# Filling Arbitrary Connected Areas by Silent Robots with Minimum Visibility Range

Attila Hideg[1], Tamás Lukovszki[2], and Bertalan Forstner[1]

[1] Department of Automation and Applied Informatics,
Budapest University of Technology and Economics, Budapest, Hungary
{Attila.Hideg,Bertalan.Forstner}@aut.bme.hu
[2] Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
lukovszki@inf.elte.hu

**Abstract.** We study the uniform dispersal problem (also called the filling problem) in arbitrary connected areas. In the filling problem robots are injected one-by-one at $k \geq 1$ Doors into an unknown area, subdivided into cells. The goal is to cover the area, i.e. each cell must be occupied by a robot. The robots are homogeneous, anonymous, autonomous, have limited visibility radius, limited persistent memory, and silent, i.e. do not use explicit communication. A fundamental question is how 'weak' those robots can be in terms of hardware requirements and still be able to solve the problem, which was initiated by Barrameda et al. [4]. In our previous paper [11] we presented an algorithm which solves the filling problem for orthogonal areas with $O(1)$ bits of persistent memory, 1 hop visibility range and without explicit communication. The algorithm utilized the timing of movements and had $O(n)$ runtime, where $n$ is the number of cells in the area. In this paper, we generalize the problem for silent robots for an arbitrary connected area represented by a graph, while maintaining the 1 hop visibility range. The algorithm is collision-free, it terminates in $O(k \cdot \Delta \cdot n)$ rounds, and requires $O(\Delta \cdot \log k)$ bits of persistent memory, where $\Delta$ is the maximum degree of the graph.

**Keywords:** Autonomous Robots · Filling · Dispersion.

## 1 Introduction

In swarm robotics a huge number of simple, cheap, tiny robots can perform complex tasks collectively. Advantages of such systems are scalability, reliability, and fault tolerance. In recent years much attention has been paid to the cooperative behavior of simple, tiny robots which have to complete a particular task collectively. Many distributed protocols have been developed for a wide range of problems, like gathering, flocking, pattern formation, dispersing, filling, coverage, exploration (e.g. [1–6, 8, 12]; see [7, 10] for recent surveys). In this paper we study the *uniform dispersal* (or *filling*) of synchronous robots in an unknown, connected area.

The filling problem was introduced by Hsiang et al. [12] for an orthogonal area, where the area is represented by pixels that form a connected subset of

integer grid. The robots are placed at the same entry point, called the Door, one-by-one and have to occupy all the pixels. There can be at most one robot per pixel at any given time. When more than one door is present in the area the problem is called *multiple door filling* or *k-door filling*.

Barrameda et al. [4, 5] investigated the minimum hardware requirements and the possibilities of solving the filling problem for orthogonal regions by robots with constant visibility radius, constant communication range, and constant number of bits of persistent memory.

In [5], the authors allowed holes to be present in the map. Two methods were proposed: one without communication (**MUTE**) and one with communication (**TALK**). Both methods worked in the asynchronous (ASYNC) model, therefore, it can be used in the fully-synchronous (FSYNC) model, as well. **MUTE** required a visibility range of 6 and was inspired by the dance of bees, where robots implicitly communicated using only their movements. In **TALK** they required a visibility and communication range of 1 and worked strictly in orthogonal areas as they could see diagonally. Both solutions [4, 5] only required a constant amount of memory.

In [4], common top-down and left-right directions and externally visible colors were assumed for the multiple door filling. In [9] Das et al. showed that allowing visible colors or lights yields a more powerful computational model than allowing infinite visibility range but no lights.

In our previous paper [11], we have further reduced the hardware requirements of the robots for filling orthogonal regions. The robots do not use any communication and also no lights. The robots have a common sense of North and East directions, but they cannot measure their absolute position and do not share a common coordinate system. They need a $O(1)$ bits of persistent memory and the visibility range of the robots is reduced to 1 hop. We have presented an algorithm solving the single-door and the multiple-door filling problem for orthogonal regions in $O(n)$ time in the synchronous computational model. A key element of the algorithm is to reserve time-slots for each possible direction of the movement (labeled as North, East, South, and West) in order to prevent collisions. Unfortunately, this idea could not be extended for the general case, where the area is represented by an arbitrary connected graph (moreover, it relies on notions that are not available in arbitrary graphs: north, south, orientation). A problematic scenario is illustrated in Fig. 1. Using a fixed assignment of time-slots to the edges incident to the occupied vertex of a robot $r_1$ can lead to collision with other robot(s) $r_2$, $r_3$, since another robots can choose the same time slot to target the unvisited vertex $v$.

