Abstract. We present the problem of finding a maximal independent set (MIS) (named as MIS Filling problem) of an arbitrary connected graph with luminous myopic mobile robots. The robots enter the graph one after another from a particular vertex called the Door and move along the edges of the graph without collision to occupy vertices such that the set of occupied vertices form a maximal independent set.

In this paper, we explore two versions of the MIS filling problem. For the MIS Filling with Single Door case, our IND algorithm forms an MIS of size \( m \) in \( O(m^2) \) epochs under an asynchronous scheduler, where an epoch is the smallest time interval in which each participating robot gets activated and executes the algorithm at least once. The robots have three hops of visibility range, \( \Delta + 8 \) number of colors, and \( O(\log \Delta) \) bits of persistent storage, where \( \Delta \) is the maximum degree of the graph.

For the MIS Filling with Multiple Doors case, our MULTIND algorithm forms an MIS in \( O(m^2) \) epochs under a semi-synchronous scheduler using robots with five hops of visibility range, \( \Delta + k + 7 \) number of colors, and \( O(\log(\Delta + k)) \) bits of persistent storage, where \( k \) is the number of doors.

Keywords: Distributed algorithms, Multi-agent systems, Mobile robots, MIS, Filling Problem, Luminous Robots

1 Introduction

1.1 Motivation

The coordination among large number of autonomous mobile robots or agents has gained significant interest in recent years. Under the framework of “Look-Compute-Move” cycles, the robots can perform various tasks such as exploration [1], gathering [6,7,12,15], pattern formation [5,19], dispersion [2,14,18], scattering [10,16,17] and others. In this work, we consider the underlying environment as a graph, and the robots can stay at the nodes and move along the edges.

In general, maximal independent set (MIS) of a network graph play a significant role in decomposing the network into clusters of low diameter, which is
often very useful in designing and implementing distributed divide and conquer algorithms. MIS vertices can also be used as a network backbone for deploying communication infrastructure. For example, information dissemination in a low latency system where all robots form a network should be located at MIS vertices so that all other vertices are just one hop away.

The Filling problem, introduced by Hsiang et al. [11], considers the robots enter via particular vertices and fill an environment (graph) composed of pixels (vertices) and robots occupy every pixel (vertex). Later Hideg et al. [9] presented the Filling problem for an arbitrary connected graph. It is of interest to cover the entire graph but using a smaller number of robots. Thus forming an MIS by the robots that enter the graph becomes a natural extension. We call this problem the \textit{MIS Filling problem}.

In this paper, we consider \textit{luminous robots}, that are mobile robots possessing externally visible persistent memory (or \textit{lights}). Each vertex can contain at most one robot at a time. We say a collision happens when two or more robots move to the same vertex. Only one robot can travel along one edge at a time. In this problem, the robots enter the graph one by one through a specific vertex called the \textit{Door} and move in the graph along the edges from one vertex to another while avoiding a collision. The objective is only to occupy vertices that form an MIS. We solve two flavors of the problem: graphs with a single Door under an asynchronous (ASYNC) scheduler and graphs with multiple Doors under a semi-synchronous (SSYNC) scheduler. We use \textit{epochs} to denote the time complexity, where an epoch is the smallest amount of time required for all the participating robots to activate once. On each activation a robot executes a \textit{Look-Compute-Move (LCM)} cycle. In ASYNC, the cycles are independently executed within finite but unpredictable time. In SSYNC, time is discretely separated into rounds, and a subset of the robots are activated in each round and finish the execution of a cycle in the same round. Having multiple Doors instead of just one offers redundancy in situations where a Door can be blocked.

1.2 Related works

Kamei and Tixeuil [13] solve two variations of the maximum independent set (MAX,IS) placement problem for grid networks. The first one assumes knowledge of port-numbering for each node. It uses three colors of light and a visibility range of two. The other one removes the assumption of port-numbering and uses seven colors of light and a visibility range of three. Barrameda et al. [3] proposed algorithms for uniform dispersal or filling problem on any simply connected orthogonal space using identical asynchronous sensors. They present two algorithms: one for the single door, where sensors have one unit of visibility range and two-bit of persistent memory, and the other for multiple doors, where sensors have two units of visibility and a constant amount of persistent memory. They also prove that oblivious sensors cannot solve the problem deterministically even if they have unlimited visibility. For multiple doors, they show that the problem is unsolvable if the visibility range is less than two, even if sensors have unbounded memory. Further, even with unbounded visibility and memory,
they show that the problem is unsolvable if the sensors are identical. Barrameda et al. [4] extended the problem of uniform dispersal for orthogonal domains with holes. They solve the problem when robots have a visibility range of six without any direct communication among themselves. Later, they solve the problem using direct communication among robots to reduce the visibility radius without increasing the memory requirement. Hideg and Lukovszki [8] solve the filling problem in orthogonal regions, where the robots enter the region through entry points, called doors. They propose two algorithms with run-time $O(n)$, one for single door and the other for multiple door case. Later Hideg and Lukovszki [9] presented the filling problem for an arbitrary connected graphs in asynchronous setting where the goal is to fill the entire graph using myopic luminous robots.

The algorithm proposed by Hideg and Lukovszki [9] cannot be directly applied in MIS Filling problem, as the communication process in their PACK algorithm designed around one hop movement of the robots. For starters, one needs to maintain a two hop gap between the chain of robots, while simultaneously ensuring that the chain never crosses itself. The chain crossing problem does not arise if the chain is closely packed at one hop distance, and it becomes challenging in the presence of multiple chains. Also, at no point we can allow more than the size of MIS of robots to enter the graph, since that would render the problem unsolvable. We show that our algorithms handle these additional requirements and correctly form an MIS. The state-of-the-art results and ours’ are given in Table 1, where $m$ is the number of robots that form MIS.

| Algorithm | PACK [9] | BLOCK [9] | Algo 1 [13] | Algo 2 [13] | IND | MULTIND |
|-----------|---------|----------|------------|------------|-----|--------|
| Scheduler | ASYNC  | ASYNC   | ASYNC      | ASYNC      | ASYNC | SSYNC  |
| Problem   | Filling Problem | Filling Problem | MAX_IS Placement | MAX_IS Placement | MIS Filling | MIS Filling |
| Topology  | Connected graph | Connected graph | Grid network | Grid network | Connected graph | Connected graph |
| Number of Doors | Single | Multiple $(k)$ | Single | Single | Single | Multiple $(k)$ |
| Visibility Range | 1 hop | 2 hops | 2 hops | 3 hops | 3 hops | 5 hops |
| Memory (in bits) | $O(\log \Delta)$ | $O(\log \Delta)$ | $O(1)$ | $O(1)$ | $O(\log \Delta)$ | $O(\log(\Delta + k))$ |
| Number of Colors | $\Delta + 4$ | $\Delta + k + 4$ | 3 | 7 | $\Delta + 8$ | $\Delta + k + 4$ |
| Time Complexity | $O(n^2)$ epochs | $O(n)$ epochs | $O(n(L + l))$ steps $^3$ | $O(n(L + l))$ steps | $O(m^2)$ epochs | $O(m^2)$ epochs |

$^3$ The steps represent the total movement of robots throughout the execution of the algorithm. The number of nodes and grid dimensions are represented by $n$, $L$ and $l$, respectively.
1.3 Contributions

In this paper, we propose two algorithms IND and MULTIND corresponding to single and multiple doors.

