Asynchronous Filling by Myopic Luminous Robots*

Attila Hideg[1] and Tamás Lukovszki[2]

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

Abstract. We consider the problem of filling an unknown area represented by an arbitrary connected graph of n vertices by mobile luminous robots. In this problem, the robots enter the graph one-by-one through a specific vertex, called the Door, and they eventually have to cover all vertices of the graph while avoiding collisions. The robots are anonymous and make decisions driven by the same local rule of behavior. They have limited persistent memory and limited visibility range. We investigate the Filling problem in the asynchronous model.
We assume that the robots know an upper bound Δ on the maximum degree of the graph before entering. We present an algorithm solving the asynchronous Filling problem with robots having 1 hop visibility range, $O(\log \Delta)$ bits of persistent storage, and $\Delta + 4$ colors, including the color when the light is off. We analyze the algorithm in terms of asynchronous rounds, where a round means the smallest time interval in which each robot, which has not yet finished the algorithm, has been activated at least once. We show that this algorithm needs $O(n^2)$ asynchronous rounds. Our analysis provides the first asymptotic upper bound on the running time in terms of asynchronous rounds.
Then we show how the number of colors can be reduced to $O(1)$ at the cost of the running time. The algorithm with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $O(1)$ colors needs $O(n^2 \log \Delta)$ rounds. We show how the running time can be improved by robots with a visibility range of 2 hops, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 4$ colors (including the color when the light is off). We show that the algorithm needs $O(n)$ asynchronous rounds. Finally, we show how to extend our solution to the k-Door case, $k \geq 2$, by using $\Delta + k + 4$ colors, including the color when the light is off.

1 Introduction

In swarm robotics, a large number of autonomous mobile robots cooperate to achieve a complex goal. The robots of the swarm are simple, cheap, and compu-

* The research has been partially supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

tationally limited. They act according to local rules of behavior. Robot swarms can achieve high scalability, fault tolerance, and cost-efficiency.

The robots can cooperatively solve different problems, as gathering, flocking, pattern formation, dispersing, filling, coverage, and exploration (e.g. [1, 3–6, 9, 12, 19, 21].

The Filling (or, Uniform Dispersal) problem was introduced by Hsiang et al. [19], where the robots enter an a priori unknown but connected area and have to disperse. The area is subdivided into pixels, and at the end of the dispersion, each pixel has to be occupied by exactly one robot.

Model: We consider the Filling problem, where the area is represented by a connected graph, which is unknown for the robots. The robots enter the graph one-by-one through a specific vertex, which is called the *Door* and have to disperse to cover all vertices of the graph while avoiding collision, i.e. two or more robots can not be at the same vertex. At the end of the dispersion, each vertex of the graph has to be occupied by exactly one robot. When the Door vertex becomes empty, a new robot is placed there immediately. We assume that the robots know an upper bound Δ on the maximum degree of the graph.

For simplicity, we assume that the degree of the Door vertex is 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 sides of a doorstep). We assume that, for each vertex v, the adjacent vertices are arranged in a fixed cyclic order. This cyclic order is only visible for robots at v, and it does not change during the dispersion. When a robot r arrives at vertex v from a vertex u, then the cyclic order of neighbors is used by r as a linear order of $\deg(v) - 1$ neighbors by cutting and removing u.

The robots act according to the Look-Compute-Move (LCM) model. In this model, their actions are decomposed into three phases: In the *Look* phase, a robot takes a snapshot of its surroundings, i.e. the vertices and the robots within its visibility range. In the *Compute* phase, it performs calculations based on the surrounding and determines a neighboring vertex as target vertex, or decide to stay at place. In the *Move* phase, if necessary, it moves to the target vertex. The next LCM cycle starts when the target is reached.

Based on the activation times of the robots, there are three main synchronization models studied in the literature: the fully synchronous (FSYNC), the semi-synchronous (SSYNC), and the asynchronous (ASYNC). In the FSYNC model, all robots are activated at the same time, and they perform their Look, Compute, and Move phases synchronously at the same time, which is ensured by a global clock. In the SSYNC model, some robots might skip an LCM cycle and stay inactive. In the ASYNC model, there is no common notion of time available: the robots activate independently after a finite but arbitrary long time, and perform their LCM cycles. Moreover, their LCM cycle length is not fixed; it also can be arbitrarily long.

The robots are *autonomous*, i.e. no central coordination is present, *homogeneous*, i.e. all the robots have the same capabilities and behaviors, *anonymous*, i.e. they cannot distinguish each other, *myopic*, i.e. they have limited visibility range, and *silent*, i.e. they have no communication capabilities and cannot directly talk to one another. However, *luminous* robots can communicate indirectly by using a light. Such robots have a light attached to them, which is externally visible by every robot in their visibility range. They can use a finite set of colors (including the color when the light is off) representing the value of a state variable. The robots are allowed to change these colors in their Compute phase. We denote by X^i the model $X \in \{$ASYNC, SSYNC, FSYNC$\}$ when every robot is enhanced by a light with $i > 1$ colors. In the ASYNC$^{O(1)}$ model, the robots use a constant number of colors (see, e.g. [10]).