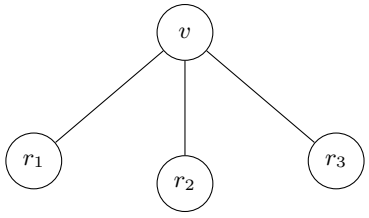


**Fig. 1.** Robots $r_1$, $r_2$ and $r_3$ would move to vertex $v$. Only one can move to $v$ at any given time, however, they are not visible by each-other (1 hop visibility means they can see adjacent vertices).

**Our contribution:** In this paper we present an approach, which is different from [11], to solve the filling problem for any arbitrary connected graphs with robots of visibility range of 1 hop, i.e. the robots can see adjacent vertices. (Note: three-dimensional scenarios and more complex topologies can be modeled.)

First, we present a method, the **Virtual Chain Method** (**VCM**), for the single door filling by a set of autonomous anonymous robots with a visibility radius of 1 hop in $O(\Delta \cdot n)$ time in the synchronous computational model, where $n$ is the number of vertices of the graph with a maximum degree of $\Delta$. The robots require $O(\Delta)$ bits of persistent memory.

Then, we consider the multiple door case, when the robots enter in $k > 1$ doors and we generalize the VCM algorithm for solving this problem. The robots need a visibility range of 1 hop, $O(\Delta \cdot \log k)$ bits of persistent memory, and the algorithm terminates in $O(k \cdot \Delta \cdot n)$ time.

Both algorithms are simple enough to be implemented by a swarm of elementary robots.

Our algorithm is optimal in term of visibility range. This follows from the fact that with a visibility range of less than 1 the robots cannot even distinguish between occupied and unoccupied neighbors. For constant $k$ and constant $\Delta$, our algorithm is asymptotically optimal in the size of the memory. This follows from the result of Barrameda et al. [4]; they proved that oblivious (memoryless) robots cannot deterministically solve the problem. Moreover, for constant $k$ and constant $\Delta$, our algorithm is asymptotically optimal in running time. The asymptotic optimality of the running time $O(n)$ follows from the fact that we can place one robot per round in the single door case and $n$ robots must be placed.

A summary of these previous results and a comparison to our contribution is presented in Table 1.

| Requirements of Filling algorithms | | | | |
|---|---|---|---|---|
| Method | Visibility range (hops) | Communication range (hops) | Memory (bits) | Graph Type |
| DFLF [12] | 2 | 2 | 2 | Arbitrary |
| TALK [5] | 2 | 2 | 4 | Orthogonal |
| MUTE [5] | 6 | 0 | 9 | Orthogonal |
| $k$-Door in [4] | 2 | 0 | $O(1)$ | Orthogonal |
| Filling in [11] | 1 | 0 | 13 | Orthogonal |
| $k$-Door in [11] | 1 | 0 | 13 | Orthogonal |
| Here: **VCM** | 1 | 0 | $O(\Delta)$ | Orthogonal |
| **MD-VCM** | 1 | 0 | $O(\Delta \cdot \log k)$ | Arbitrary |

**Table 1.** Summary of the requirements of different Filling algorithms. All of these algorithms have $O(n)$ running time in the synchronous model.

**Organization:** In Section 2 we define our model. In Section 3 we describe the Virtual Chain Method for filling a connected regions represented by an arbitrary

graph. Section 4 extends our algorithm to solve the k-door filling. Finally, Section 5 summarizes the paper.

## 2 Model

We are given an unknown, connected area, represented by a graph. Each vertex of the graph allows only one robot to occupy it at any given time. We assume that for each vertex the adjacent vertices are arranged in a fixed cyclic order, which does not change during the dispersion. The entry points of the graph are called Doors. For simplicity we assume the degree of the Door vertices are 1. Otherwise, we introduce an auxiliary vertex of Degree 1 connected only to the Door, which takes the role of the original Door. This models the two side of a doorstep.

In our model we use common concepts in distributed mobile robotics. For an excellent overview we refer to the book by Flocchini et al. [10].