- Algorithm IND solves the MIS Filling problem in graphs with a single Door under an ASYNC scheduler using robots with a visibility range of 3, $\Delta + 8$ number of colors, $O(\log \Delta)$ bits of persistent storage in $O(m^2)$ epochs, where $\Delta$ is the maximum degree of the graph, and $m$ is the number of robots that form an MIS.
- Algorithm MULTIND solves the MIS Filling problem in graphs with multiple Doors under an SSYNC scheduler using robots with visibility range of 5, $\Delta+k+7$ number of colors, $O(\log(\Delta+k))$ bits of persistent storage in $O(m^2)$ epochs, where $m$, $\Delta$, and $k$ are the number of robots, maximum degree of the graph and number of Doors, respectively.

1.4 Organization

Section 2 describes the model used in this paper. In Section 3 we consider the MIS filling problem with a single Door, i.e., $k = 1$ for robots under ASYNC scheduler and in Section 4 with multiple Doors, i.e., $k > 1$ for robots under SSYNC scheduler before concluding in Section 6.

2 Model

In this paper, we model the environment as a graph. We say that the graph contains a set of vertices that are connected to Doors from where robots can enter. The number of Doors in the graph is not known to the robots, but the robots are equipped with colors to distinguish themselves if they have entered the graph from different Doors. We assume that there are a maximum of $k$ Doors attached to the graph.

Graph: We consider an anonymous graph, i.e., the nodes of the graph are indistinguishable from each other. Each vertex $v$ of the graph contains port numbers corresponding to the incident edges from $[1, 2, \ldots, \delta(v)]$, where $\delta(v)$ is the degree of the vertex $v$. Given an anonymous connected port labeled graph $H = (V', E')$, we construct a graph $G = (V, E)$ with $k$ Doors, that adds two auxiliary vertices $\{d_i, d'_i\}$ corresponding to each Door that is connected to distinct vertices $\{v_1, v_2, \ldots, v_k\} \subset V'$. We have a path $d_i \rightarrow d'_i \rightarrow v_i$ corresponding to each Door $d_i$. The robots enter the graph through the Door, and a new robot appears at the Door immediately after it becomes empty. We say a vertex is free if none of the vertices adjacent to it are occupied by any robot. Since we add a buffer vertex corresponding to each Door, all the vertices in the original graph $H$ are free vertices in the beginning.
Robots: The focus is on completing the task using robots with minimal capabilities operating under certain adversarial conditions. The robots are autonomous (no central or external control), myopic (they have limited visibility range), anonymous (without distinguishable features or identification) and homogeneous (they all have the same capabilities/execute the same program). Additionally the robots are luminous, i.e., they have a light attached to them which can display various colors that represent the value of a state variable. This works as a mode of communication between the robots.

Time Cycle: Each robot operates in the Look-Compute-Move (LCM) model, in which the actions of the robots are divided into three phases.

- Look: The robot takes a snapshot of its surroundings, i.e., the vertices with in the visibility range and the colors of the robots occupying them.
- Compute: The robot runs the algorithm using the snapshot as the input and determines a target vertex or chooses to remain in place.
- Move: The robot moves to the target vertex if needed. A robot moves two hops in a single move phase.

Assumptions: We have the following assumptions regarding the knowledge of a robot and the properties of the underlying graph.

- The robots have no knowledge of the graph but an upper bound of $\Delta$, the maximum degree of the graph.
- For a robot placed at $v$ with a visibility range of $z$, the port numbers of all the vertices in its visibility range are visible.
- A robot knows all the colors, but they can only display one color corresponding to the Door via which it enters the graph. Note that, this color corresponding to the Door only used by one robot at a time, but all robots that come from the same Door can display it when the need arises.(We use unique colors for each Door to determine a hierarchy among them.)
- The movement of the robot is non-instantaneous.

Note that, we use directions and port-numbers interchangeably throughout the paper. Each port number corresponds to a DIR color, also the direction towards Successor or Predecessor.

Problem: We define the MIS filling problem formally as follows:

Problem 1. (MIS filling problem) Given an anonymous connected port labeled graph $G = (V, E)$ with $k$ Doors, the objective is to relocate robots that appear at Doors such that at termination, the robots occupy a set of vertices $V_1$ ($V_1 \subset V$) that forms a maximal independent set of $G$. 
3 Algorithm for MIS Filling with Single Door

We now describe the algorithm called IND, which is inspired by the PACK algorithm \cite{9} and uses the concept of Virtual Chain Method \cite{10}. The robots move throughout the graph is similar to the depth-first search (DFS). We assume that the robots operate under an asynchronous (ASYNC) scheduler. An epoch is the shortest time in which each robot not in the Finished state is activated at least once and performs an LCM cycle. Each robot requires a visibility range of 3 hops, $O(\log \Delta)$ bits of persistent memory, and $\Delta + 8$ colors.

3.1 Preliminaries

**Colors:** The colors used by the robots are described here.
- ON - Used initially when a robot arrives at the door.
- DIR - $\Delta$ colors corresponding to a port number in $[1, \Delta]$.
- CONF - Used to confirm that first DIR color has been seen and received.
- CONF2 - Used to confirm that second DIR color has been seen and received.
- CONF3 - Used to confirm that the Packed state is achieved.
- MOV - Used when a robot is in movement.
- OFF - Used by a robot in the Finished state.

Note that, a DIR color pointing towards a successor is used as a special color to indicate change of leadership.

**Definition 1.** ($k$ hops Neighborhood of a vertex $v$) For a vertex $v$, we define $k$-hops Neighborhood of $v$ to be the set of all vertices that are $k$ hops away from $v$ and denote it by $N_k^v$.

**Definition 2.** ($k$ hops Visibility Set of a robot $r$) For a robot $r$ placed at a vertex of the graph, we define the $k$ hops Visibility Set of $r$ to be the set of all vertices which are within $k$ hops from the current location of $r$. We denote it by $V_k^r$.

