Self-stabilizing robots in highly dynamic environments

Marjorie Bournat, Ajoy K. Datta, Swan Dubois

Version of Record: https://www.sciencedirect.com/science/article/pii/S0304397518307163
Manuscript_32d61dc30ef84f63e1b91e4374a1381d

Self-Stabilizing Robots in Highly Dynamic Environments[☆]

Marjorie Bournat[1]

Sorbonne Université, CNRS, Inria, LIP6 UMR 7606, 4 Place Jussieu, 75252 Paris cedex 05, France

Ajoy K. Datta

University of Nevada, Las Vegas, United States

Swan Dubois

Sorbonne Université, CNRS, Inria, LIP6 UMR 7606, 4 Place Jussieu, 75252 Paris cedex 05, France

Abstract

This paper deals with the classical problem of exploring a ring by a cohort of synchronous robots. We focus on the perpetual version of this problem in which it is required that each node of the ring is visited by a robot infinitely often.

The challenge in this paper is twofold. First, we assume that the robots evolve in a highly dynamic ring, *i.e.,* edges may appear and disappear unpredictably without any recurrence, periodicity, or stability assumption. The only assumption we made (known as the temporal connectivity assumption) is that each node is infinitely often reachable from any other node. Second, we aim at providing a self-stabilizing algorithm to the robots, *i.e.,* the algorithm must guarantee an eventual correct behavior regardless of the initial state and positions of the robots.

In this harsh environment, our contribution is to fully characterize, for each size of the ring, the necessary and sufficient number of robots to solve deterministically the problem.

Keywords: Highly dynamic graphs, evolving graphs, perpetual exploration, fully synchronous robots, self-stabilizing algorithm

[☆]A preliminary version of this work appears in the proceedings of the 18th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2016) [4].

This work was initiated while the second author was visiting professor at Sorbonne Université (formerly UPMC Sorbonne Universités).

Email addresses: marjorie.bournat@lip6.fr (Marjorie Bournat), ajoy.datta@unlv.edu (Ajoy K. Datta), swan.dubois@lip6.fr (Swan Dubois)

[1]Corresponding author

1. Introduction

We consider a cohort of autonomous and synchronous robots that are equipped with motion actuators and sensors, but that are otherwise unable to communicate [28]. They evolve in a *discrete environment*, represented by a graph, where the nodes represent the possible locations of robots and the edges the possibility for a robot to move from one location to another. Refer to [25] for a survey of results in this model. One fundamental problem is the *exploration* of graphs by robots. Basically, each node of the graph has to be visited by at least one robot. There exist several variants of this problem depending on whether or not the robots are required to stop once they completed the exploration of the graph or not.

Typically, the environment of the robots is modeled by a *static* undirected connected graph meaning that both vertex and edge sets do not evolve with time. In this paper, we consider *dynamic* environments that may change over time, for instance, a transportation network, a building in which doors are closed and open over time, or streets that are closed over time due to work in process or traffic jam in a town. More precisely, we consider dynamic graphs [30, 7] in which edges may appear and disappear unpredictably without any stability, recurrence, nor periodicity assumption. However, to ensure that the problem is not trivially unsolvable, we made the assumption that each node is infinitely often reachable from any other one through a *temporal path* (*a.k.a. journey* [7]). In the following, such dynamic graphs are interchangeably called *highly dynamic* or *connected-over-time*.

As in other distributed systems, *fault-tolerance* is a central issue in robot networks. Indeed, it is desirable that the misbehavior of some robots does not prevent the whole system to reach its objective. *Self-stabilization* [11, 13, 29] is a versatile technique to tolerate *transient* (*i.e.,* of finite duration) faults. After the occurrence of a catastrophic failure that may take the system to some arbitrary global state, self-stabilization guarantees recovery to a correct behavior in finite time without external (*i.e.,* human) intervention. In the context of robot networks, that implies that the algorithm must guarantee an eventual correct behavior regardless of the initial state and positions of the robots.

Our objective in this paper is twofold. First, we want to investigate for the first time the problem of exploration of a highly dynamic graph by a cohort of self-stabilizing deterministic robots. Second, we aim at characterizing, for the specific case of the ring, the necessary and sufficient number of robots to perform this task (in function of the size of the ring).