Each robot has a sensor allowing it to gather information from its vicinity, a computational unit, and locomotion capabilities. They are *autonomous*, i.e. no central coordination is present, *homogeneous*, i.e. all the robots have the same capabilities and behaviors, and *anonymous*, i.e. they cannot distinguish each other. They have a visibility range of 1 hop, i.e. each robot can 'see' only the vertex it occupies and the vertices adjacent to it. The robots are *silent*, i.e. they cannot communicate at all. They have limited bits of persistent memory, which is $O(\Delta)$ bits in the single door case and $O(\Delta \cdot \log k)$ when $k$-doors are present.

The robots operate corresponding to the Look-Compute-Move (LCM) model. During the Look phase, the robots take a snapshot of their surroundings. In the Compute phase, based on the snapshot they decide whether to stay idle or to move to one of its neighboring vertices, and during the Move phase they move there. The movement is atomic between two vertices, meaning it is either performed and the robot appears at the destination vertex, or does not move at all. We use the FSYNC model, where the robots perform their LCM cycles at the same time, i.e. each robot takes snapshots, computes, and moves at the same time.

The robots are placed on a predefined vertex, which is called the Door. At the beginning of each cycle, if the Door is empty, a new robot is placed there and performs its Look-Compute-Move phases during the same cycle.

## 3 Virtual Chain Method

We now present the Virtual Chain Method (VCM) for the single door case which is based on the traditional follow-the-leader principle. This principle has also been used in [4, 5, 12, 11]. One robot becomes a leader and the rest of the robots follow it until the leader is blocked, then another robot takes the leadership. During our dispersion algorithm the robots create a virtual chain and move along it. The method mimics a depth first search like exploration of the area. An example for the dispersion can be seen on Fig. 2.

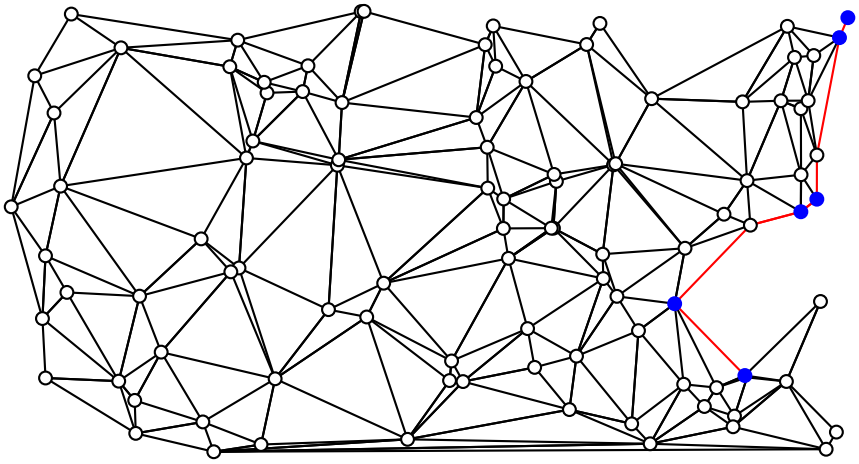

**Fig. 2.** The robots enter through one vertex called the Door (top right vertex), and follow the leader. The vertices occupied by Followers are blue, while the vertex occupied by the Leader is red. The red line denotes the path of the leader.

### 3.1 Concept

In the Virtual Chain Method the following states are permitted to the robots:

- *None*: starting state immediately after the robot is placed at the Door.
- *Leader*: the first robot placed at the Door switches to Leader state. Only the leader moves to previously unoccupied, so called *unvisited* vertices. We will ensure that there can only be at most one leader at a time.
- *Follower*: the robots following their predecessor are in Follower state. A follower can promote itself to leader, when its predecessor is in Finished state.
- *Finished*: the final state of the robots. If a robot detects it cannot move anymore it switches to this state. A Finished robot can never move again. Only the leader can switch to Finished state.

The chain is defined by the path of the current leader from the Door. The followers are following that path, other robots in the area are already in Finished state.