3.2 IND Algorithm

In this section, we present the rules for the robots that they follow to successfully form a chain with robots occupying alternative vertices. We say a vertex is free if none of its neighbors contain a robot. The first robot that enters the graph is called the Leader robot. We define Packed state as the state of a chain where all the alternative vertices are occupied by robots. We define it formally as follows:

**Definition 3.** (Packed state) Let $L$ be a positive odd integer and $P = \{v_1, v_2, \ldots, v_L\}$ be a path starting from the Door at $v_1$ and the leader at $v_L$ such that every second vertex of $P$, i.e., $v_1, v_3, \ldots, v_{L-2}$ was visited by the Leader. A chain of robots is in a Packed state if the vertices $v_1, v_3, \ldots, v_{L-2}$ are occupied by follower robots.
Table 2: List of Variables used in algorithms IND and MULTIND.

| Variable | Description |
|----------|-------------|
| State    | State of the robot - None/Follower/Leader/Finished |
| Target   | Directions to the target vertex |
| NextTarget | Directions to the vertex to which the robot has to move after reaching the Target vertex (Used by Followers) |
| Successor | Successor robot |
| Predecessor | Predecessor robot |
| Color   | Color displayed by robot’s light |
| Entry   | Previous location of the robot/location of the follower stored in terms of directions |

Only the Leader is allowed to move in the Packed state. WAIT color is used by the Leader as soon as it reaches its target vertex to indicate that it is waiting for the chain to reach the Packed state. This is so that the Leader can choose a target such that neither the target nor any of the vertices adjacent to it are going to be occupied by any other robot. Secondly, the Leader (Predecessor) $r_1$ needs to communicate the directions it will take to its follower (Successor) $r_2$ so that $r_2$ can know in which direction $r_1$ has moved. This applies to all predecessors and successors. The Color variable represents the color displayed by the robot’s light, and the Target variable represents directions to the target vertex. The NextTarget variable represents directions to the next target after reaching the target vertex. Entry variable represents the direction of the two hops a robot moved, so that the robot knows the location of its follower.

We explain communication between Predecessor and Successor and the restoration of Packed state after the movement with an example as shown in Fig. 1.

**Communicating the movement directions:** The robots establish the Predecessor and Successor relationship among them by their order of appearance from the Door. A Predecessor $r$ communicates its destination $r$.Target to a Successor by showing the colors corresponding to the port numbers at the vertex. Suppose $r_1$, $r_2$ and $r_3$ are located at $e$, $c$ and $a$, respectively as shown in Fig. 1.

![Fig. 1: Communication of color from $r_1$ to $r_2$](image)

$r_1$ determines that it will move to vertex $g$. Now, it has to communicate that to $r_2$, so that $r_2$ can follow $r_1$. First $r_1$ shows the DIR color of port 1 corresponding to the edge $ef$ as it wants to take the path $ef - fg$, and $r_2$ responds by showing CONF color. Then $r_1$ sets its color to CONFC to confirm that it has seen the CONF color. Now, $r_2$ also sets its color to CONFC to
inform \( r_1 \) that it is ready to receive the second color. (CONFC color is used to distinguish between two DIR colors that can be the same.) Next, \( r_1 \) shows the DIR color of port 1 corresponding to edge \( fg \). \( r_2 \) confirms that it has seen the color by setting its color to CONF2. Once \( r_1 \) sees CONF2, it can move to \( g \) after setting its own color to MOV. Now \( r_2 \) sets \( e \) as its target (which was the old position of \( r_1 \)). Then, \( r_2 \) does the same process to communicate DIR colors to \( r_3 \).

**Restoration of Packed state after movement:** We describe this module with the Fig. 1. After the leader \( r_1 \) moves to \( G \) with color MOV, \( E \) becomes empty, so the Packed state is now distorted. So \( r_1 \) changes its color to WAIT. Now, \( r_2 \) moves to \( E \) with color MOV after communicating DIR colors to \( r_3 \). So, \( C \) becomes empty. After that, \( r_3 \) moves to \( C \) with color MOV without communicating DIR colors as it does not have any successor at this movement. As soon as \( r_3 \) leaves the Door, a robot \( r_4 \) is placed at the Door with color ON. After \( r_3 \) reaches \( C \), it sees \( r_4 \) with color ON and \( r_3.\text{Color} = \text{MOV} \). So, it changes its color to CONF3. Now, \( r_2 \) also changes its color to CONF3 after seeing \( r_3.\text{Color} = \text{CONF3} \) and \( r_2.\text{Color} = \text{MOV} \). The leader \( r_1 \) sees \( r_2 \) with color CONF3 and \( r_1.\text{Color} = \text{WAIT} \), so it understands that the Packed state is achieved. Now, \( r_1 \) looks for a new target to move to.

**Transferring the leadership:** The current leader \( r_1 \) transfers its leadership if either of the two scenarios occurs. First, \( r_1 \) gets stuck i.e., there are no other free vertices left to move to from the current vertex of the leader. Secondly, for each free vertex \( v \in V^2_{r_1} \), \( r_1 \) finds at least one vertex \( v' \in N^2_v \cap V^3_{r_1} \) with a robot not in Finished state. In both of the above cases, \( r_1 \) transfers the leadership to its successor \( r_2 \) by communicating the direction that points towards \( r_2 \).

**Detailed description:** When a robot first appears at the Door, it initializes to the None state and sets color ON. Let \( r_1 \) be the first robot that appears at the Door. \( r_1 \) does not find a robot on any adjacent vertices, so it changes its state to Leader. Now \( r_1 \) is the Leader and chooses a target vertex two hops away and moves to it with color MOV. As soon as the Leader \( r_1 \) leaves the Door, the next robot \( r_2 \) appears at the Door. At this time, \( r_1 \) is still in motion and is nearest to \( r_2 \). \( r_2 \) sees this and becomes the follower of \( r_1 \) and sets \( r_2.\text{Color} = \text{ON} \), where \( r_1.\text{Successor} = r_2 \) and \( r_2.\text{Predecessor} = r_1 \). After \( r_1 \) reaches its target, it sets \( r_1.\text{Color} = \text{WAIT} \) to indicate that it is waiting for Packed state. When \( r_2 \) sees \( r_1.\text{Color} = \text{WAIT} \) and \( r_2.\text{Color} = \text{ON} \), \( r_2 \) changes its color to CONF3. \( r_1 \) now chooses a new free vertex (if any) as \( r_1.\text{Target} \) and communicate its directions to \( r_2 \) as described above. When \( r_1 \) gets confirmation from \( r_2 \) (\( r_2 \) sets its color to CONF2), \( r_1 \) moves to \( r_1.\text{Target} \) with color MOV and again changes its color to WAIT to indicate that it is waiting for the chain to be in Packed state.

When the chain is in Packed state, the leader \( r_1 \) chooses a free vertex \( v \) from \( V^2_{r_1} \) as target, if all \( v' \in N^2_v \cap V^3_{r_1} \) are either unoccupied or having robots at Finished state. Then, \( r_1 \) communicates the directions of \( r_1.\text{Target} \) to \( r_2 \) as follows: \( r_1 \)
displays DIR color corresponding to \( r_1.\text{Target}.\text{One} \) by setting \( r_1.\text{Color} \). \( r_2 \) sees this and stores the direction in \( r_2.\text{NextTarget}.\text{One} \). \( r_2 \) confirms that it has seen the first DIR color by displaying CONF color. \( r_1 \) confirms that it has seen CONF on \( r_2 \) by setting \( r_1.\text{Color} \) to CONF (confirmation of confirmation). \( r_2 \) sees CONFC on \( r_1 \) and in turn, sets \( r_2.\text{Color} \) to CONF to show that it is ready to receive the second DIR color. The second DIR color \( r_1.\text{Target}.\text{Two} \) is displayed by \( r_1 \) and is seen by \( r_2 \). \( r_2 \) stores this second direction in \( r_2.\text{NextTarget}.\text{Two} \) and sets \( r_2.\text{Color} \) to CONF2 to send confirmation that the second direction of \( r_1 \) is received. Once \( r_1 \) has seen CONF2 in \( r_2 \), it moves two hops to \( r_1.\text{Target} \) in Move phase by setting \( r_1.\text{Color} = \text{MOV} \). After reaching to \( r_1.\text{Target} \), \( r_1 \) sets \( r_1.\text{Color} = \text{WAIT} \). Now \( r_2 \) sets \( r_2.\text{Target} \) based on the information stored in \( r_2.\text{NextTarget} \). \( r_2 \) needs to reach the old position of \( r_1 \).