Related Work. Since the seminal work of Shannon [27], exploration of graphs by a cohort of robots has been extensively studied. There exist mainly three variants of the problem: (*i*) *exploration with stop*, where robots are required to detect the end of the exploration, then stop moving (*e.g.,* [15]); (*ii*) *exploration with return*, where robots must come back to their initial location once the exploration completed (*e.g.,* [12]); and (*iii*) *perpetual exploration*, where each node has to be infinitely often visited by some robots (*e.g.,* [1]). Even if we restrict ourselves to deterministic approaches, there exist numerous solutions to these problems depending on the topology of the graphs to explore (*e.g.,* ring-shaped [15], line-shaped [17], tree-shaped [16], or arbitrary network [8]), and the assumptions made on robots (*e.g.,* limited range of visibility [9], common sense of orientation [2], *etc.*).

Note that all the above work considers only static graphs. At the exception of a recent work on gathering [24], work on robots in dynamic graphs dealt only with the exploration problem. We review them in the following.

The first two papers [18, 20] focused on the exploration (with stop) by a single agent of so-called *PV-graphs*. A PV-graph is a very specific kind of dynamic graph in which a set of entities (called carriers) infinitely often move in a predetermined way, each of them periodically visiting a subset of nodes of the graph. An agent (controlled by the algorithm) is initially located at a node and can move from node to node only using a carrier. This is a relevant model for transportation networks. In this context, these two papers study the necessity and the sufficiency of various assumptions (like the anonymity of nodes, the knowledge of the size of the network, or of a bound on the periodicity of the carriers, *etc.*) as well as their impact on the complexity of the exploration. The main difference between these two works likes in whether the agent is able to wait a carrier on nodes [20] or not [18].

A second line of research [21, 19, 10] considers another restriction on dynamicity by targeting *T-interval-connected graphs*, *i.e.,* the graph is connected at each step and there exists a stability of this connectivity in any interval of time of length T [23]. The two first papers [21, 19] investigate an *off-line* version of the exploration meaning that the single agent knows in advance the evolution of the graph over time and uses it to compute its route before the beginning of the execution. In both papers, the authors provide lower and upper bounds on the exploration time in this context. The first one focuses on ring-shaped graphs, the second one on cactus-shaped (*i.e.,* trees of rings). Finally, [10] deals with the exploration with stop of 1-interval-connected rings by several robots. The authors study the impact of numerous assumptions (like the synchrony assumption, the anonymity of the graph, the chirality of robots, or the knowledge of some characteristic of the graph) on the solvability of the problem depending on the number of robots involved. They particularly show that these assumptions may influence the capacity of robots to *detect* the end of the exploration and hence to systematically terminate their execution or not.

In summary, previous work on exploration of dynamic graphs restricts strongly the dynamic of the considered graph. The notable exception is a recent work on perpetual exploration of highly dynamic rings [5]. This paper shows that, three (resp. two) synchronous anonymous robots are necessary and sufficient to perpetually explore highly dynamic rings of size greater than (resp. equals to) four (resp. three). Nonetheless, algorithms from [5] do not tolerate initial memory corruption nor arbitrary initial positions of robots (in particular, robots must be initially scattered). In other words, they are not self-stabilizing. Moreover, to the best of our knowledge, there exists no self-stabilizing algorithm for exploration either in a static or a dynamic environment. Note that there exist such fault-tolerant solutions in static graphs to other problems (*e.g.,* naming and leader election [3]).

Our Contribution. The main contribution of this paper is to prove that the necessary and sufficient numbers of robots for perpetual exploration of highly dynamic rings exhibited in [5] also hold in a self-stabilizing setting *at the price of the loss of anonymity of robots*. Note that this price is unavoidable in the self-stabilizing setting since a classical symmetry breaking argument shows that it is impossible to solve the exploration problem with any number of

anonymous robots. Indeed, to be self-stabilizing, an exploration algorithm must tolerate any initial positions of robots. In particular, all robots may start on the same node. Then, if the robots are anonymous, deterministic, and execute the same algorithm, no algorithm can prevent them to act as only one robot which is unable to perpetually explore a connected-over-time ring of size 3 and more [5].