Related Work: The Filling (or, Uniform Dispersal) problem was introduced by Hsiang et al. [19], where the robots enter an unknown but connected orthogonal area and have to disperse. The area is subdivided into pixels, and at the end of the dispersion, each pixel has to be occupied by exactly one robot. Hsiang et al. [19] considered this problem in the FSYNC model. They assumed that robots have a limited ability to communicate with nearby robots. They proposed two solutions, BFLF and DFLF, both modeling generally known algorithms: BFS and DFS. DFLF required a visibility range of 2 hops. It was assumed that the robots are able to detect the orientation of each other. Barrameda et al. [5, 6] investigated the asynchronous case. In [5] the authors assumed common top-down and left-right directions for the robots and showed that robots with visibility range of 1 hop and 2 bits of persistent memory could solve the problem in an orthogonal area if the area does not contain holes, without using explicit communication in finite time. In [6] Barrameda et al. presented two methods for filling an unknown orthogonal area in presence of obstacles (holes) in the ASYNC model. Their first method, called TALK, requires a visibility range of 2 hops[3] if the robots have explicit communication. The other method, called MUTE, does not use explicit communication between the robots, but it requires a visibility range of 6. Both methods need $O(1)$ bits of persistent memory and terminate in finite time.

In [16, 18] the Filling problem has been investigated in the FSYNC model. In [16] the authors gave a solution for the orthogonal Filling problem by using robots with 1 hop visibility range and $O(1)$ bits of persistent memory for both the Single and Multiple Door cases. In [18] a method for a general Filling problem has been presented, where the area is represented by an arbitrary connected graph. The robots require 1 hop visibility range and $O(\Delta)$ bits of persistent memory, where Δ is the degree of the graph. For the k-Door case, the memory requirement is $O(\Delta \cdot \log k)$. The general method is called the Virtual Chain Method (VCM), which is a leader-follower method. In the VCM, the robots form a chain and fill the area mimicking a DFS-like traversal of the graph. The algorithms presented in [16] and [18] are intensively utilizing the synchronous nature of the model to avoid collisions and backtracking.

[3] In [6] it is assumed that the robot sees all eight sourrounding cells and able to communicate with robots at that eight cells. Assuming orthogonal movements, a cell sharing only one corner with the current cell of the robot are reachable in two hops.

The model of luminous robots was introduced by Peleg [24]. Subsequently, significant amount of research has been carried for a plenty of problems using this model (e.g. [2, 7, 8, 14, 15, 20, 22, 23, 26, 25, 27]). Das et al. [10, 11] considered the model, where the robots can move in the continuous Euclidean plane, and they proved that the asynchronous model with a constant number of colors $\text{ASYNC}^{O(1)}$ is strictly more powerful than the semi-synchronous model SSYNC, i.e. $\text{ASYNC}^{O(1)} > \text{SSYNC}$. Das et al. [11] also prove that there are problems that robots cannot solve without lights, even if they are fully synchronous, but can be solved by asynchronous luminous robots with $O(1)$ colors.

D'Emidio et al. [13] have shown that on graphs one task can be solved in the fully synchronous model FSYNC but not in the asynchronous lights-enhanced model, while for other tasks, the converse holds. In this work, we show that the Filling problem can be solved in both models by robots with 1 hop visibility range and $O(1)$ bits of persistent memory.

Our Contribution: In this work, we present solutions for the Filling problem by luminous robots on graphs in the $\text{ASYNC}^{O(1)}$ model.

First, we describe a method, called PACK, which solves the problem by robots with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $\Delta+4$ colors for the single Door case, including the color when the light is off. We analyze the algorithm in terms of asynchronous rounds, where a round means the smallest time interval in which each robot, which has not yet finished the algorithm, has been activated at least once. We show that this algorithm needs $O(n^2)$ asynchronous rounds. Regarding asynchronous algorithms for the Filling problem, former works only guarantee termination within finite time. Our analysis provides the first asymptotic upper bound on the running time in terms of asynchronous rounds.

Then we show how the number of colors can be reduced to $O(1)$ at the cost of running time. The algorithm with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $O(1)$ colors needs $O(n^2 \log \Delta)$ rounds.

After this, we show how the running time can be significantly improved by robots with a visibility range of 2 hops, with no communication, $O(\log \Delta)$ bits of persistent memory, and $\Delta+4$ colors, by presenting the algorithm called BLOCK. This algorithm needs $O(n)$ rounds.

Then we extend the BLOCK algorithm for solving the k-Door Filling problem, $k \geq 2$, by using $O(\log \Delta)$ bits of memory and $\Delta + k + 4$ colors, including the color when the light is off. The visibility range of 2 hops is optimal for the k-Door case (a counterexample when this problem cannot be solved in the ASYNC model with a visibility range of 1 hop was presented in [5], also holds for the $\text{ASYNC}^{O(1)}$ model).

