



# Crash-Tolerant Perpetual Exploration with Myopic Luminous Robots on Rings

Fukuhito Ooshita    
Fukui University of Technology, Japan

Naoki Kitamura    
Osaka University, Japan



Ryota Eguchi    
Nara Institute of Science and Technology, Japan

Michiko Inoue    
Nara Institute of Science and Technology, Japan

Hirotsugu Kakugawa    
Ryukoku University, Shiga, Japan

Sayaka Kamei    
Hiroshima University, Japan

Masahiro Shibata    
Kyushu Institute of Technology, Fukuoka, Japan

Yuichi Sudo    
Hosei University, Tokyo, Japan

---

## Abstract

We investigate crash-tolerant perpetual exploration algorithms by myopic luminous robots on ring networks. Myopic robots mean that they can observe nodes only within a certain fixed distance  $\phi$ , and luminous robots mean that they have light devices that can emit a color from a set of colors. The goal of perpetual exploration is to ensure that robots, starting from specific initial positions and colors, move in such a way that every node is visited by at least one robot infinitely often. As a main contribution, we clarify the tight necessary and sufficient number of robots to realize perpetual exploration when at most  $f$  robots crash. In the fully synchronous model, we prove that  $f + 2$  robots are necessary and sufficient for any  $\phi \geq 1$ . In the semi-synchronous and asynchronous models, we prove that  $3f + 3$  (resp.,  $2f + 2$ ) robots are necessary and sufficient if  $\phi = 1$  (resp.,  $\phi \geq 2$ ).

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** mobile robots, crash faults, LCM model, exploration

**Digital Object Identifier** 10.4230/LIPIcs.OPODIS.2024.12

**Funding** *Fukuhito Ooshita*: JSPS KAKENHI 22K11903

*Naoki Kitamura*: JSPS KAKENHI 23K16838

*Hirotsugu Kakugawa*: JSPS KAKENHI 23K11059

*Sayaka Kamei*: JSPS KAKENHI 23K28037

*Masahiro Shibata*: JSPS KAKENHI 20KK0232

*Yuichi Sudo*: JSPS KAKENHI 20H04140 and 20KK0232, and JST FOREST Program JPMJFR226U

## 1 Introduction

### 1.1 Background and Motivation

In the field of distributed computing, theoretical researches on autonomous mobile robots have attracted a lot of attention. The goal of the researches is to clarify the minimum capabilities of robots to achieve a given task. For this reason, many researches consider robots with very weak capabilities, that is, they are identical (*i.e.*, robots execute the same algorithm), oblivious (*i.e.*, robots have no memory to record past history), and silent (*i.e.*, robots do not send messages explicitly). As a model of weak robots, the Look-Compute-Move (LCM) model [23] is commonly used. In the LCM model, each robot repeats a cycle of Look, Compute, and Move phases: the robot takes a snapshot in the Look phase, computes its behavior based on the snapshot in the Compute phase, and moves to a new position (if necessary) in the Move phase. By using the LCM model, solvability is clarified on various environments, various capabilities of robots, and various tasks. For example, some works focus



© Fukuhito Ooshita, Naoki Kitamura, Ryota Eguchi, Michiko Inoue, Hirotsugu Kakugawa, Sayaka Kamei, Masahiro Shibata, and Yuichi Sudo;

licensed under Creative Commons License CC-BY 4.0

28th International Conference on Principles of Distributed Systems (OPODIS 2024).

Editors: Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni; Article No. 12; pp. 12:1–12:16

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

on continuous environments (aka two- or three-dimensional Euclidean space) [13, 23, 24] or discrete environments (aka graph networks) [7, 11, 16]. Most traditional works assume oblivious robots with unlimited visibility, but recently some works consider light devices [9, 14] or limited visibility [13, 17]. As benchmark tasks, gathering [7, 23], exploration [8, 19], and pattern formation [15, 24] are mainly studied. State-of-the-art surveys are given in [12].

In this paper, we focus on myopic luminous robots in discrete environments. Recently many papers focus on myopic luminous robots because of the reasonable assumptions on the observation, communication, and memory capability [2, 3, 6, 8, 18, 19]. Myopic robots mean that they can observe nodes only within a certain fixed distance. This implies that myopic robots are less powerful than traditional robots with unlimited visibility. Luminous robots mean that they have light devices that can emit a color from a set of colors. The light color is visible to other robots and not reset at the end of each cycle, and hence the light device models a communication and memory device. This assumption is reasonable because real weak robots (such as Kilobots [22]) also have weak communication devices and small memories.

Since robots are weak, it is likely that robots crash during the execution, in particular when they work in dangerous environments. Once robots have crashed, they continue to reside in the system without moving or changing their colors, however they cannot be distinguished from correct robots with the same color. For robots with unlimited visibility in Euclidean plane, some works study crash-tolerant algorithms, for example, for gathering [1, 10], flocking [25], and complete visibility [20]. For robots with unlimited visibility in graph environments, some works study crash-tolerant algorithms for gathering [4, 5, 21]. On the other hand, for myopic luminous robots in graph environments, to the best of our knowledge, only one work [3] considers crash-tolerant algorithms. This work considers crash-tolerant gathering algorithms on line networks. Hence it is not clear what tasks can be achieved by myopic luminous robots when some robots may crash.

We focus on crash-tolerant perpetual exploration. The goal of perpetual exploration is to ensure that robots, starting from specific initial positions and colors, move in such a way that every node is visited by at least one robot infinitely often. The perpetual exploration is a benchmark task of mobile robots, and it is useful for patrolling, cleaning, disinfecting, searching, and so on. Thus, the perpetual exploration has been studied extensively. When no crash occurs, on ring networks the case of visibility one is studied in [19] and the case of visibility more than one is studied in [18]. On grid networks, many algorithms for various assumptions are proposed in [2, 6].

## 1.2 Our Contributions

We investigate crash-tolerant perpetual exploration with myopic luminous robots on ring networks. We assume that each robot observes robots on nodes within distance  $\phi$  and has a light device with  $l$  colors. In this paper, we clarify the minimum number of robots to achieve perpetual exploration when at most  $f$  robots crash. Table 1 summarizes our contributions.

First, we consider the fully synchronous (FSYNC) model. In previous works [18, 19], it is proven that, when robots do not crash, two robots are necessary for any  $\phi$  and  $l$ , and two robots are sufficient for  $\phi = 1$  and  $l = 2$ . From this fact, when at most  $f$  robots crash, we can easily obtain that  $f + 2$  robots are necessary for any  $\phi$  and  $l$ , and that  $2f + 2$  robots are sufficient for  $\phi = 1$  and  $l = 2f + 2$ . For the sufficiency, let us construct  $f + 1$  groups such that each group contains two robots, and make each group execute crash-free perpetual exploration independently by assigning different colors to groups (*i.e.*,  $l = 2f + 2$ ). This algorithm achieves perpetual exploration because at least one group can continue the

■ **Table 1** Perpetual ring exploration with myopic robots.

ref.	synchrony	visibility $\phi$	#crashed	#robots	
				necessary	sufficient
[18, 19]	FSYNC	$\geq 1$	0	2	2 (2 colors)
[19]	SSYNC, ASYNC	1	0	3	3 (2 colors)
[18]	SSYNC, ASYNC	$\geq 2$	0	2	2 (2 colors)
This	FSYNC	$\geq 1$	$f$	$f + 2$	$f + 2$ ( $3f + 2$ colors)
This	SSYNC, ASYNC	1	$f$	$3f + 3$	$3f + 3$ ( $2f + 2$ colors)
This	SSYNC, ASYNC	$\geq 2$	$f$	$2f + 2$	$2f + 2$ ( $2f + 2$ colors)

algorithm. On the other hand,  $f + 2$  robots are necessary for any  $l$  because at least two robots must remain after  $f$  robots crash. As the first main contribution, we close the gap between the necessary number and the sufficient number by proposing a crash-tolerant algorithm with  $f + 2$  robots. The algorithm uses  $l = 3f + 2$  colors, that is, it uses asymptotically the same number of colors as the trivial algorithm mentioned above.