The chain is not visible nor can be detected by the robots; their successor or predecessor might not even be in their visibility range at certain times. To avoid breaking the chain, each robot must follow its predecessor. To ensure the robots can detect their predecessor they are only allowed to move after their successor arrived to their previous vertex. That previous vertex on the chain is called the *entry vertex* of that robot. This way a robot and its predecessor can never be farther than two hops. Only the Leader is allowed to move to vertices, which were never occupied. These vertices are called *unvisited* vertices. When the Leader cannot move anymore, either because its vertex does not have any

neighbors (other than its entry) or the movement would result in a collision, the Leader switches to Finished state and the leadership will be taken by its successor. Therefore, there can be at most one Leader at a time during the dispersion. The algorithm terminates when each robot is in Finished state. The rules followed by the robots can be seen on Algorithm 1.

**Algorithm 1** (**VCM**): Rules followed by robot $r$.

1: If $r.State$ is None:
   $r$ promotes itself to Leader if $r$ has no neighbors, or to Follower otherwise
2: If $r.State$ is Follower:
   If $r.Predecessor$ moved from its place, $r$ follows it.
   If $r.Predecessor$ did not move for two rounds, $r$ switches to Leader state and performs the actions of a Leader in the same round.
3: If $r.State$ is Leader:
   If $r$ is an observer, it stores visited neighbors.
   If $r$ is an observed, it finds the first unvisited neighbor in the cyclical order.
   If $r$ has an unvisited neighbor, it moves there, otherwise it switches to Finished.

**Round Structure** The algorithm operates in rounds. A *round* is a sequence of $\Delta$ consecutive *steps* $S_1 \dots S_\Delta$ (a step is an LCM cycle of the robots). The rounds and the steps are illustrated in Fig. 3. During each round each robot is either an *observer* or an *observed* robot. The observed robot has to schedule its movement, and can move to one of its neighboring vertex $v_i$, which is the $i^{th}$ vertex in the fixed cyclic order starting from the entry vertex of the robot. It can only perform this movement in step $s_i$ (as in Fig. 4), and not allowed to move more than once in each round. The observer robot counts the number of steps its predecessor has waited before moving. At the end of each round the robots switch their roles (i.e. the observed robots become observers and vice versa). Each robot starts as an observer when it is placed at the Door. The robots also store the occupancy information of the neighboring vertices in order to determine which of them are unvisited vertices. Now we describe the exact behavior of the robots in the different states:

**Finished:** Robots that are in Finished state have terminated their actions and do not move anymore.

**Leader:** If robot $r$ is a Leader it leads the chain and moves to unvisited vertices. Thus, the number of unvisited vertices is monotonically decreasing. Robot $r$ can move to vertex $v_i$ in step $s_i$ if and only if $v_i$ is an unvisited vertex. In its observer round $r$ registers each neighbor which is occupied in any step of the round. In its observed round, $r$ moves to only those vertices which had not been registered as occupied in the current or previous round (later we show, it

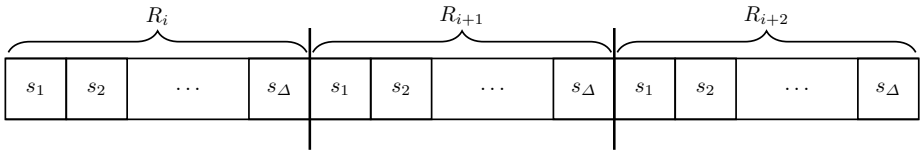


**Fig. 3.** The structure of the rounds. Three rounds (denoted by $R_i$,$R_{i+1}$, and $R_{i+2}$), each consists of $\Delta$ consecutive steps ($s_j$).

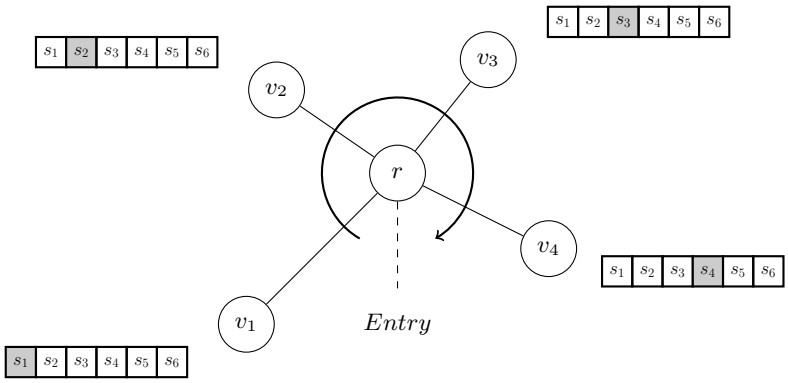


**Fig. 4.** Robot $r$, entered from below, with several neighbors. It can only move to $v_i$ in $s_i$, where $v_i$ is the $i^{th}$ vertex in the cyclic order starting from the entry vertex of $r$.

is sufficient to determine that $v_i$ is an unvisited vertex). If no such vertices are available, $r$ switches to Finished state.