2 PACK Algorithm

Now we describe the PACK algorithm to solve the Filling problem for an area represented by a connected graph of n vertices. PACK is based on the Virtual

Chain Method described in [18], in which the robots filled the area in a DFS-like dispersion.

The robots are allowed to be in one of the following states: None, Follower, Leader, Finished. They are initialized with None state when placed at the Door. The first robot becomes the Leader and moves to a vertex that has never been occupied before (these vertices are called *unvisited* vertices). The rest of the robots will become Followers and follow the Leader, until the Leader becomes stuck (i.e. no unvisited neighbors available). Then the robot behind the Leader, called the *successor* robot, becomes a new Leader and moves if possible. The previous Leader switches to Finished state. The algorithm terminates when each robot is in Finished state.

The name Virtual Chain comes from the fact that all active robots (i.e. the Leader and the Followers) are on the path traversed by the current Leader from the Door. This path is called the chain. The chain contains only visited vertices, which can be occupied by the Followers. Each Follower follows its *predecessor*, which is the previously placed robot.

The difficulty is to select the next target vertex for the Leader with a visibility range of 1 hop by ensuring that no other robot can move to that vertex because the Leader can not see all adjacent vertices of the target and robots on those vertices do not see the Leader.

We define the state *Packed* for the chain. The chain is in Packed state, when each Follower is immediately behind its predecessor, i.e. each vertex on the path traversed by the current Leader from the Door is occupied by a Follower robot. In this state none of the robots can move except the Leader. Therefore, only the Leader has to know this state.

The concept: The Leader moves to unvisited vertices until there is no such neighboring vertex. Before each movement, the Leader waits for Packed state; thus, it cannot collide with other robots, and the Leader can decide which vertex is unvisited. When the Leader has no neighboring unvisited vertex, it switches to Finished state and does not move anymore. Its successor then becomes the Leader and the new Leader moves to other unvisited vertices. The robots use $\Delta+4$ colors, including the color, when the light is off. The first Δ colors show the direction of the target vertex (for each vertex, the adjacent vertices are arranged in a fixed cyclic order), we refer to them as *DIR* colors. Furthermore, we use two colors, denoted by *CONF* and *CONF2* colors, for confirming that a robot has seen a DIR color of the predecessor, which allows the predecessor to move. For this purpose, the CONF color is sufficient, when the predecessor is a Follower robot. However, when the predecessor robot is the Leader, and it must change the target vertex after the Packed state is reached (details are provided later) or the predecessor becomes the Leader and it chooses an unvisited target vertex, it indicates the new direction with a new DIR color. Then the CONF2 color is needed for ensuring that the successor has seen the lastly shown DIR color. Furthermore, we use an additional color, called *MOV* color to indicate that a robot is on the way to its target vertex.

Now we describe the rules followed by the robots in different states.

Leader: Can only move to an unvisited vertex. When it wants to move, it shows the direction it wants to go to by setting the corresponding DIR color, and then it waits until its successor gives a confirmation that it can move by setting its CONF color. During the movement, the Leader shows the MOV color. When its successor sets its CONF color, the chain is in Packed state. This means each not occupied vertex is also an unvisited vertex (as each vertex in the path of the Leader is occupied by a robot). If the Leader is still on the Door vertex, therefore, it does not have a successor, it can move without waiting for the CONF color.

Follower: Follows its predecessor. The Follower robot r sets the CONF color if and only if i) the predecessor of r is showing its direction, and ii) the successor of r – if exists – have set its CONF color (i.e. the successor knows in which direction r will move). This allows the predecessor r' of r to move to its destination knowing: i) all the robots behind r' have set CONF color, and ii) the robots behind r' will not move until the predecessor of r moved. When r' is the Leader, the chain is in Packed state.

None: The robots are initialized with None state when they are placed at the Door. If the robot r in None state has no neighboring robot, then r changes its state to Leader, chooses the unique neighboring vertex as target vertex, sets the MOV color, and starts moving there. Otherwise, if the robot r in None state has one neighboring robot, then r becomes a Follower and sets the neighbor to its predecessor.

There are three special situations where we need the following additional rules:

Leader target change: It might happen that the Leader r chooses a target vertex v, which is unoccupied at the moment when r performs its Look operation, however, when the successor of r sets the CONF color and r could start to move to v, another robot already moved to v. In such case, the Leader r has to choose a new target, and the successor of r has to know about this choice. Assume first that r has an unvisited neighboring vertex. Then r sets the corresponding DIR color and waits until its successor sets the CONF2 color. Finally, the Leader moves to the target.

Note that the chain is in Packed state when the successor of the Leader r sets the CONF color. If the Leader changes the target, no other robot can move until r sets the CONF2 color, and the Leader moves to the target. Consequently, the Leader can change the target vertex only once between two movements.

If r does not have any unvisited neighboring vertex after r sees the CONF color of the successor, then r can not move anymore and the successor must take the leadership (see the rule below). The robot r sets the Δ direction color, which has special meaning. The successor r' confirms this by setting the CONF2 color. Then r turn off its light r' becomes the Leader. (Note that it would be possible to omit the Leader target change rule by introducing a new color for signaling the Packed state. Then the Leader would only show its direction once the Packed state is achieved, which could be acknowledged with the CONF color.)