Next, we consider the semi-synchronous (SSYNC) and asynchronous (ASYNC) models. In previous works [18, 19], it is proven that, when robots do not crash, if  $\phi = 1$  (resp.,  $\phi \geq 2$ ), three (resp., two) robots are necessary for any  $l$ , and three (resp., two) robots are sufficient for  $l = 2$ . Similarly to the FSYNC model, when at most  $f$  robots crash, we can obtain that, if  $\phi = 1$  (resp.,  $\phi \geq 2$ ),  $3f + 3$  (resp.,  $2f + 2$ ) robots are sufficient for  $l = 2f + 2$ . As the second main contribution of this paper, we prove that, different from the FSYNC model, these sufficient numbers are also necessary numbers for any  $l$ . That is, the trivial algorithms are optimal in terms of the number of robots even if we can use any number of colors. To prove this, we show that, if  $x$  robots are necessary when no robot crashes,  $(f + 1)x$  robots are necessary when at most  $f$  robots crash. Intuitively, this is because robots cannot distinguish crashed robots and slow robots. When some robot  $r$  seems crashed, other robots must leave  $r$  and continue exploration. However, if robots leave  $r$  alone and  $r$  is just slow, this means that robots abandon one correct robot. To avoid this situation, robots leave a group of  $x$  robots so that the group does not get stuck. This implies that, if  $f$  robots crash dispersedly,  $f + 1$  groups of  $x$  robots can be created. For this reason  $(f + 1)x$  robots are necessary.

## 2 Preliminaries

In this section, we describe the system model and terminologies used in this paper. Most definitions are the same as [18, 19] except for those of crash faults.

### 2.1 System model

The system consists of robots and an environment which is modeled by an  $n$ -node undirected ring  $G = (V, E)$ , where  $V = \{v_0, \dots, v_{n-1}\}$  and  $E = \{(v_i, v_{i+1 \bmod n}) \mid 0 \leq i \leq n-1\}$ . For simplicity we consider mathematical operations on node indices as operations modulo  $n$ . We say  $v_i$  and  $v_{i+1}$  are neighbors. We use  $v_0, \dots, v_{n-1}$  for notation purposes only and no robot has access to indices of nodes. The ring is unoriented, that is, robots cannot recognize the direction of the ring. Robots do not know  $n$ , the size of the ring. When a robot  $r$  is on a node  $v$ , we say  $r$  occupies  $v$ . The distance between two nodes is the number of links in a shortest path between the nodes. The distance between two robots is the distance between the two nodes occupied by them. Two robots are neighbors if the nodes occupied by them are neighbors.

Robots are *luminous*, that is, each robot has a light (or state) that is visible to itself and other robots. A robot can choose the color of its light from a set  $Col$ . The number of available colors is denoted by  $l = |Col|$ . Robots have no other persistent memory. Each robot can communicate by observing positions and colors of other robots (for collecting information), and by changing its color and moving (for sending information). Robots are *myopic*, that is, each robot can observe positions and colors of robots within a fixed distance  $\phi$  ( $\phi > 0$ ) from its current position. We assume that robots share the same  $\phi$ . Robots execute the same deterministic algorithm and their behaviors depend on only their observations including their own colors.

Each robot executes an algorithm by repeating a cycle of Look, Compute, and Move phases. At the beginning of the *Look* phase, the robot takes a snapshot of positions and colors of robots within distance  $\phi$ . During the *Compute* phase, the robot computes its next color and movement according to the snapshot in the Look phase. The robot may change its color at the end of the Compute phase. If the robot decides to move, it moves to a neighboring node during the *Move* phase. Similarly to most works for graph environments, we assume that moves are instantaneous, that is, all robots exist on nodes in a snapshot. To model asynchrony of executions, we introduce the notion of *scheduler* that decides when each robot executes phases. When the scheduler makes robot  $r$  execute some phase, we say the scheduler activates the phase of  $r$  or simply activates  $r$ . We consider three types of synchronicity: the FSYNC (fully synchronous) model, the SSYNC (semi-synchronous) model, and the ASYNC (asynchronous) model. In all models, time is represented by an infinite sequence of instants  $0, 1, 2, \dots$ . No robot has access to this global time. In the FSYNC model, the scheduler activates all robots at every instant  $t$ , and all robots execute a full cycle of Look, Compute, and Move phases synchronously and concurrently between  $t$  and  $t + 1$ . In the SSYNC model, the scheduler activates a non-empty subset of robots at every instant  $t$ , and all the activated robots execute a full cycle of Look, Compute, and Move phases synchronously and concurrently between  $t$  and  $t + 1$ . In the ASYNC model, the scheduler activates a non-empty subset of robots at every instant  $t$ , and every activated robot executes one of Look, Compute, and Move phases. If robot  $r$  executes a Look phase and another robot  $r'$  executes a Compute or Move phase at instant  $t$ ,  $r$  obtains a snapshot before  $r'$  changes its color or its position. Note that, in the ASYNC model, the amount of time between two successive phases of a robot is finite but unbounded: For example, after some robot  $r$  executes a Look phase, some other robot  $r'$  can execute an unbounded number of Look, Compute, and Move phases before  $r$  executes its next Compute phase. This implies that, in the ASYNC model, a robot can move based on an outdated view obtained during the Look phase in the past configuration. Throughout the paper we assume that the scheduler is *fair*, that is, each robot is activated infinitely often.

During execution of an algorithm, at most  $f$  robots crash. A robot can crash between two consecutive phases in all of the FSYNC, SSYNC, and ASYNC models. If a robot crashes, the robot neither changes its color nor its position after it crashes. Even after robot  $r$  crashes, other robots still observe  $r$  and the color of  $r$ , however they cannot distinguish  $r$  from a correct robot with the same color as  $r$ .

In the sequel,  $M_i(t)$  denotes the multiset of colors of robots located in node  $v_i$  at an instant  $t$ . If  $v_i$  is not occupied by any robot at  $t$ , then  $M_i(t) = \emptyset$  holds. Let  $Crashed(t)$  be the set of crashed robots at an instant  $t$ . A *configuration* of the system at an instant  $t$  is defined as  $\gamma(t) = (Crashed(t), M_0(t), M_1(t), \dots, M_{n-1}(t))$ . If  $t$  is clear from the context, we simply write  $\gamma = (Crashed, M_0, M_1, \dots, M_{n-1})$ .

When a robot takes a snapshot of its environment, it gets a *view* up to distance  $\phi$ . Consider a robot  $r$  on node  $v_i$ ; then,  $r$  obtains two views: the forward view and the backward view. The forward and backward views of  $r$  are defined as  $V_f = (c_r, M_{i-\phi}, \dots, M_{i-1}, M_i, M_{i+1}, \dots, M_{i+\phi})$  and  $V_b = (c_r, M_{i+\phi}, \dots, M_{i+1}, M_i, M_{i-1}, \dots, M_{i-\phi})$ , respectively, where  $c_r$  denotes  $r$ 's color. Since robots cannot recognize the direction, they cannot recognize which of the two views is forward and which is backward. If the forward view and the backward view of  $r$  are identical, then  $r$ 's view is *symmetric*. In this case,  $r$  cannot distinguish between the two directions when it moves, and the scheduler decides which direction  $r$  moves to. Note that the scheduler can independently decide the direction regardless of any other movements. If  $r$  observes no other robot in its view,  $r$  is *isolated*.

## 2.2 Execution, problem, and exploration problem

In the FSYNC and SSYNC models, we say that an infinite sequence of configurations  $E = \gamma_0, \gamma_1, \dots, \gamma_i, \dots$  is an execution from initial configuration  $\gamma_0$  if, for any  $j > 0$ ,  $\gamma_j$  is obtained from  $\gamma_{j-1}$  after every non-crashed robot activated in instant  $j - 1$  executes a cycle. In the ASYNC model, we say that an infinite sequence of configurations  $E = \gamma_0, \gamma_1, \dots, \gamma_i, \dots$  is an execution from initial configuration  $\gamma_0$  if (i) for any  $j > 0$ ,  $\gamma_j$  is obtained from  $\gamma_{j-1}$  after every non-crashed robot activated in instant  $j - 1$  executes a Look, Compute, or Move phase, and (ii) every robot repeats Look, Compute, and Move phases in this order.