More precisely, our contribution is achieved through the following technical achievements. Section 3 presents two impossibility results establishing that at least two (resp. three) self-stabilizing robots are necessary to perpetually explore highly dynamic rings of size greater than 3 (resp. 4) *even if robots are not anonymous*. Note that these necessity results are not implied by the ones of [5] (that focuses on anonymous robots). Then, Sections 4 and 5 present and prove two algorithms showing the sufficiency of these conditions.

2. Model

In this section, we propose an extension of the classical model of robot networks in static graphs introduced by [22] to the context of dynamic graphs.

Dynamic graphs. In this paper, we consider the model of *evolving graphs* introduced in [30]. We hence consider the time as discretized and mapped to $\mathbb{N}$. An evolving graph $\mathcal{G}$ is an ordered sequence $\{G_0, G_1, G_2, \ldots\}$ of subgraphs of a given static graph $G = (V, E)$. The (static) graph G is called the *footprint* of $\mathcal{G}$. In the following, we restrict ourselves to evolving graphs whose footprints are anonymous, bidirectional, unoriented, and simple graphs. For any $i \geq 0$, we have $G_i = (V, E_i)$ and we say that the edges of E_i are *present* in $\mathcal{G}$ at time i. The *underlying graph* of $\mathcal{G}$, denoted $U_{\mathcal{G}}$, is the static graph gathering all edges that are present at least once in $\mathcal{G}$ (*i.e.,* $U_{\mathcal{G}} = (V, E_{\mathcal{G}})$ with $E_{\mathcal{G}} = \bigcup_{i=0}^{\infty} E_i$). An *eventual missing edge* is an edge of $E_{\mathcal{G}}$ such that there exists a time after which this edge is never present in $\mathcal{G}$. A *recurrent edge* is an edge of $E_{\mathcal{G}}$ that is not eventually missing. The *eventual underlying graph* of $\mathcal{G}$, denoted $U^{\omega}_{\mathcal{G}}$, is the static graph gathering all recurrent edges of $\mathcal{G}$ (*i.e.,* $U^{\omega}_{\mathcal{G}} = (V, E^{\omega}_{\mathcal{G}})$ where $E^{\omega}_{\mathcal{G}}$ is the set of recurrent edges of $\mathcal{G}$). In this paper, we chose to make minimal assumptions on the dynamicity of our graph since we restrict ourselves on *connected-over-time* evolving graphs. The only constraint we impose on evolving graphs of this class is that their eventual underlying graph is connected [14] (intuitively, that means that any node is infinitely often reachable from any other one). For the sake of the proof, we also consider the weaker class of *edge-recurrent* evolving graphs where the eventual underlying graph is connected and matches the footprint. In the following, we consider only connected-over-time evolving graphs whose footprint is a ring of arbitrary size called connected-over-time rings for simplicity. We call n the size of the ring. Although the ring is unoriented, to simplify the presentation and discussion, we, as external observers, distinguish between the clockwise and the counter-clockwise direction in the ring.

For the sake of some proofs in this paper, we need to introduce two operators on evolving graphs. The first one, denoted $\setminus$, removes some edges of an evolving graph for some time ranges. More formally, from an evolving graph $\mathcal{G} = \{(V, E_0), (V, E_1), (V, E_2), \ldots\}$, we define the evolving graph $\mathcal{G}\setminus\{(\tilde{E}_1, \tau_1), \ldots (\tilde{E}_k, \tau_k)\}$ (with for any $i \in \{1, \ldots, k\}$, $\tilde{E}_i \subseteq E$ and $\tau_i \subseteq \mathbb{N}$)

as the evolving graph $\{(V, E'_0), (V, E'_1), (V, E'_2), \ldots\}$ such that: $\forall t \in \mathbb{N}, \forall e \in E, e \in E'_t \Leftrightarrow e \in E_t \wedge (\forall i \in \{1, \ldots, k\}, e \notin \tilde{E}_i \vee t \notin \tau_i)$. The second operator, denoted $\otimes$, concatenates a prefix of an evolving graph with a suffix of another one. Formally, given two evolving graphs, $\mathcal{G}$ and $\mathcal{H}$, and an integer t, the evolving graph $\mathcal{G} \otimes_t \mathcal{H}$ is the evolving graph $\mathcal{G}'$ defined by: $e \in G'_i$ if and only if $i \leq t \wedge e \in G_i$ or $i > t \wedge e \in H_i$.

