end if
}
```

Algorithm 6 Function IsValidRight($< sensor >$)

```
bool IsValidRight( r ){
  if ( (p_{i,j+1} is occupied) ∨
       ( (p_{i-1,j+1} is not obstacle) ∧ (p_{i-1,j} is not obstacle) ) )  then
    return  false
  else if ( (p_{i+1,j+1} is empty) ∧ (p_{i-1,j+1} is empty) ∧
            ( (p_{i,j+2} is empty) ∨
              ( (p_{i-1,j+2} is not obstacle) ∧ (p_{i-1,j+1} is not obstacle) ) ) ) then
    return  true
  else if ( ((p_{i+1,j+1}).Color = r.Color) ∨ ((p_{i-1,j+1}).Color = r.Color) ∨
            ( ((p_{i,j+2}).Color = r.Color) ∧
              ( (p_{i-1,j+2} is obstacle) ∨ (p_{i-1,j+1} is obstacle) ) ) ) then
    return  true
  else if ( ( ( ((p_{i,j+2}).Color = (p_{i+1,j+1}).Color) ∨
                ((p_{i,j+2}).Color = (p_{i+2,j+1}).Color) ) ∧
              ( (p_{i-1,j+2} is obstacle) ∨ (p_{i-1,j+1} is obstacle) ) ) ∨
            ( ( ((p_{i,j+2}).Color = (p_{i-1,j+1}).Color) ∨
                ((p_{i,j+2}).Color = (p_{i-2,j+1}).Color) ) ∧
              (p_{i-1,j+2} is obstacle) ) ∨
            ( ((p_{i+1,j+1}).Color = (p_{i+1,j+2}).Color) ∧
              ( (p_{i-1,j+2} is obstacle) ∨ (p_{i-1,j+1} is obstacle) ) ) ∨
            ( ((p_{i+1,j+1}).Color = (p_{i-1,j+2}).Color) ∧
              ( (p_{i-1,j+1} is obstacle) ∨ (p_{i-2,j+2} is obstacle) ∨
                (p_{i-2,j+1} is obstacle) ) ) ∨
            ((p_{i+1,j+1}).Color = (p_{i-1,j+1}).Color) ∨
            ((p_{i+1,j+1}).Color = (p_{i-2,j+1}).Color) ∨
            ( ((p_{i-1,j+1}).Color = (p_{i+1,j+2}).Color) ∧
              ( (p_{i-1,j+2} is obstacle) ∨ (p_{i,j+2} is obstacle) ) ) ∨
            ((p_{i-1,j+1}).Color = (p_{i+2,j+1}).Color) ) ) then
    return  false
  else
    return  true
  end if
}
```

6 Conclusions and Open Problems

We have shown that, for uniform dispersal in simply connected orthogonal spaces, synchronicity and explicit communication are not necessary, while persistent memory is needed. More precisely, we have presented localized algorithms (one in the case of a single entry point, and one in the case of multiple entry points) that allow asynchronous mobile sensors to fill simply connected orthogonal spaces of unknown shape; the sensors do so without collisions and without any explicit direct communication, endowed with only $O(1)$ bits of persistent memory and $O(1)$ visibility radius. In both cases, the protocols are memory- and radius- optimal; in fact, we have shown that filling is impossible without persistent memory (even if visibility is unlimited); it is also impossible with less visibility than that used by our algorithms (even if memory is unbounded).

There are many interesting research problem still open. For example, orthogonal spaces that are not simply connected (i.e., containing holes) can be filled by synchronous sensors [11], but asynchronous solutions are not yet available. The study of the filling problem in more general classes of spaces is still open both in the synchronous and asynchronous settings. Another interesting direction for future research is to study the impact of having communication capabilities.

References

1. N. Agmon and D. Peleg, "Fault-tolerant gathering algorithms for autonomous mobile robots". *SIAM J. on Computing* 36: 56-82, 2006.
2. F. Bullo, J. Cortes, and S. Martinez. "Distributed algorithms for robotic networks". In R. Meyers (Ed.), *Encyclopedia of Complexity and Systems Science*. Springer Verlag, to appear, 2008.
3. R. Cohen and D. Peleg. "Local algorithms for autonomous robot systems". In Proc. . *13th Colloquium on Structural Information and Communication Complexity*, 29–43, 2006.
4. P. Flocchini, G. Prencipe, and N. Santoro. "Self-deployment of mobile sensors on a ring". *Theoretical Computer Science*, Special issue of ALGOSENSORS'06 selected papers, 2008.
5. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. "Gathering of asynchronous mobile robots with limited visibility". *Theoretical Computer Science* 337:147–168, 2005.
6. A. Ganguli, J. Cortes, and F. Bullo. "Visibility-based multi-agent deployment in orthogonal environments". In Proceedings *American Control Conference*, 3426-3431, 2007.
7. N. Heo and P. K. Varshney. "A distributed self spreading algorithm for mobile wireless sensor networks". In Proceedings *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* 35(1):78–92, 2005.
9. A. Howard, M.J. Mataric, and G.S. Sukahatme. "An incremental self-deployment algorithm for mobile sensor networks". *IEEE Transactions on Robotics and Automation* 13(2): 113-126, 2002.

10. A. Howard, M. J. Mataric, and G. S. Sukhatme. "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem". In Proceedings *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 environment". In Proc. *5th Workshop on Algorithmic Foundations of Robotics* (WAFR), 77-94, 2002.
12. J. Lee, S. Venkatesh, and M. Kumar. "Formation of a geometric pattern with a mobile wireless sensor network". *Journal of Robotic Systems* 21(10): 517–530,2004.
13. L. Loo, E. Lin, M. Kam, and P. Varshney. "Cooperative multi-agent constellation formation under sensing and communication constraints". *Cooperative Control and Optimization*, 143–170, 2002.
14. E. Martinson and D. Payton. "Lattice formation in mobile autonomous sensor arrays". In Proc. *International Workshop on Swarm Robotics* (SAB'04), 98-111, 2004
15. S. E. Nikoletseas. "Models and algorithms for wireless sensor networks". In Proc. *32nd Conference on Current Trends in Theory and Practice of Computer Science*, 64–83, 2006.
16. S. Poduri and G.S. Sukhatme. "Constrained coverage for mobile sensor networks". In Proc. *IEEE Int. Conference on Robotic and Automation*, 165–173, 2004.
17. O. Powell, P. Leone, and J. Rolim. "Energy optimal data propagation in wireless sensor networks". *Journal of Parallel and Distributed Computing*, 67(3):302–317, 2007.
18. G. Prencipe and N. Santoro, "Distributed algorithms for mobile robots". In *Proc. 5th IFIP International Conference on Theoretical Computer Science* (TCS'06), 2006.
19. J.H. Reif and H. Wang. "Social potential fields: A distributed behavioral control for autonomous robots". Robotics and Autonomous Systems 27(3): 171-194, 1999.
20. S. Susca, S. Martinez, and F. Bullo. "Monitoring enviromental boundaries with a robotic sensor network". *IEEE Transactions on Control Systems Technology* 16(2):288-296, 2008.
21. I. Suzuki and M. Yamashita. "Distributed anonymous mobile robots: Formation of geometric patterns". *SIAM J. Comput.* 28(4): 1347-1363, 1999.
22. G. Wang, G. Cao, and T. La Porta. Movement-assisted sensor deployment. In Proc. *IEEE INFOCOM*, volume 4, pages 2469–2479, 2004.