A problem  $\mathcal{P}$  is defined as a set of executions: An execution  $E$  solves  $\mathcal{P}$  if  $E \in \mathcal{P}$  holds. An algorithm  $\mathcal{A}$  solves problem  $\mathcal{P}$  from initial configuration  $\gamma_0$  if any execution from  $\gamma_0$  solves  $\mathcal{P}$ . An algorithm  $\mathcal{A}$  solves problem  $\mathcal{P}$  if there exists a configuration  $\gamma_0$  such that  $\mathcal{A}$  solves  $\mathcal{P}$  from  $\gamma_0$ .

In this paper, we consider the perpetual exploration problem.

► **Definition 1 (Perpetual exploration problem).** *Perpetual exploration is defined as a set of executions  $E$  such that every node is visited by at least one robot infinitely often in  $E$ .*

Since robots do not know  $n$ , they should explore a ring with any number of nodes in principle. However some algorithms for large rings cannot work on small rings, and consequently many works assume the minimum size of rings to propose algorithms. In this paper, we also assume the minimum size of rings to propose algorithms. On the other hand, for the impossibility, we prove that no algorithm exists for large rings.

## 3 The FSYNC model

In this section, we consider crash-tolerant perpetual exploration in the FSYNC model.

### 3.1 Impossibility

First, we show the necessary number of robots to achieve crash-tolerant perpetual exploration.

► **Theorem 2.** *If the number of robots is smaller than  $f + 2$ , the robots cannot achieve perpetual exploration regardless of the number of colors when at most  $f$  robots crash.*

**Proof.** Consider a sufficiently large ring. Assume that there exists an algorithm that solves perpetual exploration with  $f + 1$  robots. Assume that initially  $f$  robots crash. Then only one correct robot moves, and let  $r_c$  be the correct robot. Since we consider a sufficiently large ring, there exist  $2\phi + 2$  consecutive nodes  $v_{i-\phi}, \dots, v_i, v_{i+1}, \dots, v_{i+\phi+1}$  where no crashed robots exist. To achieve perpetual exploration,  $r_c$  should eventually visit  $v_i$ . At that time,  $r_c$

■ **Algorithm 1** Crash-tolerant perpetual exploration with  $f + 2$  robots with  $3f + 2$  colors. Initially  $F_0, S_1, S_2, \dots, S_f$  stay on a node  $w$  and  $B_0$  stays on a node neighboring to  $w$ . The view of  $r$  consists of  $c_r$  (its color),  $C_0$  (the set of colors in its current node), and  $C_{-1}$  and  $C_1$  (the sets of colors in its two neighboring nodes).

---

```

1: // A cycle of robot  $r$  with view  $(c_r, C_{-1}, C_0, C_1)$ 
2:  $C_{all} \leftarrow C_{-1} \cup C_0 \cup C_1$ ,  $m \leftarrow \max\{i \mid F_i \in C_{all} \vee B_i \in C_{all}\}$ 
3: if  $c_r = S_i$  for some  $i$  then
4:   if  $F_m \in C_0 \wedge \exists j \in \{-1, 1\}[B_m \in C_j]$  then
5:     move toward  $C_{-j}$  ▷ Rule S1
6:   else if  $\exists j \in \{-1, 1\}[B_m \in C_j \wedge F_m \notin C_j]$  then
7:     change the color to  $S_{i-1}$  (resp.,  $F_m$ ) if  $i \geq 2$  (resp.,  $i = 1$ ) ▷ Rule S2
8:   else
9:     change the color to  $S_{i-1}$  (resp.,  $F_{m+1}$ ) if  $i \geq 2$  (resp.,  $i = 1$ ) ▷ Rule S3
10:  end if
11: else if  $c_r = F_m$  then
12:   if  $\exists j \in \{-1, 1\}[B_m \in C_j]$  then
13:     move toward  $C_{-j}$  ▷ Rule F1
14:   else if  $B_m \notin C_{-1} \cup C_1$  then
15:     change the color to  $B_{m+1}$  and move toward  $C_{-1}$  or  $C_1$  ▷ Rule F2
16:   end if
17: else if  $c_r = B_m$  then
18:   if  $\exists j \in \{-1, 1\}[F_m \in C_{-j}]$  then
19:     move toward  $C_{-j}$  ▷ Rule B1
20:   else if  $F_m \in C_0$  then
21:     change the color to  $B_{m+1}$  ▷ Rule B2
22:   end if
23: end if

```

---

is isolated and consequently its view is symmetric. Hence, if  $r_c$  moves, the scheduler can decide the direction and makes  $r_c$  move to  $v_{i+1}$ . Similarly, since  $r_c$  is isolated on  $v_{i+1}$ , the scheduler makes  $r_c$  move to  $v_i$  if  $r_c$  moves. This implies that  $r_c$  can visit only  $v_i$  and  $v_{i+1}$  after that. This is a contradiction. ◀

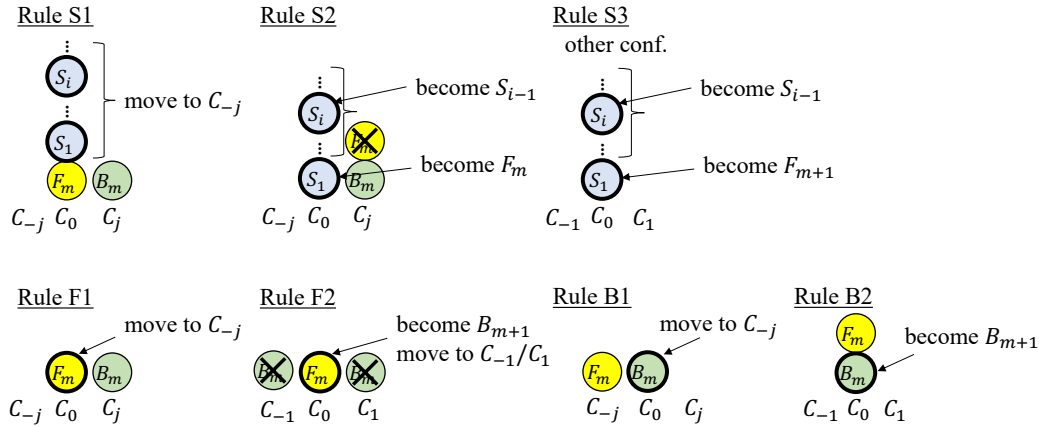
## 3.2 Possibility

In this subsection, we propose an algorithm with  $f + 2$  robots in case of visibility  $\phi = 1$ . From Theorem 2, this algorithm is optimal in terms of the number of robots.

### 3.2.1 An algorithm

The algorithm uses  $f + 2$  robots with  $3f + 2$  colors, and achieves perpetual exploration on rings when  $n \geq 4$ . The set of colors is divided into  $f + 1$  forward colors,  $f + 1$  backward colors, and  $f$  spare colors. Forward colors are represented by  $F_i$  ( $0 \leq i \leq f$ ), backward colors are represented by  $B_i$  ( $0 \leq i \leq f$ ), and spare colors are represented by  $S_i$  ( $1 \leq i \leq f$ ). We refer a robot with a forward, backward, and spare color as a forward, backward, and spare robot, respectively. For simplicity, we refer a robot with color  $X$  as robot  $X$ . We give the details of the algorithm in Algorithm 1. We also draw the behavior in Fig. 1.