Taking the leadership: When the Leader r cannot move anymore, its successor has to become the new Leader. The Leader r indicates that it does not have any unvisited neighboring vertex by setting its direction color to Δ. I.e. this color has a special meaning: it indicates that the Leader cannot move anymore and wants to switch to Finished state, and the leadership must be taken by its successor. When this is detected by the successor r', it sets its CONF color, waits for the previous Leader to turn off its light, then r' becomes the Leader. Afterwards, r' tries to move to an unvisited vertex.

Setting movement color: Before performing the movement, the robots have to set their color to MOV. Keeping the old color could lead to an error. E.g., consider the following situation. 1. The Leader sets a DIR color. 2: The Follower confirms it by the CONF color. 3: The Leader moves by keeping the DIR color. 4: The Follower shows the corresponding DIR color, receives a CONF, and follows the Leader. 5: The Follower reaches its target, sees the old DIR color of the Leader and sets the CONF color, before the Leader chooses the new target. In order to prevent such situations, the moving robots set their color to MOV and keep this color until the target is reached, and a new target is determined. After the movement, the robot sets the previous position as its *Entry* vertex.

Pseudocode of the PACK algorithm is provided in the full version of this article [17].

2.1 Analysis

The proofs of Lemmata 1 - 4 are available in the full version of the article [17].

Lemma 1. *Leader only moves to unvisited vertices.*

Lemma 2. *There can be at most one Leader at any time.*

Lemma 3. *Robots cannot collide.*

Lemma 4. *PACK fills the area represented by a connected graph.*

Theorem 1. *Algorithm PACK fills an area represented by a connected graph in the ASYNC model by robots having a visibility range of* 1 *hop,* $O(\log \Delta)$ *bits of persistent storage, and* $\Delta + 4$ *colors, including the color when the light is off.*

Proof. As the area is filled (by Lemma 4), and collisions are not possible (by Lemma 3), the area will be filled without collisions. The robots require $O(\log \Delta)$ bits of memory to store the following: $State$ (4 states: 2 bits), $Target$ (direction of the target vertex: $\lceil \log \Delta \rceil$ bits), $NextTarget$ (direction of the vertex, where the robot needs to move after the vertex $Target$ is reached: $\lceil \log \Delta \rceil$ bits). Regarding the number of colors, the robots use Δ colors to show the direction where the target of the robot is. There are two additional colors (CONF and CONF2) for confirming the robot saw the signaled direction of the predecessor and one color (MOV) during the movement.

Now we analyze the running time of the algorithm in terms of asynchronous rounds. An asynchronous round means the shortest time in which each robot, which is not in Finished state yet, has been activated at least once and performed an LCM cycle.

Theorem 2. *The algorithm PACK runs in $O(n^2)$ asynchronous rounds.*

Proof. Assume a chain containing r_1, r_2, ..., r_i (where r_1 is the active Leader, and r_2, ..., r_i are on the path from the Leader to the Door), and assume that the chain is in Packed state.

Assume first that the Leader r_1 has an unoccupied neighboring vertex. Denote by T the time between two consecutive movements of the Leader. We divide T into three time intervals: $T = T_1 + T_2 + T_3$. T_1 starts with the movement of the Leader, it includes the time, when all robots in the chain, making one step forwards. T_2 starts with placing a new robot at the Door. In T_2 the robots, starting from the Door, set their CONF color one by one. This CONF color is 'propagated' to the Leader, meaning that the Packed state is reached. T_3 starts when the Leader recognizes the CONF color of the successor, i.e. after achieving the Packed state. Then the Leader might find its target occupied by another robot. In this case, the *Leader target change* rule will be used.

Let t be the first asynchronous round of T_1, i.e. in round t the Leader r_1 moves to its target and sets its direction color. At the latest in round $t+1$ the robot r_2 detects that r_1 left its previous vertex v and in that round r_2 moves to v and sets its direction color. This argument can be repeated to all robots until the last one r_i moves at the latest in round $t+i-1$. Therefore, $T_1 \leq i$.

Now the second phase T_2 starts with the placing of a new robot r_{i+1} at the latest in the round $t+i$. In that round, the new robot r_{i+1} sets its color to CONF. The robot r_i sees this color at the latest in the next round t_{i+1} and sets its color to CONF in that round. Repeating this argument for $r_{i-1}, \ldots, r_2$, we obtain that r_2 sets the CONF color at the latest in round t_{2i-1}. Therefore, $T_2 \leq i$.

Now T_3 starts. The Leader r_1 recognizes the CONF color of its successor at the latest in round t_{2i}. Then Leader knows that the chain is in Packed state. If the target vertex v of the Leader is unoccupied, the Leader can move immediately, since in Packed state each unoccupied vertex is unvisited. Otherwise, if v is occupied, the *Leader target change* protocol is performed, i.e. 1: the Leader chooses a new unoccupied neighboring vertex and shows the corresponding DIR color (in round t_{2i} at the latest), 2: then its successor sets its color to CONF2 (in round t_{2i+1} at the latest). At the latest in the round t_{2i+2} the Leader recognizes this and can move. Then $T = T_1 + T_2 + T_3 \leq 2i + 2 \leq 2n$ rounds.