**Follower:** Each Follower robot has to know which robot to follow, i.e. which robot is its predecessor. We will keep the invariant that for each Follower robot $r$ in the beginning of its observer round the predecessor $r'$ of $r$ is in a neighboring vertex and $r'$ moves in that round. We will ensure that $r$ is able to determine the next position of $r'$ based on that in which step $r'$ moves away (similarly to the Leader, $r'$ has to move to $v_i$ in $s_i$). In the next observed round of $r$, $r$ moves to the previous position of $r'$ in step $s_j$, where $j$ is the $j^{th}$ neighbor in the cyclic order from the previous position of $r$.

There are two special cases. First, if $r'$ moves in $s_\Delta$, $r$ will only notice it in $s_1$ of its next observed round, which will be taken into account. The second special case, if $r'$ cannot move anymore, which can only happen if $r'$ is the leader and do not have any more unvisited neighboring vertices, $r$ will notice it in $s_1$ of its next observed round. In this case $r$ switches to Leader state and acts accordingly in the same round (i.e. $r$ moves to an unvisited neighbor, or switches to Finished

state if there is no unvisited neighboring vertex). For this reason Followers have to maintain unvisited vertices around them.

**None:** The robot $r$ placed at the Door is initialized with None state. In this state the robot does not move and waits for state transition. This transition can be one of the following: if the neighboring vertex $v$ of $r$ is occupied then $r$ becomes a Follower of the robot on $v$, otherwise it becomes a Leader. Before this state transition the robot must perform the following tasks.

The first task to solve is how the robots know which step they are in. The first robot is placed in the area in step $s_1$ of the first round. However, if that robot would move from the Door, the next one would be placed in $s_2$. Based on its surroundings, the new robot cannot acquire this information and it should not be provided explicitly when it is placed. If the robots are only allowed to move from the Door in $s_\Delta$ the new robot is always placed in $s_1$ of the next round. This can easily be achieved by not letting the robots in None state to switch states before $s_\Delta$. (Note: the robot at the Door does not have a successor yet, therefore, it does not have to schedule its movement.)

The second task for robots at the Door is to ensure their role (observer/observed) in each round. When robot $r$ moves from the Door in round $R_i$ a new robot $r'$ is placed at the Door immediately. The robot $r'$ becomes active in step $s_1$ of round $R_{i+1}$, which is the observer round of $r$. As $r'$ is initialized as an observer they would be both observers in $R_{i+1}$. To achieve distinct roles, $r'$ stays inactive in $R_{i+1}$ and becomes an observer in $R_{i+2}$ again (and $r'$ observes $r$ moving in that round).

### 3.2 Analysis

**Lemma 1.** *The predecessor of the Follower is either in a neighboring vertex $v$ or it was in $v$ in the previous round. In the latter case the Follower moves to the previous position of the predecessor, which is $v$.*

*Proof.* Assume $r$ follows its predecessor $r'$. Let $R_i$ be the round when $r$ observes $r'$ from the Door ($r$ was placed in the previous round). In $R_i$ $r'$ moves to one of its neighbors. In $R_{i+1}$ $r$ moves to the vertex which is left by $r'$. After this they become neighbors again. This argument can also be repeated for the following rounds. Therefore the claim of the lemma holds. □

**Lemma 2.** *Each vertex can be considered an unvisited vertex if it is not occupied in two consecutive rounds.*

*Proof.* There are three cases regarding vertex $v$: *i*) it is occupied, *ii*) it is unoccupied, but has been occupied before, or *iii*) it is an unvisited vertex (unoccupied, and never been occupied before). In case *i*), any robot observing $v$ knows it is not an unvisited vertex. In case *ii*), let $r$ be the last robot that occupied $v$, and let $R_i$ be the round in which $r$ moved from $v$. According to Lemma 1 the successor of $r$ moves to $v$ in round $R_{i+1}$. Consequently, $v$ will not be unoccupied through two consecutive rounds. Therefore, observing $v$ for two consecutive rounds allows any robot to identify cases *i*) and *ii*). In any other case vertex $v$ is an unvisited vertex, i.e. case *iii*) holds. □

**Lemma 3.** *Each robot in Follower state always knows where its predecessor is.*

*Proof.* Using induction, we show that after movement $i \geq 0$ a Follower $r$ knows where its predecessor $r'$ is.