Finally, when the Leader \( r_1 \) can no longer find free vertices to move to, it communicates this information to its follower \( r_2 \) using the DIR color that points towards \( r_2 \). Then, \( r_1 \) changes its color to OFF and goes into the Finished state, and \( r_2 \) becomes the Leader and continues exploring the graph.

When a robot moves to its target vertex, it sets \( r.\text{Entry} \) to the directions of the two hops it moved under \( \text{Entry} \) variable (so that the robot knows the location of its follower).

**Pseudocode of IND Algorithm:** The subroutine \textbf{Communication}(\( r.\text{Target} \)) communicates the target set by a robot \( r \) to its successor. The subroutine \textbf{Receive}(\( r.\text{Predecessor}.\text{Target} \)) works when \( r \) receives directions from its predecessor. \textbf{Packed\_State}(\( r,r.\text{Successor} \)) works for achieving packed state after the movement of the robots. \textbf{Leadership\_Transfer}(\( r \)) works when the current leader \( r \) finds no vertex to move to.

```
Algorithm 1: Communication(r.Target)
1 if r.Target is set & r.Successor.Color = CONF3 then
2   Set r.Color to match r.Target.One \( \triangleright \) (Showing the first DIR color)
3 else if r.Target is set & r.Successor.Color = CONF then
4   Set r.Color = CONFC \( \triangleright \) (Confirming the confirmation sent by successor)
5 else if r.Target is set & r.Successor.Color = CONFC then
6   Set r.Color to match r.Target.Two \( \triangleright \) (Showing the second DIR color)
7 else if r.Target is set & r.Successor.Color = CONF2 then
8   Set r.Color = MOV \( \triangleright \) (Moving towards the target)
9   Move to r.Target
```

**Example** Now, we illustrate the execution procedure of Algorithm (IND) with the following example. The location of the Door and the initial setting is shown
**Algorithm 2: Receive(r, Predecessor, Target)**

1. if \( r.\text{NextTarget} \) is not set then
2.  
3.  
4.  
5.  
6.  
7.  
8.  
9.  

**Algorithm 3: Packed State**

1. if \( r.\text{Color} = \text{MOV} \) \& \( r.\text{Successor}.\text{Color} = \text{ON} \) then
2.  
3.  
4.  
5.  
6.  

**Algorithm 4: Leadership Transfer**

1. if \( r.\text{Successor} \) is set then
2.  
3.  
4.  
5.  
6.  
7.  
Algorithm 5: IND

1. if r.State is Leader then
2.   if r.Target is not set then
3.     if r.Entry is set and r.Successor.Color = CONF3 then
4.       if ∃ a vertex v ∈ V_r^2 such that all v′ ∈ V_r^2 ∩ N_v^2 is either unoccupied
5.         or occupied with robots in Finished state then
6.           r sets v as r.Target by setting r.Target.One and r.Target.Two
7.           COMMUNICATE(r.Target)
8.           Set r.Color = WAIT
9.           r clears r.Target by clearing r.Target.One and r.Target.Two
10.          PACKED_STATE(r, r.Successor)
11.         else
12.            LEADERSHIP_TRANSFER(r)
13.       end if
14.     end if
15.   end if
16. end if
17. if r.State is Follower then
18.   if r.NextTarget.One is not set then
19.     Receive(r.Predecessor.Target)
20.   else
21.     if r.Target is set then
22.       COMMUNICATE(r.Target)
23.       Clear r.NextTarget.One and r.NextTarget.Two
24.       PACKED_STATE(r, r.Successor)
25.     end if
26.   end if
27. end if
28. if r.State is None & r.Color = ON then
29.   if r does not find any robot within a distance of 2 hops then
30.     Find a vertex v ∈ V_r^2
31.     Set v as r.Target by setting r.Target.One and r.Target.Two
32.     Change r.State to Leader
33.     Set r.Color = MOV
34.     r moves to r.Target ⊿ (r moves to r.Target.Two)
35.     Set the direction of the 2 hops r moves as r.Entry
36.     r sets r.Color to WAIT
37.   else
38.     r chooses the nearest robot as r.Predecessor
39.     Change r.State to Follower
40.     if r.Predecessor.Color = WAIT & r.Color = ON then
41.       Set r.Color = CONF3
in Fig. 2(a). The robot $r_1$ first appears at the Door in Fig. 2(b). As soon as $r_1$ moves away from the Door, robot $r_2$ appears there and sets $r_1$ as its predecessor. This situation is shown in Fig. 2(c). In Fig. 2(d) $r_1$ reaches its target vertex but $r_2$ does not move until $r_1$ has moved away from its current position. $r_1$ reaches a new vertex and $r_2$ follows it, $r_3$ appears at the Door and follows $r_2$ which is shown in Fig. 2(e). Observe that $r_1$ is stuck so it transfers its leadership to its follower, i.e., $r_2$. In Fig. 2(f) $r_2$ reaches a new target vertex and its followers follow.

3.3 Analysis of IND Algorithm

First, we present few lemmas that establish the behavior of the robots. The proofs of the following lemmas and theorem can be found in Appendix.

Lemma 1. There can be at most one Leader robot, and the Leader robot $r_1$ moves to a free vertex $v \in V^2_{r_1}$ such that every $v' \in N^2_{r_1} \cap V^3_{r_1}$ is either unoccupied or occupied by robots in Finished state.

Proof. In the rules for Transferring the Leadership, when a Leader $r_1$ signals to its successor $r_2$ that it is stuck, $r_2$ can become the Leader only after the previous Leader $r_1$ has switched to Finished state (recognized by OFF color on $r_1$). The first robot placed becomes the Leader, and the robots appearing next can become a Leader only after the previous one reaches the Finished state. Therefore, there can be at most one Leader at any time during the dispersion.