Assume now that the Leader r_1 has no unoccupied neighboring vertex. If r_1 is at the Door, then it turns the light off and switches to Finished state; the graph is filled. Otherwise, r_i sees the CONF color of r_2 and recognizes the Packed state in the round t_{2i} at the latest. Then r_1 sets its Δ color in that round. The robot r_2 recognizes it in round t_{2i+1} at the latest and sets its CONF color. The robot r_1 sees the CONF color in round t_{2i+2} at the latest, r_1 turns its light off and switches to Finished state. The robot r_2 sees it in round t_{2i+3} at the latest, r_2 becomes the new Leader in that round, and checks if there is an unoccupied neighboring vertex. If so, r_2 sets the corresponding DIR color in the same round and waits for the CONF2 color from the successor. The successor sets the CONF2 color in round t_{2i+4} at the latest, and r_2 sees the CONF2 color at the latest in round t_{2i+5}. Since the chain is already in Packed state, the robot r_2 can move in the same round (in round t_{2i+5} at the latest). Otherwise, if r_2 has no unoccupied neighboring vertex, then the leadership has to be taken by its successor when the successor exists, i.e. r_2 is not at the Door. If r_2 is at the Door, then r_2 turns the light off and switches to Finished state; the graph is filled.

When a Leader can move, it occupies an unvisited vertex within $2n$ asynchronous rounds. Otherwise, its successor takes the leadership and performs a target change. Taking the leadership and the target change need at most 5 rounds. Since the leadership is taken at most once by each robot during the whole algorithm, and there are n robots in the filled graph, at most $5n$ time is used for all leadership taking with target change altogether. Therefore, after at most $2n^2 + 5n = O(n^2)$ rounds all vertices of the graph become filled.

Remark: In the ASYNC model, a robot can be inactive between two LCM cycles. Since the inactive phase allowed to be finite but arbitrarily long, an asynchronous round and the runtime of the algorithm can also be arbitrarily long. In the case where we do not allow inactive intervals between the LCM cycles and every LCM cycle of every robot takes at most t_{max} time then we can upper bound the time of an asynchronous round by $2 \cdot t_{max}$.

Corollary 1. (i) *Assume that every LCM cycle of every robot takes at most t_{max} time and there are no inactive intervals between the LCM cycles. Then the running time of the PACK algorithm is $O(n^2 t_{max})$.* (ii) *In the FSYNC model the PACK algorithm needs $O(n^2)$ LCM cycles.*

2.2 Filling of graphs using constant number of colors

The PACK algorithm uses $\Delta + 4$ colors (including the color when the light is off). We can reduce the number of colors to $O(1)$ at the cost of the running time, as follows. We encode the $L = \Delta + 4$ colors by a sequence of $\lceil \log L \rceil$ bits and transmit this sequence by emulating the Alternating Bit Protocol (ABP), also referred to as Stop-and-wait ARQ (see, e.g. [28]). This protocol uses a sequence number from $\{0, 1\}$ alternately to transmit the bits. The sender has four states corresponding to the transmitted bit $b \in \{0, 1\}$ and the sequence number. The receiver has two states that represent which sequence number is awaited. The data bits are accepted with alternating sequence numbers. This protocol ensures the correct transmission of the bit sequence without duplicates.

We emulate the ABP by using six different colors, one for each of the four states of the sender and one for each of the two states of the receiver. Seeing the current color of the sender, the receiver can decode the sequence number and the data bit. When a color corresponding to the correct sequence number is seen, the receiver sets its color, indicating that it waits for the next bit. When the sender sees the changed color of the receiver, it sets its color corresponding to the next data bit and the next sequence number. Therefore, encoding an original color in a sequence of $\lceil \log L \rceil = O(\log \Delta)$ bits and transmitting this sequence takes $O(\log \Delta)$ rounds. This leads to the following Theorem.

Theorem 3. *The modified Algorithm PACK fills an area represented by a connected graph in the ASYNC model by robots having a visibility range of* 1*,* $O(\log \Delta)$ *bits of persistent storage and* $O(1)$ *colors. The algorithm needs* $O(n^2 \log \Delta)$ *asynchronous rounds.*

3 BLOCK Algorithm

The PACK algorithm solves the Filling problem in arbitrary connected graphs by robots with a visibility range of 1 hop. An important property of the PACK algorithm is that the Leader can only move when the chain has reached the Packed state. Now we consider robots with a visibility range of 2 hops. Then the robots see each robot, that potentially could choose the same target vertex. The idea is that the Leader only chooses a vertex v as the target, if the 1 hop neighborhood of v does not contain any other robot with the light turned on, except when the light showing direction Δ (i.e. the robot will not move anymore, it wants to switch to Finished state, and waiting for the confirmation of the successor). A vertex neighboring to a robot with its light on (except the color Δ) is considered as *blocked* vertex for the Leader.