**Induction start**: We show that after movement 0 (i.e. before its first movement), $r$ knows where $r'$ is.

Movement 0 means $r$ did not move yet, and is still at the Door. Let $v$ be its only neighboring vertex. Assume a robot is occupying $v$, as $r$ is in Follower state (if $v$ had not been occupied, $r$ would became a Leader). We show that the robot on $v$ must be predecessor $r'$ of $r$.

The robots placed at the Door can only move to $v$. After this movement a new robot is placed at the Door, whose predecessor is the robot which has moved to $v$. If any other robot had moved to $v$, a new robot would not been placed at the Door. Therefore, if $r$ is placed at the Door its predecessor is on $v$.

**Inductive step**: We show, if $r$ knows where its predecessor is after movement $i$, it will know where it is after movement $i + 1$.

After movement $i$ robots $r$ and its predecessor $r'$ are in two neighboring vertices ($v$ and $v'$) as $r$ just moved after $r'$. Therefore, $r$ knows which robot is to observe. If $r'$ moves, it can only move during an observed round while $r$ is observing it. While observing $r$ counts the steps until $r'$ move to its next vertex $v''$. As assumed, the neighbors of $v'$ are in a fixed cyclic order, $r$ will know which is the next vertex $v''$ when it moves to $v'$ (after movement $i + 1$) and the robot occupying $v''$ will be $r'$. □

**Lemma 4.** *Two Followers cannot have the same predecessor.*

*Proof.* When a robot $r$ moves from the Door, only one robot will be placed at a time, therefore, only one robot can Follow $r$ (and choose it as its predecessor). According to Lemma 3. each Follower knows where its predecessor is after every movement and will not change predecessors. As a result $r$ will never be the predecessor of two (or more) different robots. □

**Lemma 5.** *The Leader only moves to unvisited vertices.*

*Proof.* As stated in Lemma 2. it is enough to observe each neighboring vertex for two consecutive rounds to determine whether it is an unvisited vertex or not. Assume $R_j$ is the observer round of $r$ and $R_{j+1}$ is the observed round. In $R_{j+1}$, in which $r$ moves, it already knows which neighbor is unvisited and – if any – it moves to the first unvisited vertex in the cyclic order of the neighbors in the corresponding step of $R_{j+1}$. □

**Lemma 6.** *There can be at most one Leader at a time.*

*Proof.* The first robot placed at the Door becomes the Leader. Afterwards, the Leader only moves to unvisited vertices. If there is no unvisited neighboring vertex (this can be determined in two rounds), the Leader changes its state to Finished. After changing the state its successor becomes a new Leader, the

number of Leaders is still one. The condition of becoming a new Leader is that the old Leader did not move for two consecutive rounds, which implies it has changed to Finished state (otherwise, the old leader would have moved to an unvisited vertex).

**Lemma 7.** *No collision can occur during the dispersion.*

*Proof.* The robots in None and Finished states are not allowed to move, therefore, it only has to be shown that robots in Follower and Leader state cannot collide with other robots.

The robots in Follower state are following their predecessor and, according to Lemma 1, if they are not in neighboring vertices, the successor can only move to the previous position of the predecessor. As two robots cannot have the same predecessor (see Lemma 4), it is not possible that multiple Followers would move to the same vertex at the same time.

The Leader can only move to unvisited vertices (see Lemma 5) and Followers can only move to the position of its predecessor (i.e. not unvisited vertices), and as only one Leader is in the system (see Lemma 6), it cannot collide with other robots.