Initially we deploy  $f + 2$  robots as follows. Put robot  $F_0$  and  $f$  robots  $S_1, \dots, S_f$  on a single node, and on its neighboring node put robot  $B_0$ . That is, a forward robot and a backward robot occupy two neighboring nodes, and other robots are spare robots and stay with a forward robot (see the case of  $m = 0$  in Fig. 2a). In this algorithm, all robots move



■ **Figure 1** Behaviors of robots. A robot with a cross mark means that the robot does not exist in the specified node. In Rules F1, F2, B1, and B2, we omit spare robots because the behavior of robots does not depend on spare robots.

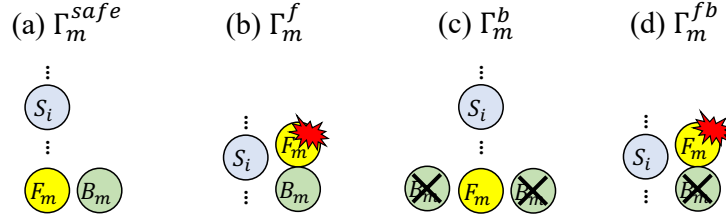
based on the forward and backward robots. In normal configurations the forward robot and spare robots move away from the backward robot (Rules F1 and S1), and the backward robot moves toward the forward robot (Rule B1). Unless either the forward or backward robot crashes, robots can maintain the same formation and continue exploration.

If the forward or backward robot crashes, robots create a new forward or backward robot. Note that robots can detect the crashes because the formation becomes different from the correct one. When robots create new forward and backward robots, we use indices of forward and backward colors to recognize new ones. Recall that initially the forward robot is  $F_0$  and the backward robot is  $B_0$ . If at least one of them crashes, we make a new forward robot  $F_1$  and a new backward robot  $B_1$ . Similarly, if forward robot  $F_m$  or backward robot  $B_m$  crashes, we make a new forward robot  $F_{m+1}$  and a new backward robot  $B_{m+1}$ . This means that forward and backward robots with smaller indices have crashed, and consequently correct robots can ignore such robots.

In principle, robots create new forward and backward robots as follows. If only forward robot  $F_m$  (see Fig. 2b), backward robot  $B_m$  becomes  $B_{m+1}$  (Rule B2) and spare robot  $S_1$  becomes  $F_{m+1}$  (Rule S3). If only backward robot  $B_m$  crashes (see Fig. 2c), forward robot  $F_m$  becomes  $B_{m+1}$  and changes its position (Rule F2) and spare robot  $S_1$  becomes  $F_{m+1}$  (Rule S3). If both forward robot  $F_m$  and backward robot  $B_m$  crash (see Fig. 2d), spare robot  $S_1$  becomes  $F_{m+1}$  (Rule S3). In the last case, the resultant configuration is similar to the configuration where robot  $B_{m+1}$  crashes when the forward and backward robots are  $F_{m+1}$  and  $B_{m+1}$ , and consequently robots create  $F_{m+2}$  and  $B_{m+2}$  similarly to the first case. Note that, during these movements, some robot such as  $S_1$  can additionally crash. To treat this situation, when robots execute the above actions, every spare robot  $S_i$  becomes  $S_{i-1}$  at the same time (Rules S2 and S3). This maintains that all correct spare robots have different indices. Since at most  $f - 1$  spare robots can crash (when a forward or backward robot crashes), eventually exactly one correct spare robot becomes  $S_1$  and then becomes  $F_{m+1}$ . We treat all possible situations in the algorithm.

### 3.2.2 Correctness

In the following, we prove the correctness of Algorithm 1. Let  $m(\gamma)$  be the maximum index of forward or backward colors that exist in configuration  $\gamma$ . For configuration  $\gamma$ , let  $f(\gamma)$  be the number of crashed robots, and let  $f'(\gamma)$  be the number of crashed robots other than



■ **Figure 2** Four types of legitimate configurations. A robot with a cross mark means that the robot does not exist in the specified node. A robot with another red mark means that the robot has crashed.

$F_{m(\gamma)}$  and  $B_{m(\gamma)}$ . As we will show, we always have  $m(\gamma) \leq f'(\gamma) \leq f(\gamma)$  and consequently the maximum index  $f$  of forward and backward colors is sufficient. Next we define *legitimate* configurations, which specify necessary conditions of configurations reachable from the initial configuration.

► **Definition 3.** Configuration  $\gamma$  is legitimate iff all the following conditions hold in  $\gamma$ :

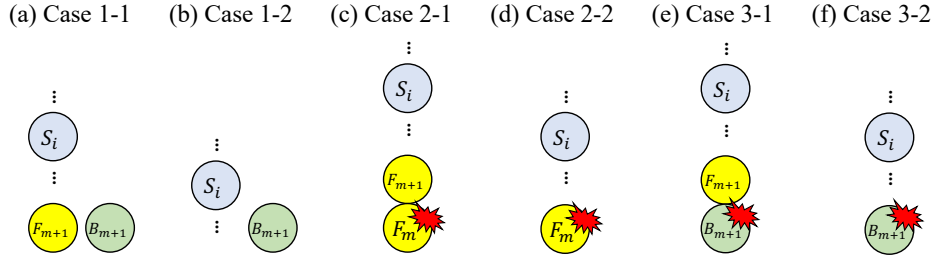
1.  $m(\gamma) \leq f'(\gamma)$ ,
2. all correct robots stay on a single node or two neighboring nodes,
3. each correct robot has a color in  $\{S_i \mid 1 \leq i \leq f\} \cup \{F_{m(\gamma)}, B_{m(\gamma)}\}$ ,
4. all correct spare robots have different colors and stay on a single node,
5. at most one robot has color  $F_{m(\gamma)}$ ,
6. at most one robot has color  $B_{m(\gamma)}$ ,
7. if both correct robot  $F_{m(\gamma)}$  and a correct spare robot exist, then they stay on the same node, and
8. if both correct robot  $B_{m(\gamma)}$  and a correct spare robot exist, they stay on different but neighboring nodes.

In the following, we define four types of legitimate configurations  $\Gamma_m^{safe}$ ,  $\Gamma_m^f$ ,  $\Gamma_m^b$ , and  $\Gamma_m^{fb}$ . Intuitively  $\Gamma_m^{safe}$  includes configurations such that robots can continue exploration based on robots  $F_m$  and  $B_m$ . This configuration changes one in  $\Gamma_m^f$ ,  $\Gamma_m^b$ , and  $\Gamma_m^{fb}$  if  $F_m$ ,  $B_m$ , and both of them crash, respectively. Note that, since  $n \geq 4$ , correct spare robots and correct forward robot  $F_m$  do not observe crashed backward robot  $B_m$ . The configurations are illustrated in Fig. 2.

► **Definition 4.** We define configurations  $\Gamma_m^{safe}$ ,  $\Gamma_m^f$ ,  $\Gamma_m^b$ , and  $\Gamma_m^{fb}$  as follows.

- Configuration  $\gamma$  is in  $\Gamma_m^{safe}$  iff 1)  $\gamma$  is legitimate, 2)  $m = m(\gamma)$  holds, 3) forward robot  $F_m$  and correct spare robots stay on the same node, and 4) backward robot  $B_m$  stays on a neighboring node of  $F_m$ .
- Configuration  $\gamma$  is in  $\Gamma_m^f$  iff 1)  $\gamma$  is legitimate, 2)  $m = m(\gamma)$  holds, 3) forward robot  $F_m$  has crashed and stays on the same node as  $B_m$ , and 4) backward robot  $B_m$  stays on a neighboring node of correct spare robots.
- Configuration  $\gamma$  is in  $\Gamma_m^b$  iff 1)  $\gamma$  is legitimate, 2)  $m = m(\gamma)$  holds, 3) forward robot  $F_m$  stays on the same node as correct spare robots, 4) if backward robot  $B_m$  exists, it has crashed and does not stay on the neighboring nodes of  $F_m$ , and 5) if backward robot  $B_m$  does not exist,  $m + 1 \leq f'(\gamma)$  holds.
- Configuration  $\gamma$  is in  $\Gamma_m^{fb}$  iff 1)  $\gamma$  is legitimate, 2)  $m = m(\gamma)$  holds, 3) forward robot  $F_m$  has crashed and stays on a node neighboring to correct spare robots, and 4) backward robot  $B_m$  has crashed and does not stay on the neighboring nodes of correct spare robots.