We introduce the following additional rules for the robots:

Leader: The Leader must not choose a blocked vertex as the target. As the visibility range of the robots is 2 hops, the Leader can identify the blocked neighbors. When only blocked or occupied vertices surround the Leader, it chooses to terminate its actions (sets the color Δ and after the confirmation of the successor it switches to Finished state), and the leadership will be taken by its successor.

Follower: Follower robots 'block' all their unoccupied neighboring vertices. As a result, all unoccupied vertices that are part of the chain are blocked: Before a Follower r would move from a vertex v, it sets the DIR color corresponding to the target and blocks all of its unoccupied neighboring vertices. In particular, it blocks the target vertex. Thus the Leader cannot choose the same target. Then r waits until the successor r' sets its CONF color and r moves from v. During the movement, the MOV color is set, which keeps the same unoccupied vertices blocked. When r leaves v, the vertex v is blocked by r'.

These rules ensure that each vertex on the chain is either occupied or blocked. Consequently, the Leader only moves to unvisited vertices. The pseudocode of the BLOCK algorithm is provided in the full version of the article [17].

3.1 Analysis

The proofs of Lemmata 5 - 8 are available in the full version of the article [17].

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

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

Lemma 7. *Robots cannot collide.*

Lemma 8. *BLOCK fills the area represented by a connected graph.*

Theorem 4. *Algorithm BLOCK fills the area represented by a connected graph in the ASYNC model by robots having a visibility range of* 2 *hops,* $O(\log \Delta)$ *bits of persistent storage, and using* $\Delta + 4$ *colors, including the color when the light is off.*

Proof. We can use the arguments of the proof of Theorem 1 as the area is filled (by Lemma 8), and collisions are not possible (by Lemma 7), the area will be filled without collisions. The robots store the same data in their persistent storage as in Theorem 1 and use the same set of colors.

Theorem 5. *In the ASYNC model, the BLOCK algorithm fills the area represented by a connected graph in* $O(n)$ *asynchronous rounds.*

Proof. Assume that the chain contains the robots r_1, r_2, ..., r_j, where r_1 is the current Leader and r_2, ..., r_j are on the path from the Leader to the Door, and assume that the Leader r_1 occupied its position and its successor r_2 has arrived at the previous position of r_1. When the first robot r_1 is placed at the Door (i.e. $j = 1$), it detects in the first asynchronous round, whether it is a Leader or a Follower. If the only neighbor is unoccupied, it becomes a Leader and moves in the first round. The first round ends. After r_1 left the Door, the next robot is placed there.

Assume now that $j \geq 2$. Let r_i, $i < j$ be a robot (r_i is either a Leader or a Follower) and assume that its successor r_{i+1} at its previous vertex. Let t be the current asynchronous round. If r_i is Leader, i.e. $i = 1$, we additionally assume that it has an unblocked and unoccupied neighboring vertex v. Otherwise, if r_i is not the Leader, assume that the target vertex of v of r_i is unoccupied, i.e. the predecessor r_{i-1} left v already. Then r_i sets the corresponding DIR color in round t. At the latest in round $t + 1$ the robot r_{i+1} sees the DIR color and sets its color to the CONF, allowing r_i to move. At the latest in round $t+2$ the robot r_i detects this and moves to its target v and at the end of that round r_i and r_{i-1} become neighbors again. Then, at the latest in round $t + 3$ the robot r_{i+1} detects that r_i left the neighboring vertex v'. If r_{i+1} is at the Door it moves at the latest in round $t + 3$. Otherwise, if r_{i+1} is not at the Door, and therefore, a successor robot r_{i+2} exists, r_{i+1} has to wait for the confirmation of r_{i+2} before the movement. We will show that r_{i+2} must be at the neighbor vertex behind r_{i+1} in round $t + 4$ at the latest. At the latest in round $t + 4$ the robot r_{i+2} sets its CONF color. Therefore, r_{i+1} can move to v' at the latest in round $t + 5$ and at the end of that round r_{i+1} and r_i become neighbors again. At the latest in round $t + 6$ we have the same situation regarding r_i and $r_i + 1$ as in round t, i.e. r_{i-1} shows its DIR color and r_i confirms it.

It remains to show that in round $t + 4$ at the latest r_{i+2} must be on the neighbor vertex of r_{i+1}. If r_{i+2} is at the Door, then it appeared there after r_{i+1} left the Door, i.e. before round t and r_{i+1} did not move; therefore they must be neighbors in round $t + 4$. Otherwise, let t' be the latest round before t, where r_{i+1} and r_{i+2} were neighbors and the robot r_{i+1} detects that its predecessor r_i moved from the neighboring vertex and r_{i+1} sets the DIR color. Then we can repeat the arguments with robots r_{i+1}, r_{i+2}, and round t' described above, and we obtain that (i) r_{i+1} moves in round $t'' \leq t' + 2$ and at the end of round $t' + 2$ the robots r_i and r_{i+1} become neighbors again, and (ii) r_{i+2} can move again at the latest in round $t'' + 3$ and in round $t'' + 4$ at the latest r_{i+2} and r_{i+1} must be neighbors. Since $t'' \leq t$, in round $t + 4$ at the latest r_{i+2} and r_{i+1} must be neighbors.