Finally, we say that two evolving graphs $\mathcal{G}$ and $\mathcal{G}'$ share the same prefix if there exists a time t such that for any $i \leq t$, $G_i = G'_i$.

Robots. We consider systems of autonomous mobile entities called robots moving in a discrete and dynamic environment modeled by an evolving graph $\mathcal{G} = \{(V, E_1), (V, E_2) \ldots\}$, V being a set of nodes representing the set of locations where robots may be, E_i being the set of bidirectional edges representing connections through which robots may move from a location to another one at time i. Robots execute the same algorithm. They are identified (each of them has a distinct identifier in a finite set ID). Robots have persistent memory that is divided in two distinct areas: a corruptible one containing variables and an incorruptible one containing their algorithm and its constants (as identifiers). Robots are unable to directly communicate with one another by any means. Robots are endowed with local strong multiplicity detection (*i.e.,* they are able to detect the exact number of robots located on their current node). They have no a priori knowledge about the ring they explore (size, diameter, dynamicity, ...) nor on the robots (number, bound on size of identifiers...). Finally, each robot has its own stable chirality (*i.e.,* each robot is able to locally label the two ports of its current node with *left* and *right* consistently over the ring and time but two different robots may not agree on this labeling). We assume that each robot has a variable *dir* that stores a direction (either *left* or *right*). At any time, we say that a robot points to *left* (resp. *right*) if its variable *dir* is equal to this (local) direction. We say that a robot considers the clockwise (resp., counter-clockwise) direction if the (local) direction pointed to by this robot corresponds to the (global) direction seen by an external observer.

Execution. The configuration of the system at time t (denoted γ_t) captures the position (*i.e.,* the node where the robot is currently located) and the state (*i.e.,* the value of the corruptible memory area of the robot) of each robot at a given time. We say that robots form a *tower* on a node v in γ_t if at least two robots are co-located on v in γ_t. Given an evolving graph $\mathcal{G} = \{G_0, G_1, G_2, \ldots\}$, an algorithm $\mathcal{A}$, and an initial configuration γ_0, the execution $\mathcal{E}$ of $\mathcal{A}$ on $\mathcal{G}$ starting from γ_0 is the infinite sequence $(G_0, \gamma_0), (G_1, \gamma_1), (G_2, \gamma_2), \ldots$ where, for any $i \geq 0$, the configuration γ_{i+1} is the result of the execution of a synchronous round by all robots from (G_i, γ_i) as explained below.

The round that transitions the system from (G_i, γ_i) to (G_{i+1}, γ_{i+1}) is composed of three atomic and synchronous phases: Look, Compute, Move. During the Look phase, each robot gathers information about its environment in G_i. More precisely, each robot updates the value of the following local predicates: (i) $NumberOfRobotsOnNode()$ that returns the exact number of robots present at the node of the robot; (ii) $ExistsEdgeOnLeft()$ that returns true if an edge in the left direction of the robot is present, false otherwise; (iii) $ExistsEdgeOnRight()$ that returns true if an edge in the right direction of the robot is