A free vertex has none of its adjacent vertices occupied by a robot. As the current Leader $r_1$ can only move when the chain is in a Packed state, it chooses a vertex in $V^2_{r_1}$ that is free by checking all the vertices adjacent to potential target vertex are not occupied by any robot. This can be done since the robots have a visibility range of 3. Further, there is no possibility of this target vertex or any vertex adjacent to it being occupied by any other robot as the Leader is allowed to move only when the chain is in the Packed state. If there is a free vertex $v \in V^2_{r_1}$ as a potential target, then every 2 hops neighbor $v'$ of $v$ ($v' \in N^2_{r_1} \cap V^3_{r_1}$), that is visible to $r_1$ needs to be either free or occupied by robots in Finished state. Due to this condition, a chain does not cross itself as shown in Fig. 3. If no such free vertex $v$ is found, then $r_1$ transfers the leadership to $r_2$ and switches to Finished state. \hfill \Box

Corollary 1. The leader $r$ moves to a free vertex $v$ such that every vertex $v' \in N^2_v \cap V^3_v$ is either unoccupied or occupied by a robot in Finished state. Consequently, the chain never crosses itself.

Lemma 2. The Robots do not collide.

Proof. When a robot appears at the Door, it has only one neighbor robot except for the first robot. That neighbor robot will become the Predecessor, which the robot will follow during the movement. The predecessor shows the directions to the Successor in which it will move, before moving so that the Successor can
(a) Location of the Door and initial setting of the graph.

(b) The robot $r_1$ appears at the Door and becomes the Leader.

(c) As soon as $r_1$ moves away from the Door and $r_2$ becomes its follower.

(d) $r_1$ reaches its target vertex.

(e) $r_1$ transfers its leadership to $r_2$ as there are no free vertex for it.

(f) $r_2$ is now the Leader and moves to new free vertices.

(g) $r_2$ keeps exploring the graph.

Fig. 2: An example execution of the IND algorithm
always follow it. After the target is set, the Predecessor moves first and only then the follower moves in the direction of its Predecessor. Each Successor robot has one Predecessor, which means they cannot collide with each other. Also, from Corollary 1, a chain does not cross itself and hence the collision never happens.

\[ \square \]

**Lemma 3.** No two robots in the Finished state occupy adjacent vertices.

**Proof.** A robot can go to the Finished state only from the Leader state. From Lemma 1 we know that a Leader robot moves only to free vertices. As only free vertices are occupied, the robots are not present in adjacent nodes. \[ \square \]

From Lemma 3, we can say that when a robot enters the Finished state, none of the vertices adjacent to it are occupied by a robot which means eventually, all the vertices occupied by robots form an independent set.

**Remark 1.** If a vertex is occupied by a robot in the packed state, it remains occupied therefrom.

**Theorem 1.** Algorithm IND fills a maximal independent set.

**Proof.** From Lemma 3, the filled vertices form an independent set. Suppose the independent set is not maximal. As the graph is connected, there exists a vertex \( v \) which is free and has a robot \( r_1 \) in the Finished state two hops away since the execution of the algorithms is done. Before the robot \( r_1 \) entered the Finished state, it was in the Leader state. The Leader \( r_1 \) switches into the Finished state in two cases. First, it cannot find any free vertex two hops away. Secondly, it finds a free vertex \( v \) such that there exists a neighbor \( v' \) of \( v \) in \( N_v \cap V_{r_1}^3 \) occupied by a robot not in Finished state. If all the neighbors of \( v \) in \( N_v \cap V_{r_1}^3 \) are unoccupied or have robots in Finished state, \( r_1 \) has a clear path to \( v \). Hence, it cannot switch to Finished state, which leads to a contradiction. If the 2 hops neighbor \( v' \) of \( v \) has a robot not in Finished state, \( r_1 \) does not have a clear path to \( v \). In that case, \( r_1 \) transfers the leadership to its successor by communicating the direction pointing towards its successor and eventually the robot on \( v' \) will becomes the leader. Since \( v' \) is 2 hops away from \( v \), the robot on \( v' \) cannot switch to Finished state as it gets a clear path to the free vertex \( v \), which again leads to contradiction.

\[ \square \]

**Lemma 4.** Algorithm IND fills an MIS of \( G \) with luminous robots having visibility range of 3 hops, \( O(\log \Delta) \) bits of memory, and \( \Delta + 8 \) colors.

**Proof.** The visibility range of 3 hops for robots executing IND algorithms is necessary. Otherwise, the robots cannot check if the target vertex is free or not. The robots require \( O(\log \Delta) \) bits of memory to store the following: State (4 states; 2 bits), Target (directions to the target vertex: \( 2[\log \Delta] \) bits), Next-Target (directions to the vertex to which the robot has to move after the Target vertex is reached: \( 2[\log \Delta] \) bits).

\[ ^4 \text{Two port numbers are needed to be stored corresponding to the movement.} \]
The colors used by the robots are $\Delta$ colors to show the directions to the target of the robot, including the special color that points in the direction of Follower to switch to the Finished state. Initially when the robot is placed at the Door, the robot sets its color to ON. There are four additional colors (CONF, CONFC, CONF2 and CONF3) for confirming that the robot saw the signaled direction and confirmations of the Predecessor or the Successor and the Packed state is achieved, one color WAIT used when the Leader is waiting for the Packed State, one color MOV used during the movement and the OFF color.

Now we analyze the time complexity of the algorithm in terms of epochs. To find the total time required by the algorithm, we first individually establish the time-bound on the movement of robots. Consider a chain containing $i$ robots $\{r_1, r_2, \ldots, r_i\}$, where $r_1$ is the Leader and the foremost robot in the chain. The robots $r_1, r_2, \ldots, r_i$ are on alternative vertices on the path from the vertex occupied by the Leader to the Door vertex. Suppose the chain is in the Packed state. We determine the time required for two consecutive movements of the Leader. We first determine the time required for all the robots in the chain to occupy the position of their Predecessor. As a result, a new robot appears at the Door, increasing the size of the chain. Next, we find the time required for the robots in the chain to set their colors to CONF2, indicating that they have received the movement direction of their Predecessor. We also find the time required for the chain to reach in the Packed state again after the leader of the chain moves.

**Lemma 5.** Algorithm IND takes at most $i$ epochs for all the robots in a chain of length $i$ to perform one MOVE operation.

*Proof.* Once the chain is in the Packed state, the leader robot $r_1$ moves to its target vertex in one epoch and $r_1$ sets its color to WAIT. By the next epoch, $r_2$ observes that $r_1$ has left its previous vertex, so it moves to $v$. In a cascading manner all Successors move to their Predecessors location. Thus in the worst case, it takes $i$ epochs for all the robots in the chain to move. \hfill $\Box$

**Lemma 6.** Algorithm IND takes at most $7i$ epochs for the robots in a chain of length $i$ to reach the Packed state after movement.