■ **Figure 3** Cases in the proof of Lemma 7.

An initial configuration is in  $\Gamma_0^{safe}$ . In Lemma 5, we prove that, if a configuration is in  $\Gamma_m^{safe}$  for some  $m$ , robots can keep a configuration in  $\Gamma_m^{safe}$  and continue exploration unless forward robot  $F_m$  or backward robot  $B_m$  crashes. If forward robot  $F_m$ , backward robot  $B_m$ , or both of them crash in  $\Gamma_m^{safe}$ , the configuration becomes one in  $\Gamma_m^f$ ,  $\Gamma_m^b$ , or  $\Gamma_m^{fb}$ , respectively. In Lemmas 7, 8, and 9, we prove that, from these configurations, robots reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ . Since at most  $f$  robots crash, after some configuration no new crash occurs forever, which implies robots achieve perpetual exploration.

► **Lemma 5.** Consider  $\gamma \in \Gamma_m^{safe}$  for some  $m$  and let  $\gamma'$  be a next configuration of  $\gamma$ . If neither  $F_m$  nor  $B_m$  crashes between  $\gamma$  and  $\gamma'$ ,  $\gamma'$  is in  $\Gamma_m^{safe}$  and each correct robot moves in the direction from  $B_m$  to  $F_m$  between  $\gamma$  and  $\gamma'$ .

**Proof.** In  $\gamma$ , spare robots can execute Rule S1, forward robot  $F_m$  can execute Rule F1, and backward robot  $B_m$  can execute Rule B1. Since neither  $F_m$  nor  $B_m$  crashes, all correct robots move in the direction from  $B_m$  to  $F_m$  and the resultant configuration  $\gamma'$  is clearly in  $\Gamma_m^{safe}$ . ◀

Lemma 5 implies that, if robots reach a configuration in  $\Gamma_m^f$ ,  $\Gamma_m^b$ , or  $\Gamma_m^{fb}$ , at least one of  $F_0$  and  $B_0$  has crashed. This implies that, in these configurations, at least one spare robot never crashes because the initial number of spare robots is  $f$  and at most  $f$  robots crash. Hence the following corollary holds.

► **Corollary 6.** Consider  $\gamma \in \Gamma_m^f \cup \Gamma_m^b \cup \Gamma_m^{fb}$  for some  $m$ . After  $\gamma$ , at least one spare robot never crashes.

► **Lemma 7.** Consider  $\gamma \in \Gamma_m^b$  for some  $m$ . After  $\gamma$ , robots eventually reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ .

**Proof.** From the definition of legitimate configurations,  $m \leq f'(\gamma)$  holds. Recall that, from the definition of  $\Gamma_m^b$ , either  $B_m$  exists (and has crashed) in  $\gamma$  or  $m+1 \leq f'(\gamma)$  holds. In  $\gamma$ , spare robots can execute Rule S3 and forward robot  $F_m$  can execute Rule F2. Note that from Corollary 6 at least one spare robot never crashes. Let  $\gamma'$  be a next configuration of  $\gamma$ . We consider three cases depending on the behavior of  $F_m$  between  $\gamma$  and  $\gamma'$ . The configurations are illustrated in Fig. 3. For all cases, we prove that robots reach a configuration in  $\Gamma_{m+1}^{safe}$  or  $\Gamma_{m+1}^b$ .

- Case 1:  $F_m$  does not crash between  $\gamma$  and  $\gamma'$ . In this case,  $F_m$  changes its color to  $B_{m+1}$  and moves to its neighboring node by Rule F2.
  - Case 1-1: Some spare robot becomes  $F_{m+1}$  at the same time. In this case  $m(\gamma') = m+1$  holds. If  $B_m$  exists and has crashed in  $\gamma$ , we have  $f'(\gamma') \geq f'(\gamma) + 1$  (because  $B_m$  is not counted in  $f'(\gamma)$  but counted in  $f'(\gamma')$ ) and hence  $m(\gamma') = m+1 \leq f'(\gamma) + 1 \leq f'(\gamma')$  holds. Otherwise,  $m(\gamma') = m+1 \leq f'(\gamma) \leq f'(\gamma')$  holds. Since  $m(\gamma') \leq f'(\gamma')$  holds and we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^{safe}$  holds.

## 12:10 Crash-Tolerant Perpetual Exploration with Myopic Luminous Robots on Rings

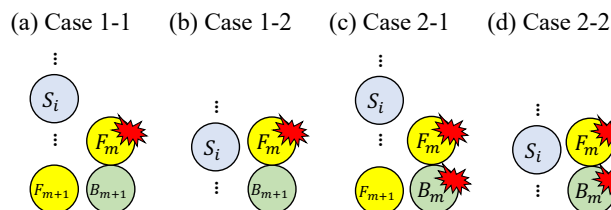
- Case 1-2: No spare robot becomes  $F_{m+1}$ . In this case,  $B_{m+1}$  cannot execute any rule and spare robots can execute rule S2 in  $\gamma'$ . Hence eventually some spare robot becomes  $F_{m+1}$  in some configuration  $\gamma''$ . Similarly to the above case,  $\gamma'' \in \Gamma_{m+1}^{safe}$  holds.
  - Case 2:  $F_m$  crashes before it changes its color.
    - Case 2-1: Some spare robot becomes  $F_{m+1}$  in  $\gamma'$ . In this case,  $m(\gamma') = m + 1$  holds. If  $B_m$  exists and has crashed in  $\gamma$ , since  $F_m$  also crashes, we have  $f'(\gamma') \geq f'(\gamma) + 2$  and hence  $m(\gamma') + 1 = m + 2 \leq f'(\gamma) + 2 \leq f'(\gamma')$ . Otherwise,  $m + 1 \leq f'(\gamma)$  holds, and, since  $F_m$  crashes between  $\gamma$  and  $\gamma'$ , we have  $f'(\gamma') \geq f'(\gamma) + 1 \geq m + 2 = m(\gamma') + 1$ . Hence  $m(\gamma') + 1 \leq f'(\gamma')$  holds, and, since we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^b$  holds.
    - Case 2-2: No spare robot becomes  $F_{m+1}$ . In this case, spare robots can execute rule S3 in  $\gamma'$ . Hence eventually some spare robot becomes  $F_{m+1}$  in some configuration  $\gamma''$ . Similarly to the above case,  $\gamma'' \in \Gamma_{m+1}^b$  holds.
- Case 3:  $F_m$  crashes after it changes its color to  $B_{m+1}$  before it moves.
  - Case 3-1: Some spare robot becomes  $F_{m+1}$  at the same time. In this case,  $m(\gamma') = m + 1$  holds. If  $B_m$  exists and has crashed in  $\gamma$ ,  $f'(\gamma') \geq f'(\gamma) + 1$  (because  $B_m$  is not counted in  $f'(\gamma)$  but counted in  $f'(\gamma')$ ) and hence  $m(\gamma') = m + 1 \leq f'(\gamma) + 1 \leq f'(\gamma')$  holds. Otherwise,  $m(\gamma') = m + 1 \leq f'(\gamma) \leq f'(\gamma')$  holds. Hence  $m(\gamma') \leq f'(\gamma')$  holds, and, since  $B_{m+1}$  has crashed and we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^b$  holds.
  - Case 3-2: No spare robot becomes  $F_{m+1}$ . In this case, spare robots can execute rule S3. Hence eventually some spare robot becomes  $F_{m+1}$  in some configuration  $\gamma''$ . Similarly to the above case,  $\gamma'' \in \Gamma_{m+1}^b$  holds.