Summarizing the above description, the robot r_i moves at the latest in every 6^{th} round if r_i is a Follower or it is a Leader with an unblocked and unoccupied neighbor.

Assume now that r_i is Leader, its successor r_{i+1} is at its previous vertex, and all neighboring vertices of r_i are blocked or occupied in round t. Then r_i sets its Δ color to show the successor that it has to switch to Finished state. The successor r_{i+1} confirms it at the latest in round $t+1$. At the latest in round $t+2$ the robot r_i becomes Finished and turns the light off. At the latest in round $t + 3$ the robot r_{i+1} becomes the new Leader. Therefore, the leadership is taken within 4 rounds. At the latest in round $t + 3$ the new Leader r_{i+1} shows its new target if there is an unblocked and unoccupied neighboring vertex, or it sets the Δ to show the successor that it has to switch to Finished state.

When a Leader can move, it occupies an unvisited unblocked vertex in every 6^{th} round. Otherwise, its successor takes the leadership. Since the leadership is taken at most once by each robot during the whole algorithm, and there are n robots in the filled graph, at most $4n$ rounds used for all 'leadership taking'. Therefore, after $6n + 4n = 10n$ rounds, all vertices of the graph become filled.

4 Multiple Doors

We now consider the case in which there are $k \geq 2$ Doors. For the k-Door Filling, there is a situation that cannot be solved by the above methods: Let v be an unvisited vertex, which is neighboring to (at least) two Leaders r_1 and r_2. In order to fill the graph, exactly one of the Leaders, r_1 or r_2, has to move to vertex v. If one of the robots, say r_1, has been activated earlier, then r_1 sets the direction color corresponding to v, and it prevents r_2 to move to v (r_1 blocks v from r_2). However, if the activation times of r_1 and r_2 are exactly the same, then they would set the direction color at the same time, meaning they mutually block each other from moving to v. If r_1 or r_2 has no other unvisited vertex in their neighborhood, then none of them could move, and particularly, none of them would occupy v.

We use the concept from [5] and assume that robots entering from different doors have distinct colors. We propose a protocol, which uses a strict priority order between the Leaders originating from different Doors.

Priority protocol: The robots have k additional different colors corresponding to the Door they used for entering the area, where k is the number of Doors. We define a strict total order between these colors, called priority order. We call these k colors priority colors. After showing the direction to the successor and after the successor has confirmed it, the Leader sets its color to the corresponding priority color (instead of the MOV color) and starts its movement. It arrives to its target showing its priority color. We modify the blocking rule for the Leader in the following way: If there is a robot with a direction color (except the special color Δ), or confirmation color, or MOV color, or priority color with higher priority than r, then its neighbors are considered as blocked. Since there is a strict total order between the priority colors, in such a situation exactly one of them is allowed to move there.

We slightly change the rule *taking the leadership*: when the successor robot r notices that the Leader is switching to Finished state (by setting the direction color to Δ), r confirms it by setting its color to the priority color of the old Leader.

The proofs of Lemmata 9 and 10 are available in the full version of the article [17].

Lemma 9. *Priority protocol does not allow collisions.*

Lemma 10. *The BLOCK algorithm extended with the Priority protocol fills the connected graph.*

Theorem 6. *Algorithm BLOCK extended with the Priority protocol solves the k-Door Filling problem, $k \geq 2$, in the ASYNC model in finite time, with 2 hops of visibility, $O(\log \Delta)$ bits of memory and using $\Delta + k + 4$ colors including the color when the light is off.*

Proof. We can use the arguments of the proof of Theorem 1 as the area is filled (by Lemma 10), and collisions are not possible (by Lemma 9), the area will be filled without collisions. The robots store the same data in their persistent storage as in Theorem 1 and use $\Delta + k + 4$ colors.

5 Summary

In this work, we have presented solutions for the Filling problem by luminous robots in the ASYNC$^{O(1)}$ model. We have presented a method, called PACK, which solves the problem by robots with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 4$ colors for the single Door case, including the color when the light is off. We have shown that this algorithm needs $O(n^2)$ asynchronous rounds. Regarding asynchronous algorithms for the Filling problem, former works only guarantee termination within finite time. Our analysis provides the first asymptotic upper bound on the running time in terms of asynchronous rounds.

Then we have shown how the number of colors can be reduced to $O(1)$ at the cost of running time. The algorithm with 1 hop visibility range, $O(\log \Delta)$ bits of persistent memory, and $O(1)$ colors needs $O(n^2 \log \Delta)$ rounds.

After this, we have shown how the running time can be significantly improved by robots with a visibility range of 2 hops, with no communication, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 4$ colors, by presenting the algorithm called BLOCK. This algorithm needs $O(n)$ rounds.