*Proof.* As per the algorithm description, the communication between a Predecessor $r_p$ and its Successor $r_q$ by showing a series of seven colors: first DIR color at $r_p$, CONF at $r_q$, CONFC at $r_p$, CONFC at $r_q$, second DIR color at $r_p$, CONF2 at $r_q$ and CONF3 at $r_q$. A leader can only move when the chain is in the Packed state. For a chain to be in the Packed state, the Successor of the leader robot has to show CONF3 color. The process of communication of colors starts from the new robot $r_{i+1}$ that appears at the Door. It takes at most seven epochs for the communication between $r_i$ and $r_{i+1}$. Similar communication happens between all $i$ pairs of consecutive robots on the chain. So it takes at most $7i$ epochs for the chain to reach Packed state. \hfill $\Box$

**Lemma 7.** Algorithm IND takes at most 4 epochs to have a transfer of leadership from a leader to its Successor.
Proof. Now consider the situation where the Leader $r_1$ cannot find any free vertices two hops away. If $r_1$ is at the Door, it sets its color to OFF and switches to Finished state; the maximal independent set is filled. Otherwise, $r_1$ switches to the Finished state by setting the DIR towards its follower. $r_2$ sees the DIR color and sets $r_2\.Color$ to CONF. $r_1$ sees the CONF color, it switches to color OFF. $r_2$ becomes the new Leader once it sees the color OFF at $r_1$. In total, this needs at most 4 epochs because of the sequence of colors (DIR, CONF and OFF) and final state change at $r_2$. \hfill $\Box$

Theorem 2. The algorithm IND runs in $O(m^2)$ epochs.

Proof. For an MIS of size $n$, we can have a chain of length at most $n$. For each increase in a chain of length $i$, it takes at most $7i$ epochs from Lemma 5 and 6. Also, we can have at most $m$ leadership transfers. From Lemma 7, each transfer of leadership takes at most 4 epochs. So in total we need, $\sum_{i=1}^{m} 7i + 4m = O(m^2)$ epochs. Therefore, after at most $O(m^2)$ epochs, an MIS of vertices of the graph becomes filled. \hfill $\Box$

The corollary below follows from Corollary 1 in [9].

Corollary 2. (i) Assume that there are no inactive intervals between the LCM cycles and that every LCM cycle of every robot takes at most $t_{max}$ time. Then the running time of the IND algorithm is $O(m^2t_{max})$. (ii) The IND algorithm needs $O(m^2)$ LCM cycles under an FSYNC scheduler.

4 Algorithm for MIS Filling with Multiple Doors

The MULTIND is largely similar to algorithm IND with a few modifications. It works under a Semi Synchronous (SSYNC) scheduler, where a subset of robots is activated in each round, where each activated robot finishes its LCM cycle in the same round. We define an epoch similarly as the minimum number of rounds where all robots are activated at least once. Note that an epoch can have a variable number of rounds. The graph has a maximum of $k$ number of Doors. The robots do not have any knowledge about the number of Doors. The visibility range of the robots is five hops. Each robot has $O(\log(\Delta + k))$ bits of persistent memory, and $\Delta + k + 7$ colors where $\Delta$ number of DIR colors, CONF, CONF2, CONF3, MOV, ON, OFF and $k$ number of WAIT colors denoted by WAIT-1, WAIT-2, ..., WAIT-$k$ representing that a robot is waiting as well as their rank. All the robots entering from a particular Door can only display WAIT color corresponding to that Door. The WAIT colors can be compared against each other to establish dominance between the robots. Initially a robot has color ON when it is placed at the Door. The proposed MULTIND runs in $O(m^2)$ epochs.

4.1 The MULTIND Algorithm

The Leader robots need to avoid collision with other Leader robots and the follower robots in chains. The robots make use of the hierarchy among the $k$
WAIT colors to avoid collision with another Leader robot. The Leader robots also avoid cutting through a chain to avoid collision with follower robots in another chain. In the multiple Door situation, the Leaders display the WAIT-i color instead of the WAIT color used in a single Door case. In the Look phase, if a Leader robot $r_i$ with color WAIT-i sees any other Leader $r_j$ with color WAIT-j such that $j < i$, then we say that the Leader robot $r_j$ dominates Leader robot $r_i$. The Leader robot $r_j$ is said to be the dominating and the Leader robot $r_i$ is said to be dominated. Note that a dominating Leader robot can be dominated by another Leader robot at the same time. If a Leader robot is not dominated by any other robot, it can choose a target.

The rest of the model is the same as in the Single Door case. The robots can be in any one these states during execution: None, Leader, Follower, Finished. When the robots appear at the Door, they are initialized with None state and with color ON. We define $\mathcal{P}_k(v_i, v_j)$ as the set of vertices that are part of all the paths from $v_i$ to $v_j$ of length $k$. Note that there can be multiple paths of length $k$, and we include vertices of all those paths. The additional rules are applicable to the Leader as follows.

**Movement of a Leader robot:** If a chain is in Packed state, the corresponding leader $r_L$ chooses a free vertex in $V^2_{r_L}$ as target in one of the following way.

- If $V^5_{r_L}$ has a dominating leader $r_{L}'$ of some other chain, then a free vertex $v \in V^2_{r_L}$ is chosen such that $v \notin \mathcal{P}_5(r_L, r_{L}')$. If no such vertex $v$ is found, then $r_L$ transfers the leader to its successor by pointing the direction towards the successor and goes into Finished state by changing its color to OFF.
- If $r_L$ is the leader dominating some other leader $r_{L}'$ in $V^5_{r_L}$, then a free vertex $v \in V^2_{r_L}$ is chosen such that every vertex $v' \in N^2_v \cap V^2_{r_L}$ is either unoccupied or occupied with robot in Finished state. If no such vertex $v$ is present, $r_L$ transfer its leadership to its successor.
- If $V^5_{r_L}$ does not have any other leader robot, then a free vertex is chosen as target such that every vertex $v' \in N^2_v \cap V^2_{r_L}$ is either unoccupied or occupied with robot in Finished state. If no such free vertex is there, $r_L$ transfers the leadership to its successor.

![Fig. 3: A chain $r_L - r_2 - r_3 - r_4 \cdots$ does not cross itself](image1)

![Fig. 4: Two chain do not cross each other](image2)
After the target is fixed for a leader $r_L$, it communicates the target to its successor using DIR colors. After getting confirmation from its successor, it moves to the target with color MOV and waits for the chain to be in Packed state with respective WAIT color. Additionally, irrespective of whether a Leader robot is dominated or not, the target chosen is such that the Leader robot does not cut through a chain while moving to that target vertex, i.e., the vertex between the target vertex and the current vertex does not have more than one of its neighboring vertices occupied at least one of which is an active robot (a robot not displaying OFF color) other than the current Leader robot. As shown in Fig. 4, $r_L$ and $r'_L$ are the dominating and dominated Leader at an instance and $v$ is a free vertex which is 2 hops away from $r_L$. $r_f_1$ and $r_f_2$ are some followers of some other chain. $r_L$ does not reach to $v$ even if $r_L$ is the dominating Leader, as $r_f_1$ and $r_f_2$ is not in Finished state. As shown in Fig. 3, a Leader $r_L$ sees a free vertex $v$ two hops away. But $r_L$ cannot reach to $v$ (self cross) as the two hops neighboring vertices are occupied by $r_3$ and $r_4$.