For all cases, we have proved that robots reach a configuration in  $\Gamma_{m+1}^{safe}$  or  $\Gamma_{m+1}^b$ . If they reach a configuration in  $\Gamma_{m+1}^b$ , by the same discussion they reach a configuration in  $\Gamma_{m+2}^{safe}$  or  $\Gamma_{m+2}^b$ . We can repeat this discussion, however robots change the configuration from one in  $\Gamma_{m+i}^b$  for some  $i$  to one in  $\Gamma_{m+i+1}^b$  only if  $F_{m+i}$  crashes. Since the number of crashes is at most  $f$ , eventually robots reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ . ◀

► **Lemma 8.** *Consider  $\gamma \in \Gamma_m^f$  for some  $m$ . After  $\gamma$ , robots eventually reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ .*

**Proof.** From the definition of legitimate configurations,  $m \leq f'(\gamma)$  holds. In  $\gamma$ , spare robots can execute Rule S3 and backward robot  $B_m$  can execute Rule B2. Note that from Corollary 6 at least one spare robot never crashes. Let  $\gamma'$  be a next configuration of  $\gamma$ . We consider two cases depending on the behavior of  $B_m$  between  $\gamma$  and  $\gamma'$ . The configurations are illustrated in Fig. 4. For all cases, we prove that robots reach a configuration in  $\Gamma_{m+1}^{safe}$  or  $\Gamma_{m+1}^b$ . This derives the lemma because from a configuration in  $\Gamma_{m+1}^b$  robots reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m + 1$  by Lemma 7.

- Case 1:  $B_m$  does not crash between  $\gamma$  and  $\gamma'$ . In this case,  $B_m$  changes its color to  $B_{m+1}$ .
  - Case 1-1: Some spare robot becomes  $F_{m+1}$  at the same time. In this case  $m(\gamma') = m + 1$  holds. Since  $F_m$  has crashed in  $\gamma$ , we have  $f'(\gamma') \geq f'(\gamma) + 1$  because  $F_m$  is not counted in  $f'(\gamma)$  but counted in  $f'(\gamma')$ . Hence  $m(\gamma') = m + 1 \leq f'(\gamma) + 1 \leq f'(\gamma')$  holds, and, since we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^{safe}$  holds.
  - Case 1-2: No spare robot becomes  $F_{m+1}$ . In this case,  $B_{m+1}$  cannot execute any rule and spare robots can execute rule S2. Hence eventually some spare robot becomes  $F_{m+1}$  in some configuration  $\gamma''$ . Similarly to the above case,  $\gamma'' \in \Gamma_{m+1}^{safe}$  holds.



■ **Figure 4** Cases in the proof of Lemma 8.

- Case 2:  $B_m$  crashes before it changes its color.
  - Case 2-1: Some spare robot becomes  $F_{m+1}$  in  $\gamma'$ . In this case,  $m(\gamma') = m + 1$  holds. Since  $F_m$  and  $B_m$  have crashed in  $\gamma'$ , we have  $f'(\gamma') \geq f'(\gamma) + 2$  because  $F_m$  and  $B_m$  are not counted in  $f'(\gamma)$  but counted in  $f'(\gamma')$ . Hence  $m(\gamma') + 1 = m + 2 \leq f'(\gamma) + 2 \leq f'(\gamma')$  holds, and, since we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^b$  holds.
  - Case 2-2: No spare robot becomes  $F_{m+1}$ . In this case, spare robots can execute rule S3. Hence eventually some spare robot becomes  $F_{m+1}$  in some configuration  $\gamma''$ . Similarly to the above case,  $\gamma'' \in \Gamma_{m+1}^b$  holds. ◀

► **Lemma 9.** Consider  $\gamma \in \Gamma_m^{fb}$  for some  $m$ . After  $\gamma$ , robots eventually reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ .

**Proof.** From the definition of legitimate configurations,  $m \leq f'(\gamma)$  holds. In  $\gamma$ , spare robots can execute Rule S3. Note that from Corollary 6 at least one spare robot never crashes. Hence eventually some spare robot becomes  $F_{m+1}$  in  $\gamma'$ . Now  $m(\gamma') = m + 1$  holds. Since  $F_m$  and  $B_m$  have crashed in  $\gamma$ , we have  $f'(\gamma') \geq f'(\gamma) + 2$  because  $F_m$  and  $B_m$  are not counted in  $f'(\gamma)$  but counted in  $f'(\gamma')$ . Hence  $m(\gamma') + 1 = m + 2 \leq f'(\gamma) + 2 \leq f'(\gamma')$  holds, and, since we can easily check other conditions,  $\gamma' \in \Gamma_{m+1}^b$  holds. This derives the lemma because from a configuration in  $\Gamma_{m+1}^b$  robots reach one in  $\Gamma_{m'}^{safe}$  for some  $m' > m + 1$  by Lemma 7. ◀

► **Theorem 10.** Algorithm 1 solves perpetual exploration on rings with  $n \geq 4$  in the FSYNC model even if at most  $f$  robots crash. It uses  $f + 2$  robots and  $3f + 2$  colors.

**Proof.** The initial configuration is in  $\Gamma_0^{safe}$ . If  $F_m$  or  $B_m$  crashes in a configuration in  $\Gamma_m^{safe}$  for some  $m$ , robots reach a configuration in  $\Gamma_m^f$ ,  $\Gamma_m^b$ , or  $\Gamma_m^{fb}$ . However, by Lemmas 7, 8, and 9, they eventually reach a configuration in  $\Gamma_{m'}^{safe}$  for some  $m' > m$ . Since at most  $f$  robots crash, eventually they keep configurations in  $\Gamma_{m''}^{safe}$  for some  $m''$  forever. Hence, by Lemma 5, robots achieve perpetual exploration. From the definition, the algorithm uses  $f + 2$  robots and  $3f + 2$  colors. ◀

## 4 The SSYNC and ASYNC models

In this section, we consider crash-tolerant perpetual exploration in the SSYNC and ASYNC models.

### 4.1 Possibility

As described in Section 1.2, we can design simple crash-tolerant algorithms by having  $f + 1$  teams of robots independently execute crash-free algorithms. Note that the team size depends on  $\phi$ . When  $\phi \geq 2$ , a team of two robots (*e.g.*, red and blue robots) can achieve perpetual

exploration as follows: if the two robots are neighbors, the red robot moves away from the blue robot; otherwise, the blue robot moves toward the red robot [18]. This behavior is not feasible when  $\phi = 1$ , however by adding an additional robot, they can achieve perpetual exploration as demonstrated in [19]. We formalize these facts in the following theorem.

► **Theorem 11.** *In the SSYNC and ASYNC models, when at most  $f$  robots crash,  $3f + 3$  (resp.,  $2f + 2$ ) robots with  $2f + 2$  colors can achieve perpetual exploration on rings with  $n \geq 3$  (resp.,  $n \geq 5$ ) if  $\phi = 1$  (resp.,  $\phi \geq 2$ ).*

**Proof.** In case of  $\phi = 1$ , there exists an algorithm  $A$  such that three robots with two colors can achieve perpetual exploration on rings with  $n \geq 3$  when no robot crashes [19]. Let  $Col_A$  be the set of colors used in  $A$ . Assume that, in the initial configuration of  $A$ , three robots  $r_1, r_2$ , and  $r_3$  have colors  $c_1, c_2$ , and  $c_3$  and occupy nodes  $w_1, w_2$ , and  $w_3$ , respectively. In the following, we convert algorithm  $A$  to algorithm  $A'$  that works when at most  $f$  robots crash. The set of colors used in  $A'$  is defined as  $Col_{A'} = \{0, \dots, f\} \times Col_A$  (this implies  $|Col_{A'}| = 2f + 2$ ). Algorithm  $A'$  divides  $3f + 3$  robots  $r_1, \dots, r_{3f+3}$  into  $f + 1$  groups and makes each group execute algorithm  $A$ . That is, for  $g \in \{0, \dots, f\}$  and  $i \in \{1, 2, 3\}$ ,  $A'$  makes robot  $r_{3g+i}$  have color  $(g, c_i)$  and occupy node  $w_i$  in the initial configuration. After that, for each  $g$ , three robots with colors  $(g, c)$  ( $c \in Col_A$ ) make the same behavior as  $A$  by changing the second elements of their colors. When at most  $f$  robots crash, all robots in at least one group execute algorithm  $A$  correctly. Hence  $A'$  achieves perpetual exploration.