Then we have extended the BLOCK algorithm for solving the k-Door Filling problem, $k \geq 2$, by using $O(\log \Delta)$ bits of memory, and $\Delta + k + 4$ colors, including the color when the light is off. The visibility range of 2 hops is optimal for the k-Door case (a counterexample when this problem cannot be solved in the ASYNC model with a visibility range of 1 hop was presented in [5], also holds for the ASYNC$^{O(1)}$ model).

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM J. Comput. **29**(4), 1164–1188 (2000)
2. Aljohani, A., Poudel, P., Sharma, G.: Complete visitability for autonomous robots on graphs. In: IPDPS 2018. pp. 733–742 (2018)
3. Amir, M., Bruckstein, A.M.: Minimizing travel in the uniform dispersal problem for robotic sensors. In: AAMAS 2019. pp. 113–121 (2019)
4. Augustine, J., Moses, Jr., W.K.: Dispersion of mobile robots: A study of memory-time trade-offs. In: ICDCN 2018. pp. 1:1–1:10 (2018)
5. Barrameda, E.M., Das, S., Santoro, N.: Deployment of asynchronous robotic sensors in unknown orthogonal environments. In: 4th ALGOSENSORS 2008. pp. 125–140. LNCS 5389 (2008)
6. Barrameda, E.M., Das, S., Santoro, N.: Uniform dispersal of asynchronous finite-state mobile robots in presence of holes. In: 9th ALGOSENSORS 2013. pp. 228–243. LNCS 8243 (2014)
7. Bhagat, S., Mukhopadhyaya, K.: Optimum algorithm for mutual visibility among asynchronous robots with lights. In: SSS 2017. pp. 341–355. LNCS 10616 (2017)
8. Bose, K., Kundu, M.K., Adhikary, R., Sau, B.: Arbitrary pattern formation by asynchronous opaque robots with lights. In: SIROCCO 2019. pp. 109–123. LNCS 11639 (2019)
9. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithm in asynchronous robot systems. SIAM J. Comput. **34**(6), 1516–1528 (2005)
10. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: Synchronizing asynchronous robots using visible bits. In: ICDCS 2012. pp. 506–515 (2012)
11. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theoretical Computer Science **609**, 171–184 (2016)
12. Daymude, J.J., Hinnenthal, K., Richa, A.W., Scheideler, C.: Computing by programmable particles. In: Distributed Computing by Mobile Entities, Current Research in Moving and Computing, pp. 615–681. LNCS 11340 (2019)
13. D'Emidio, M., Frigioni, D., Navarra, A.: Synchronous robots vs asynchronous lights-enhanced robots on graphs. Electronic Notes in Theoretical Computer Science **322**, 169–180 (2016)
14. Feletti, C., Mereghetti, C., Palano, B.: Uniform circle formation for swarms of opaque robots with lights. In: SSS 2018. pp. 317–332. LNCS 11201 (2018)
15. Flocchini, P., Santoro, N., Wada, K.: On memory, communication, and synchronous schedulers when moving and computing. In: OPODIS 2019. pp. 25:1–25:17 (2019)
16. Hideg, A., Lukovszki, T.: Uniform dispersal of robots with minimum visibility range. In: ALGOSENSORS 2017. pp. 155–167. LNCS 10718 (2017)
17. Hideg, A., Lukovszki, T.: Asynchronous filling by myopic luminous robots. CoRR **abs/1909.06895** (2019), http://arxiv.org/abs/1909.06895

18. Hideg, A., Lukovszki, T., Forstner, B.: Filling arbitrary connected areas by silent robots with minimum visibility range. In: ALGOSENSORS 2018. pp. 193–205. LNCS 11410 (2018)
19. 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)
20. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S., Wada, K.: Gathering on rings for myopic asynchronous robots with lights. In: OPODIS 2019. pp. 27:1–27:17 (2019)
21. Lukovszki, T., Meyer auf der Heide, F.: Fast collisionless pattern formation by anonymous, position-aware robots. In: OPODIS 2014. pp. 248–262. LNCS 8878 (2014)
22. Luna, G.D., Flocchini, P., Chaudhuri, S.G., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Information and Computation **254**(3), 392–418 (2017)
23. Ooshita, F., Tixeuil, S.: Ring exploration with myopic luminous robots. In: SSS 2018. pp. 301–316. LNCS 11201 (2018)
24. Peleg, D.: Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In: IWDC 2005. pp. 1–12. LNCS 3741 (2005)
25. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for asynchronous robots with lights. In: SSS 2017. pp. 265–281. LNCS 10616 (2017)
26. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Complete visibility for robots with lights in O(1) time. In: SSS 2016. pp. 327–345. LNCS 10083 (2016)
27. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: O(log N)-time complete visibility for asynchronous robots with lights. In: IPDPS 2017. pp. 513–522 (2017)
28. Tanenbaum, A.S., Wetherall, D.J.: Computer Networks. Prentice Hall Press, 5th edn. (2010)