**Pseudocode of MULTIND:** The pseudocode of the MULTIND (Algorithm 6) is given below. The subroutine MULTIND_FindTarget($r$) is used for finding $r$.Target. All the other subroutines such as COMMUNICATION($r$.Target), RECEIVE($r$.Predecessor.Target), PACKED_STATE($r$, $r$.Successor) and LEADERSHIP_TRANSFER($r$) remain same as IND algorithm.

```
Algorithm 6: MULTIND_FindTarget($r$)

1. if $\exists$ a dominating leader $r' \in V^5_r$ then
2.   if $\exists$ a free vertex $v \in V^2_r$ such that $v \notin P(r, r')$ then
3.     Set $v$ as $r$.Target
4.   else
5.     LEADERSHIP_TRANSFER($r$) \triangleright (No such free vertex to move to)
6. else
7.   if $\exists$ a free vertex $v \in V^2_r$ such that all $v' \in N^2_r \cap V^2_r$ is either unoccupied or occupied with robots in Finished state then
8.     Set $v$ as $r$.Target
9.   else
10.    LEADERSHIP_TRANSFER($r$)
```

4.2 Example of MULTIND

We use an example to illustrate the execution of the MULTIND algorithm. Consider the graph with respective Door positions in Fig. 5(a). Door 1 ranks higher in the hierarchy than Door 2, which ranks higher than Door 3. Assume
Algorithm 7: MULTIND

1. if r.State is Leader then
2.   if r.Target is not set then
3.     if r.Entry is set and r.Successor.Color = CONF3 then
4.       MULTIND_FindTarget(r) & Communicate(r.Target)
5.       r clears r.Target by clearing r.Target.One and r.Target.Two
6.       Set r.Color = WAIT-i \(\triangleright\) (Wait color corresponding to Door-i)
7.       PACKED_STATE(r, r.Successor)
8.   end if
9. end if
10. end if
11. end if
12. if r.State is Follower then
13.   if r.NextTarget.One is not set then
14.     RECEIVE(r.Predecessor.Target)
15.   else
16.     if r.Target is set then
17.       Communicate(r.Target)
18.       r clears r.Target by clearing r.Target.One and r.Target.Two
19.       PACKED_STATE(r, r.Successor)
20.     end if
21.   end if
22. end if
23. if r.State is None & r.Color = ON then
24.   if r does not find any robot less than a distance of 2 hops then
25.     Change r.State to Leader
26.     Set r.Color = WAIT-i \(\triangleright\) (Wait color corresponding to the Door-i)
27.     MULTIND_FindTarget(r) & Set direction of movement as r.Entry
28.     Set r.Color = MOV & Move to r.Target
29.     Set r.Color = WAIT-i
30. else
31.   r chooses the nearest robot as r.Predecessor
32.   Change r.State to Follower
33.   if r.Predecessor.Color = WAIT-i & r.Color = ON then
34.     Set r.Color = CONF3
35. end if
36. end if
that all the robots are activated in each epoch. The positions of the robots in the next epoch are shown in Fig. 5(b). Each Leader robot makes a move. Observe that Leader-1 and Leader-2 are within the visibility range of each other. Similarly, Leader-2 and Leader-3 are within the visibility range. Leader-2 is dominated by Leader-1 and hence, its target vertex should not lie on the path to Leader-1. A similar argument applies for Leader-3 which is dominated by Leader-2. Fig. 5(c) shows the positions of the robots after the next epoch. All the Leaders are now stuck and have to transfer their Leadership. The result of that is shown in Fig. 5(d). Leader-2 has no possible target vertices and has to transfer its leadership again. Leader-3 is dominated by Leader Leader-2. In the next epoch shown in Fig. 5(e), Leader-1 and Leader-3 make a move. Leader-2 is at its Door. In the next epoch, Leader-2 goes into Finished state. Similarly, in the next epoch, Leader-1, and Leader-3 transfer their leaderships which is shown in Fig. 5(f). Leader-3 makes one final move in Fig. 5(g).

4.3 Analysis of MULTIND Algorithm

Lemma 8. A Leader robot always moves to and occupies free vertices.

Proof. Since the robots have a visibility range of 5 hops, a Leader robot can choose a target vertex such that none of its neighboring vertices are occupied by a robot that is not a Leader. Also two leaders do not occupy adjacent vertices. For a leader \( r_L \), if there is another Leader \( r'_L \) within 5 hops visibility of \( r_L \), one of them dominates the other. Let \( r_L \) be the dominating robot. MULTIND algorithm ensures that \( r'_L \) does not choose a vertex as its target in the path where it is being dominated. So, either \( r'_L \) finds a target on some other path where is not getting dominated or transfers the leadership (in case of no such target is there). So, the dominating Leader \( r_L \) can choose a free vertex as its target from the path \( P_5(r_L, r'_L) \). Hence, the robots cannot choose target vertices such that they are adjacent. \( \square \)

Lemma 9. The Robots do not collide.

Proof. The proof for collision avoidance within a chain is same as Lemma 2 for a single Door case. In the Look phase of the robots, if a Leader robot encounters another Leader robot within its visibility range, one of the robots gets dominated and will choose a target such that it does not lie on any path (of maximum length of 5 hops) connecting the two Leader robots. Thus, any two Leader robots avoid collision with each other. A Leader robot avoids collision with robots in another chain by avoiding cutting through a chain. When a leader robot finds a free vertex \( v \), it checks every vertices in \( N^2_v \cap V^2_{r_L} \). If at least one of the vertex in \( N^2_v \cap V^2_{r_L} \) is occupied by a robot not in Finished state, the Leader does not move. This prevents the crossover of two chains. Hence, a leader cannot collide with a follower robot from another chain. \( \square \)

Consequently, we can state following corollary.
Fig. 5: An example execution of the MULTIND algorithm
Corollary 3. A chain does not cross itself. Moreover, two chains do not cross each other.

Lemma 10. No two robots in the Finished state occupy adjacent vertices.

Proof. A robot goes into the Finished state only after becoming a Leader. According to Lemma 8, a Leader only moves to and occupies free vertices. Hence, when a robot enters the Finished state, none of the vertices adjacent to it are occupied. □

Theorem 3. Let \( G \) be a connected graph with \( k \) Doors. Algorithm MULTIND fills an MIS of vertices in \( O(m^2) \) epochs of \( G \) under an SSYNC scheduler, without collisions, by mobile luminous robots having the following capabilities: visibility range of 5 hops, persistent storage of \( O(\log(\Delta + k)) \) bits, and \( \Delta + k + 7 \) colors.

Proof. By Lemma 9 and Lemma 10, the filled vertices in \( G \) form a MIS and the filling is done without collisions. The robots require \( O(\log(\Delta + k)) \) bits of memory to store the following: State (4 states: 2 bits), Target (directions to the target vertex: \( 2\lceil \log \Delta \rceil + \lceil \log k \rceil \) bits), NextTarget (directions to the vertex to which the robot has to move after the Target vertex is reached: \( 2\lceil \log \Delta \rceil \) bits).