In case of  $\phi \geq 2$ , there exists an algorithm such that two robots with two colors can achieve perpetual exploration on rings with  $n \geq 5$  when no robot crashes [18]. Similarly to the above case, we can construct a crash-tolerant algorithm. ◀

## 4.2 Impossibility

In the previous subsection, we proved that, when at most  $f$  robots crash,  $3f + 3$  (resp.,  $2f + 2$ ) robots can achieve perpetual exploration if  $\phi = 1$  (resp.,  $\phi \geq 2$ ). In this subsection, we prove that these numbers of robots are necessary. To prove this fact, we use the following lemmas for the case of no crashed robots.

► **Lemma 12** ([19]). *If the number of robots is one, the robot cannot visit more than two nodes regardless of the number of colors in the FSYNC, SSYNC, and ASYNC models.*

► **Lemma 13** ([19]). *If the number of robots is two and  $\phi = 1$  holds, the robots cannot visit more than five nodes regardless of the number of colors in the SSYNC and ASYNC models.*

Since these lemmas are almost trivial, we give brief proofs of them. If the number of robots is one, the view of the robot is always symmetric and hence the scheduler decides the direction of the movement. This makes the robot continue to move backward and forward on two neighboring nodes. If the number of robots is two and  $\phi = 1$  holds, the scheduler can make two robots separated. This is because, when two robots move in the same directions, the scheduler can activate only the head robot. Once two robots get separated, the views of the two robots are symmetric, and hence they continue to move backward and forward on neighboring nodes.

In the rest of this section, we prove the main impossibility. To prove the impossibility, we consider the behaviors of robots on an infinite line. This is because, if robots can achieve perpetual exploration in any ring, they should visit infinite nodes on an infinite line. Indeed, if robots can visit at most  $n$  nodes on an infinite line, they can also visit at most  $n$  nodes on a ring with more than  $n + \phi$  nodes and hence they cannot achieve perpetual exploration. On

an infinite line, we define a *hole* as a set of successive non-occupied nodes such that each end node is a neighbor of an occupied node. The size of a hole is defined as the number of nodes in the hole.

► **Theorem 14.** *Fix the SSYNC or ASYNC model. Fix visibility  $\phi$ . In this setting, assume that, if the number of robots is smaller than  $x$ , the robots cannot visit more than  $n_0$  nodes regardless of the number of colors when no robots crash. In the same setting, there exists an integer  $n_f$  such that, if the number of robots is smaller than  $(f + 1)x$ , the robots cannot visit more than  $n_f$  nodes regardless of the number of colors when at most  $f$  robots crash.*

**Proof.** We prove the following proposition by induction.

Proposition A: For any integer  $h \geq 0$ , there exists an integer  $n_h$  such that, if the number of robots is smaller than  $(h + 1)x$ , the robots cannot visit more than  $n_h$  nodes regardless of the number of colors when at most  $h$  robots crash.

From the assumption of the theorem, Proposition A holds for  $h = 0$ . In the following, we assume that, for some  $k \geq 1$ , the proposition holds for the case of  $h = 0, \dots, k - 1$ . To prove the case of  $h = k$ , we prove the following proposition by induction.

Proposition B: For any integer  $i$  ( $0 \leq i \leq x - 1$ ), there exists an integer  $m_i$  such that, if the number of robots is  $kx + i$ , the robots cannot visit more than  $m_i$  nodes regardless of the number of colors when at most  $k$  robots crash.

Recall that we assume Proposition A for  $h = k - 1$ . This implies that, if the number of robots is smaller than  $kx$ , the robots cannot visit more than  $n_{k-1}$  nodes when at most  $k - 1$  robots crash. Hence Proposition B holds for  $i = 0$  by assigning  $m_0 = n_{k-1}$ . In the following claim, we consider the inductive case.

▷ **Claim 15.** Assume that, for some  $\ell$  ( $1 \leq \ell \leq x - 1$ ), Proposition B holds for  $i = 0, \dots, \ell - 1$ . In this case, Proposition B holds for  $i = \ell$ .

**Proof.** For contradiction, assume that  $kx + \ell$  robots can visit any number of nodes when at most  $k$  robots crash. Let  $D = 2(\max\{n_0, \dots, n_{k-1}, m_0, \dots, m_{\ell-1}\} + \phi)$ .

Let  $r$  be some robot. Starting from an initial configuration, the scheduler activates all robots except for  $r$ . Note that such an activation is possible during an arbitrarily long period in the SSYNC and ASYNC model. Since robots can visit any number of nodes even when  $r$  has crashed, eventually a hole with size  $D$  is created. Let  $\gamma$  be the first such configuration. We divide a set of all robots into two sets  $A$  and  $B$  so that a hole with size  $D$  exists between  $A$  and  $B$  in  $\gamma$ . Without loss of generality, we assume  $1 \leq |A| \leq |B|$ . We consider two cases.

- **Case 1:** Consider the case that  $|A| \leq \ell$  holds. Consequently  $|B| = kx + j$  holds for  $j = \ell - |A| \geq 0$ . In this case, after configuration  $\gamma$ , we assume that no robots in  $A$  crash and that at most  $k$  robots in  $B$  crash. Since  $|A| < x$ , robots in  $A$  cannot visit more than  $n_0$  nodes unless they meet some robot in  $B$ . From the assumption of Claim 15 and  $|B| = kx + j$  for  $j = \ell - |A| \leq \ell - 1$ , robots in  $B$  cannot visit more than  $m_j$  nodes unless they meet some robot in  $A$ . From the definition of  $D$ , robots in  $A$  and  $B$  cannot meet, and hence the number of nodes they can visit is finite. That is, this case derives a contradiction.
- **Case 2:** Consider the case that  $|A| > \ell$  holds. Let  $a$  and  $b$  be integers such that  $|A| = \ell + ax + b$ ,  $0 \leq a$ , and  $0 \leq b \leq x - 1$ . Note that  $a > 0 \vee b > 0$  holds. Consequently  $|B| = (k - a)x - b$  holds, and we have  $a < k/2$  from  $|A| \leq |B|$ . We further consider three sub-cases depending on  $k$  and  $b$ .

- Case 2-1: Consider the case of  $k = 1$ . From  $a < k/2$ , we have  $a = 0$  and consequently  $b > 0$ . This implies that  $|A| \leq |B| = x - b < x$ . Hence, from the inductive assumption of Proposition A, robots in  $A$  (resp.  $B$ ) cannot visit more than  $n_0$  nodes unless they meet some robot in  $B$  (resp.  $A$ ). From the definition of  $D$ , robots in  $A$  and  $B$  cannot meet, and hence the number of nodes they can visit is finite. That is, this sub-case derives a contradiction.
- Case 2-2: Consider the case of  $k \geq 2$  and  $b > 0$ . In this case, after configuration  $\gamma$ , we assume that at most  $a + 1$  robots in  $A$  crash and at most  $k - a - 1$  robots in  $B$  crash. From  $a + 1 < k/2 + 1 \leq k/2 + k/2 = k$ , we have  $a + 1 \leq k - 1$  because  $a$  and  $k$  are integers. Hence, from the inductive assumption of Proposition A and  $|A| = \ell + ax + b < (a + 2)x$ , robots in  $A$  cannot visit more than  $n_{a+1}$  nodes unless they meet some robot in  $B$ . Similarly, from  $k - a - 1 \leq k - 1$  and  $|B| = (k - a)x - b < (k - a)x$ , robots in  $B$  cannot visit more than  $n_{k-a-1}$  nodes unless they meet some robot in  $A$ . From the definition of  $D$ , robots in  $A$  and  $B$  cannot meet, and hence the number of nodes they can visit is finite. That is, this sub-case derives a contradiction.
- Case 2-3: Consider the case of  $k \geq 2$  and  $b = 0$ . This implies  $a > 0$ . In this case, after configuration  $\gamma$ , we assume that at most  $a$  robots in  $A$  crash and at most  $k - a$  robots in  $B$  crash. Since  $a \leq k - 1$  holds, from the inductive assumption of Proposition A and  $|A| = \ell + ax < (a + 1)x$ , robots in  $A$  cannot visit more than  $n_a$  nodes unless they meet some robot in  $B$ . Similarly, from  $k - a \leq k - 1$  and  $|B| = (k - a)x < (k - a + 1)x$ , robots in  $B$  cannot visit more than  $n_{k-a}$  nodes unless they meet some robot in  $A$ . From the definition of  $D$ , robots in  $A$  and  $B$  cannot meet, and hence the number of nodes they can visit is finite. That is, this sub-case derives a contradiction.