Therefore, no collision can occur. □

**Lemma 8.** *Algorithm VCM fills the area (represented by the graph).*

*Proof.* By contradiction, assume some vertices left unoccupied after the algorithm terminated (meaning each robot is in Finished state). Since the graph is connected and the Door is occupied all of the time, there exists an unoccupied vertex $v$ having at least one occupied neighbor $v'$. Let $r$ be the robot on $v'$, which is in Finished state. The condition for a robot to switch to Finished state is to be a Leader and not to have unvisited neighbors. However, according to Lemma 2, $v$ will be identified as an unvisited vertex by $r$ and $r$ can occupy it. This contradicts the assumption that $v$ remains unoccupied. □

**Theorem 1.** *By algorithm VCM an area (represented by a connected graph) with a single door is filled in $O(\Delta \cdot n)$ rounds without collisions by robots with visibility range of 1 hop and $O(\Delta)$ bits of persistent memory.*

*Proof.* After placing the robot at the Door it is in None state, in which the robot *skips* one round. In the next round it observes its predecessor moving, then it moves in the third round. In the same round the next robot will be placed at the Door. For this reason, the robots are placed in every third round at the Door. As each round consists of $\Delta$ steps it takes $3 \cdot \Delta \cdot n = O(\Delta \cdot n)$ steps to place $n$ robots.

Regarding the memory requirement, the robots require to store the index of the current step within a round, the unvisited neighbors, the direction of their predecessor known from the index in which step it moved away (each requiring at most $\Delta$ bits of memory), and some additional information, requiring constant amount of bits: current state, observer/observed role, entry vertex. As a result, $O(\Delta)$ bits of persistent memory is required for the Virtual Chain Method. □

## 4 Multiple Doors

In the Multiple Door Filling, the biggest challenge is how the robots entering from different Doors avoid collisions. This is usually defined by some sort of priority order, e.g. in [4], the robots had 2 hops visibility and $k$ different externally visible colors, where $k > 1$ is the number of Doors. In [11] the cells could be entered form different directions in each step.

In the Multiple Door Virtual Chain Method, (MD-VCM) the robots from each distinct Door will form a distinct chain, which are lead by their Leader robot. For each Door (for each chain) we introduce distinct time-slots, in which they can perform their actions. As opposed to the single door case, each step is substituted by $k$ steps. The new round consists of $k \cdot \Delta$ steps. Step $s_{i,j}$ within a round corresponds to the $j^{th}$ step of the chain originating from the $i^{th}$ Door $D_i$, $1 \leq i \leq k$, $1 \leq j \leq \Delta$. Each robot from $D_i$ only performs their actions in $s_{i,*}$.

**Rules for the k-Doors case:** The robots in Finished, Leader, and Follower state follow the same algorithm as in the original VCM. The actions of the robots in None state has to be modified, as they are not necessarily placed in first step of the corresponding chain, e.g., if $i < k$, the robots entering from Door $D_i$ will move from it in $s_{i,\Delta}$, resulting the new robot to be placed at $D_i$ in step $s_{i+1,\Delta}$ (which is the time-slot of the next Door). To make sure the new robot starts their actions in step $s_{1,1}$ of the next round the newly placed robot stays inactive for $k - i$ steps. The only exceptions to this rule are the robots which has been initially placed in the first round $R_1$. They know they are the first robots in that Door if there are no neighboring robots for two rounds. In this case they become active immediately (not skipping $k - i$ steps) and switch to Leader state in step $s_{i,\Delta}$ of the second round.

Note: on a rare occasion it is possible two Doors are neighbors with eachother. Since we assumed that the degree of each Door vertex is 1 and the graph is connected, it is only possible for $n = 2$. In this very special case, the robots at the Door do not know correctly in which step they were placed there. However, the filling problem is solved right after both robots are placed.

### 4.1 Analysis

In the Multiple Door case Lemma 1, 2, and 3 still hold. In case of Lemma 4, 5 it still has to be considered that there are multiple chains in the area.

**Lemma 9.** *In the MD-VCM, each Leader only moves to unvisited vertices and cannot collide with each other.*

*Proof.* Unlike in Lemma 5 there can be $k > 1$ robots in the system with Leader state, meaning that multiple robots can choose the same unvisited vertex as their destination. However, different leaders (which are from different Doors) are assigned to different steps. As a result, one of them moves there first, after which the vertex cannot be considered an unvisited one and the other Leaders have to choose a new destination.

**Lemma 10.** *The MD-VCM distinct chains cannot cross each other.*

*Proof.* The vertices on the paths of the leaders (the chains) are not unvisited vertices, since they were already occupied by the leader. Unvisited vertices are detected by the leaders according to Lemma 2. Because of Lemma 5 the Leaders only move to unvisited vertices. Consequently, the leaders do not cross other chains. Each Follower only moves on the path of the Leader of the chain, chains originating from different Doors are disjoint.