The colors used by the robots are \( \Delta \) colors to show the directions to the target of the robot, that also acts as a special color to switch to the Finished state. Initially, when a robot is placed for the first time at Door, it is colored with ON. There are \( k \) numbers of WAIT colors - one for each Door. There are three additional colors (CONF, CONFC, and CONF2) for confirming that the robot saw the signaled direction and confirmations of the Predecessor or the Successor, one color MOV used during the movement, one color CONF3 to indicate that the chain is in Packed state and the OFF color.

Consider a graph where all the \( k \) vertices that are connected to Doors node form a clique. In this case, only the Leader corresponding to the highest color Door would occupy one of the nodes of the \( k \)-clique. In this particular case, only one of the Doors remain active, and robots at all other Doors would go into the Finished state. So the multi-Door case would behave as a single Door case and thus replicate the time-bound for a single Door case. Thus, from Theorem 1 it follows that, MULTIND solves the MIS Filling problem in a graph with multiple Doors in \( O(m^2) \) epochs. □

5 Discussion

On the requirement of the colors: We can assert that at least \( \Delta \) colors are required. Since the movement of robots via a port number at a node is marked by a corresponding DIR color, if we have less than \( \Delta \) colors, there exist two ports that are marked by the same color, thus resulting in a configuration with adjacent vertices occupied by robots. In the absence of \( k \) colors corresponding to the \( k \) Doors, it is impossible to have a winner among all leader robots, resulting in a collision. While forming MIS, to avoid collision, we also need to ensure that a chain of robots should not cross or overlap itself.
While we need $\Delta$ colors to communicate the port numbers, we can minimize the number of colors at the cost of increasing round complexity. We can use Hoffman encoding to reduce the number of colors used for the port numbers from $\Delta$ to a constant number of colors. Now, a sequence of colors would represent a port number instead of a particular color and increases time complexity by a factor of the size of the largest encoding.

**One hop vs Two hops movement:** Our model considers that a robot moves two hops in one LCM cycle. While moving, the color of the robot is set to MOV. However, we can easily avoid this 2 hops movement of a robot by replacing the color MOV with two colors, MOV1 and MOV2. After a robot chooses its target, it sets its color to MOV1 for the first hop and then changes its color to MOV2 before reaching the target.

6 Conclusion

In this paper, we presented and analyzed two algorithms for solving two flavors of the problem of filling a maximal independent set of vertices in an arbitrary connected graph using luminous mobile robots. The first algorithm IND for graphs with a single Door works under an asynchronous scheduler. It uses robots having three hops of visibility range, $\Delta + 8$ number of colors, and $O(\log \Delta)$ bits of persistent storage and solves the problem in $O(n^2)$ epochs. The second
algorithm, MULTIND, works in graphs with \( k (> 1) \) Doors. It forms an MIS under a semi-synchronous scheduler using robots with five hops of visibility range, \( \Delta + k + 7 \) number of colors and having \( O(\log(\Delta + k)) \) bits of persistent storage, completing in \( O(n^2) \) epochs. It is open to extending this algorithm to the generalized asynchronous scheduler. The model of attaching robot splitting Doors to a graph is a new direction in multi-robot coordination problems, and many other graph problems can be explored under the same model.

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM Journal on Computing 29(4), 1164–1188 (2000)
2. Augustine, J., Moses Jr, W.K.: Dispersion of mobile robots: A study of memory-time trade-offs. In: Proceedings of the 19th International Conference on Distributed Computing and Networking. pp. 1–10 (2018)
3. Barrameda, E.M., Das, S., Santoro, N.: Deployment of asynchronous robotic sensors in unknown orthogonal environments. In: Fekete, S.P. (ed.) Algorithmic Aspects of Wireless Sensor Networks. pp. 125–140. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
4. Barrameda, E.M., Das, S., Santoro, N.: Uniform dispersal of asynchronous finite-state mobile robots in presence of holes. In: Flocchini, P., Gao, J., Kranakis, E., Meyer auf der Heide, F. (eds.) Algorithms for Sensor Systems. pp. 228–243. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
5. Bose, K., Kundu, M.K., Adhikary, R., Sau, B.: Arbitrary pattern formation by asynchronous opaque robots with lights. Theoretical Computer Science (2020)
6. d’Angelo, G., Di Stefano, G., Klasing, R., Navarra, A.: Gathering of robots on anonymous grids and trees without multiplicity detection. Theoretical Computer Science 610, 158–168 (2016)
7. Défago, X., Potop-Butucaru, M., Raipin-Parvédy, P.: Self-stabilizing gathering of mobile robots under crash or byzantine faults. Distributed Computing 33(5), 393–421 (2020)
8. Hideg, A., Lukovszki, T.: Uniform dispersal of robots with minimum visibility range. In: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics. pp. 155–167. Springer (2017)
9. Hideg, A., Lukovszki, T.: Asynchronous filling by myopic luminous robots. In: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics. pp. 108–123. Springer (2020)
10. Hideg, A., Lukovszki, T., Forstner, B.: Filling arbitrary connected areas by silent robots with minimum visibility range. In: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics. pp. 193–205. Springer (2018)
11. Hsiang, T.R., Arkin, E.M., Bender, M.A., Fekete, S.P., Mitchell, J.S.: Algorithms for rapidly dispersing robot swarms in unknown environments. In: Algorithmic Foundations of Robotics V, pp. 77–93. Springer (2004)
12. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S., Wada, K.: Gathering on rings for myopic asynchronous robots with lights. arXiv preprint arXiv:1911.04757 (2019)
13. Kamei, S., Tixeuil, S.: An asynchronous maximum independent set algorithm by myopic luminous robots on grids. CoRR abs/2012.03399 (2020). https://arxiv.org/abs/2012.03399
14. Kshemkalyani, A.D., Ali, F.: Efficient dispersion of mobile robots on graphs. In: Proceedings of the 20th International Conference on Distributed Computing and Networking. pp. 218–227 (2019)

15. Peleg, D.: Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In: International Workshop on Distributed Computing. pp. 1–12. Springer (2005)

16. Poudel, P., Sharma, G.: Time-optimal uniform scattering in a grid. In: Hans-dah, R.C., Krishnaswamy, D., Vaidya, N.H. (eds.) Proceedings of the 20th International Conference on Distributed Computing and Networking. ICDCN 2019, Bangalore, India, January 04-07, 2019. pp. 228–237. ACM (2019). https://doi.org/10.1145/3288599.3288622

17. Poudel, P., Sharma, G.: Fast uniform scattering on a grid for asynchronous oblivious robots. In: Devismes, S., Mittal, N. (eds.) Stabilization, Safety, and Security of Distributed Systems - 22nd International Symposium, SSS 2020, Austin, TX, USA, November 18-21, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12514, pp. 211–228. Springer (2020). https://doi.org/10.1007/978-3-030-64348-5_17

18. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Complete visibility for robots with lights in o (1) time. In: International Symposium on Stabilization, Safety, and Security of Distributed Systems. pp. 327–345. Springer (2016)

19. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. 28(4), 1347–1363 (1999). https://doi.org/10.1137/0009753979628292X