Since all cases derive a contradiction, we have proved the claim.  $\triangleleft$

Since both the base case and the inductive case (Claim 15) hold, Proposition B holds for any  $i$  ( $0 \leq i \leq x - 1$ ). This implies that Proposition A holds for  $h = k$ . Hence, the theorem holds.  $\blacktriangleleft$

From Theorem 14 and Lemmas 12 and 13, we have the following corollaries.

► **Corollary 16.** *In the SSYNC and ASYNC models, if the number of robots is smaller than  $2f + 2$ , the robots cannot achieve perpetual exploration regardless of the number of colors when at most  $f$  robots crash.*

► **Corollary 17.** *In the SSYNC and ASYNC models, if the number of robots is smaller than  $3f + 3$  and  $\phi = 1$  holds, the robots cannot achieve perpetual exploration regardless of the number of colors when at most  $f$  robots crash.*

## 5 Conclusions

We investigate crash-tolerant perpetual exploration algorithms by myopic luminous robots on ring networks. As a main contribution, we clarify the tight necessary and sufficient number of robots to realize perpetual exploration when at most  $f$  robots crash. In the fully synchronous model, we prove that  $f + 2$  robots are necessary and sufficient for any  $\phi \geq 1$ . In the semi-synchronous and asynchronous models, we prove that  $3f + 3$  (resp.,  $2f + 2$ ) robots are necessary and sufficient if  $\phi = 1$  (resp.,  $\phi \geq 2$ ).

This paper leaves many interesting open issues.

- What is the minimum number of colors to achieve perpetual exploration with optimal number of robots?

- What is the minimum number of robots and colors to achieve terminating exploration? It is easy to observe that one additional robot with one additional color is sufficient to realize terminating exploration. The special robot just stops as a landmark, and other robots execute perpetual exploration from the landmark. Then, when the robots come back to the landmark from the opposite direction, they stop moving. The question is whether such an additional robot and/or an additional color is necessary for any  $f$ ?
- Is it possible to weaken the assumptions about initial positions and colors? The proposed algorithms, like those in related works, assume specific initial positions and colors. Some assumptions of initial positions are essential, as exploration is impossible if no robot observes another initially. The question is whether the problem is solvable given certain initial positions and arbitrary colors.
- It is also interesting to study other tasks such as pattern formation and complete visibility.

---

### References

- 1 Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006. doi:10.1137/050645221.
- 2 Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. Infinite grid exploration by disoriented robots. In *The 8th International Conference on Networked Systems*, pages 129–145, 2020. doi:10.1007/978-3-030-67087-0\_9.
- 3 Quentin Bramas, Hirotsugu Kakugawa, Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, Masahiro Shibata, and Sébastien Tixeuil. Stand-up indulgent gathering on lines for myopic luminous robots. In *The 38th International Conference on Advanced Information Networking and Applications*, pages 110–121, 2024. doi:10.1007/978-3-031-57853-3\_10.
- 4 Quentin Bramas, Sayaka Kamei, Anissa Lamani, and Sébastien Tixeuil. Stand-up indulgent gathering on lines. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 451–465, 2023. doi:10.1007/978-3-031-44274-2\_34.
- 5 Quentin Bramas, Sayaka Kamei, Anissa Lamani, and Sébastien Tixeuil. Stand-up indulgent gathering on rings. In *31st International Colloquium on Structural Information and Communication Complexity*, pages 119–137, 2024. doi:10.1007/978-3-031-60603-8\_7.
- 6 Quentin Bramas, Pascal Lafourcade, and Stéphane Devismes. Optimal exclusive perpetual grid exploration by luminous myopic opaque robots with common chirality. *Theoretical Computer Science*, 977:114162, 2023. doi:10.1016/J.TCS.2023.114162.
- 7 Gianlorenzo D’Angelo, Alfredo Navarra, and Nicolas Nisse. A unified approach for gathering and exclusive searching on rings under weak assumptions. *Distributed Computing*, 30(1):17–48, 2017. doi:10.1007/S00446-016-0274-Y.
- 8 Omar Darwich, Ahmet-Sefa Ulucan, Quentin Bramas, Anissa Lamani, Anaïs Durand, and Pascal Lafourcade. Perpetual torus exploration by myopic luminous robots. *Theoretical Computer Science*, 976:114143, 2023. doi:10.1016/J.TCS.2023.114143.
- 9 Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171–184, 2016. doi:10.1016/J.TCS.2015.09.018.
- 10 Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin Parvédy. Self-stabilizing gathering of mobile robots under crash or byzantine faults. *Distributed Computing*, 33(5):393–421, 2020. doi:10.1007/S00446-019-00359-X.
- 11 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013. doi:10.1007/S00453-011-9611-5.
- 12 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*. Springer, 2019. doi:10.1007/978-3-030-11072-7.

- 13 Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005. doi:10.1016/J.TCS.2005.01.001.
- 14 Paola Flocchini, Nicola Santoro, Yuichi Sudo, and Koichi Wada. On asynchrony, memory, and communication: Separations and landscapes. In *27th International Conference on Principles of Distributed Systems*, 2023. doi:10.4230/LIPICS.OPODIS.2023.28.
- 15 Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Pattern formation by oblivious asynchronous mobile robots. *SIAM Journal on Computing*, 44(3):740–785, 2015. doi:10.1137/140958682.
- 16 Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science*, 411(34-36):3235–3246, 2010. doi:10.1016/J.TCS.2010.05.020.
- 17 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, and Giovanni Viglietta. Turing-mobile: a turing machine of oblivious mobile robots with limited visibility and its applications. *Distributed Computing*, 35(2):105–122, 2022. doi:10.1007/S00446-021-00406-6.
- 18 Shota Nagahama, Fukuhito Ooshita, and Michiko Inoue. Ring exploration of myopic luminous robots with visibility more than one. *Information and Computation*, 2023. doi:10.1016/J.IC.2023.105036.
- 19 Fukuhito Ooshita and Sébastien Tixeuil. Ring exploration with myopic luminous robots. *Information and Computation*, 2022. doi:10.1016/J.IC.2021.104702.
- 20 Pavan Poudel, Aisha Aljohani, and Gokarna Sharma. Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. *Theoretical Computer Science*, 850:116–134, 2021. doi:10.1016/J.TCS.2020.10.033.
- 21 Sergio Rajsbaum, Armando Castañeda, David Flores Peñaloza, and Manuel Alcántara. Fault-tolerant robot gathering problems on graphs with arbitrary appearing times. In *2017 IEEE International Parallel and Distributed Processing Symposium*, pages 493–502, 2017. doi:10.1109/IPDPS.2017.70.
- 22 Michael Rubenstein, Christian Ahler, Nick Hoff, Adrian Cabrera, and Radhika Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975, 2014. doi:10.1016/J.ROBOT.2013.08.006.
- 23 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.
- 24 Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. *Journal of the ACM*, 64(3):16:1–16:43, 2017. doi:10.1145/3060272.
- 25 Yan Yang, Samia Souissi, Xavier Défago, and Makoto Takizawa. Fault-tolerant flocking for a group of autonomous mobile robots. *Journal of Systems and Softwares*, 84(1):29–36, 2011. doi:10.1016/J.JSS.2010.08.026.