**Theorem 2.** *An area (represented by a connected graph) with a multiple doors is filled by the MD-VCM in $O(k \cdot \Delta \cdot n)$ rounds without collisions by robots with visibility range of 1 hop and $O(\Delta \cdot \log k)$ bits of persistent memory.*

*Proof.* Similarly to Theorem 1 the robots are placed in each Door in every three rounds if the chain from that Door is able to move, otherwise no further robots can be placed at that Door anymore. In the worst case, when a single chain blocks all the other Doors, a Door is used to cover the area. Consider a graph which is a simple chain of $n$ vertices whose first $k$ vertices are the Doors $D_1 \dots D_k$. In this case only the robots placed on $D_k$ can move, the robot in the other Doors are blocked. This is equivalent to the single door case, in which the running time is $3n$ rounds, where each round length consists of $k \cdot \Delta$ steps, yielding $O(k \cdot \Delta \cdot n)$ runtime.

Regarding the hardware requirements, the robots do not need additional visibility, nor other equipment. As the round length increases, and the current step index has to be stored, the memory increases to $O(\Delta \cdot \log k)$. □

## 5 Summary

In this paper, we have presented the Virtual Chain Method for solving the filling problem in unknown region, represented by an arbitrary connected graph, with autonomous robots having a minimum visibility range of 1 hop. The robots are not equipped with communication capabilities, they are working synchronously. We only assumed that the neighbors of each vertex are arranged in a fixed cyclic order, which is the same for each robot stepping into that vertex. For the single door case, the robots need $O(\Delta)$ bits of persistent memory, and $O(\Delta \cdot n)$ time steps, where $n$ is the number of vertices of the graph. We have extended this method to solve the $k$-door filling problem in $O(k \cdot \Delta \cdot n)$ time steps for robots with memory requirement $O(\Delta \cdot \log k)$. The Multiple Door Virtual Chain Method does not add further hardware requirements for the robots. It remains an open question how the multiplicative factor $k$ can be eliminated in the running time.

## Acknowledgment

This work was partly performed in the frame of FIEK_16-1-2016-0007 project, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 funding scheme.

## References


1. Albers, Kursawe, Schuierer: Exploring unknown environments with obstacles. Algorithmica **32**(1), 123–143 (2002)
2. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM J. Comput. **29**(4), 1164–1188 (2000)
3. Augustine, J., Moses, Jr., W.K.: Dispersion of mobile robots: A study of memory-time trade-offs. In: Proc. 19th International Conference on Distributed Computing and Networking (ICDCN '18). pp. 1:1–1:10 (2018)
4. Barrameda, E.M., Das, S., Santoro, N.: Deployment of asynchronous robotic sensors in unknown orthogonal environments. In: Algorithmic Aspects of Wireless Sensor Networks: 4th Int. Workshop (ALGOSENSORS 2008), Revised Selected Papers. pp. 125–140 (2008)
5. Barrameda, E.M., Das, S., Santoro, N.: Uniform dispersal of asynchronous finite-state mobile robots in presence of holes. In: Algorithms for Sensor Systems - 9th Int. Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS 2013), Revised Selected Papers. pp. 228–243 (2014)
6. Brass, P., Cabrera-Mora, F., Gasparri, A., Xiao, J.: Multirobot tree and graph exploration. IEEE Transactions on Robotics **27**(4), 707–717 (2011)
7. Bullo, F., Cortés, J., Marttınez, S.: Distributed algorithms for robotic networks. Applied Mathematics Series, Princeton University Press (2009)
8. Cohen, R., Peleg, D.: Local spreading algorithms for autonomous robot systems. Theoretical Computer Science **399**(1), 71 – 82 (2008)
9. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theoretical Computer Science" **609**, 171–184 (2016)
10. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2012)
11. Hideg, A., Lukovszki, T.: Uniform dispersal of robots with minimum visibility range. In: Algorithms for Sensor Systems - 13th Int. Symp. on Algorithms and Experiments for Wireless Networks (ALGOSENSORS 2017), Revised Selected Papers. pp. 155–167 (2017)
12. Hsiang, T., Arkin, E.M., Bender, M.A., Fekete, S.P., Mitchell, J.S.B.: Algorithms for rapidly dispersing robot swarms in unknown environments. Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics **7**, 77–93 (2004)