present, false otherwise; (*iv*) $ExistsAdjacentEdge()$ returns true if an edge adjacent to the current node of the robot is present, false otherwise. During the Compute phase, each robot executes the algorithm $\mathcal{A}$ that may modify some of its variables (in particular dir) depending on its current state and on the values of the predicates updated during the Look phase. Finally, the Move phase consists of moving each robot trough one edge in the direction it points to if there exists an edge in that direction, otherwise, *i.e.,* if the edge is missing at that time, the robot remains at its current node. Note that the i^{th} round is entirely executed on G_i and that the transition from G_i to G_{i+1} occurs only at the end of this round. We say that a robot is *edge-activated* during a round if there exists at least one edge adjacent to its location during that round. To simplify the pseudo-code of the algorithms, we assume that the robots have access to two predicates: $ExistsEdgeOnCurrentDirection()$ (that returns true if an edge is present at the direction currently pointed by the robot, false otherwise) and $ExistsEdgeOnOppositeDirection()$ (that returns true if an edge is present in the direction opposite to the one currently pointed by the robot, false otherwise). Both of these two predicates depend on the values of the predicates $ExistsRightEdge()$ and $ExistsLeftEdge()$, and on the value of the variable dir.

We introduce here a notation used through technical parts of the paper. We denote by $name(r, t)$ the value of a variable or a predicate $name$ of a given robot r after the Look phase of a given round t.

Self-Stabilization. Intuitively, a self-stabilizing algorithm is able to recover in a finite time a correct behavior from any arbitrary initial configuration (that captures the effect of an arbitrary transient fault in the system). More formally, an algorithm $\mathcal{A}$ is *self-stabilizing* for a problem on a class of evolving graphs $\mathcal{C}$ if and only if it ensures that, for any configuration γ_0, the execution of $\mathcal{A}$ on any $\mathcal{G} \in \mathcal{C}$ starting from γ_0 contains a configuration γ_i such that the execution of $\mathcal{A}$ on $\mathcal{G}$ starting from γ_i satisfies the specification of the problem. Note that, in the context of robot networks, this definition implies that robots must tolerate both arbitrary initialization of their variables and arbitrary initial positions (in particular, robots may be stacked in the initial configuration).

Perpetual Exploration. Given an evolving graph $\mathcal{G}$, a perpetual exploration algorithm guarantees that every node of $\mathcal{G}$ is infinitely often visited by at least one robot (*i.e.,* a robot is infinitely often located at every node of $\mathcal{G}$). Note that this specification does not require that every robot visits infinitely often every node of $\mathcal{G}$.

3. Necessary Number of Robots

This section is devoted to the proof of the necessity of two (resp. three) self-stabilizing identified robots to perform perpetual exploration of highly dynamic rings of size at least 3 (resp. 4). To reach this goal, we provide two impossibility results.

First, we prove (see Theorem 3.1) that two robots with distinct identifiers are not able to perpetually explore in a self-stabilizing way connected-over-time rings of size greater than 4. Then, we show that we can borrow arguments from [5] to prove Theorem 3.2 that states

that only one robot cannot complete the self-stabilizing perpetual exporation of connected-over-time rings of size greater than 3.

3.1. Highly Dynamic Rings of Size 4 or More

The proof of Theorem 3.1 makes use of a generic framework proposed in [6]. Note that, even if this generic framework is designed for another model (namely, the classical message passing model), it is straightforward to borrow it for our current model. Indeed, its proof only relies on the determinism of algorithms and indistinguishability of dynamic graphs, these arguments being directly translatable in our model. We present briefly this framework here. The interested reader is referred to [6] for more details.

This framework is based on a theorem that ensures that, if we take a sequence of evolving graphs with ever-growing common prefixes (that hence converges to the evolving graph that shares all these common prefixes), then the sequence of corresponding executions of any deterministic algorithm also converges. Moreover, we are able to describe the execution to which it converges as the execution of this algorithm on the evolving graph to which the sequence converges. This result is useful since it allows us to construct counter-examples in the context of impossibility results. Indeed, it is sufficient to construct an evolving graphs sequence (with ever-growing common prefixes) and to prove that their corresponding execution violates the specification of the problem for ever-growing time to exhibit an execution that never satisfies the specification of the problem.

In order to build the evolving graphs sequence suitable for the proof of our impossibility result, we need the following technical lemma.

Lemma 3.1. *Any self-stabilizing deterministic perpetual exploration algorithm in connected-over-time rings of size 4 or more using 2 robots r_1 and r_2 with distinct identifiers satisfies: for any states s_1 and s_2, for any distinct identifiers id_1 and id_2, it exists an integer t' ($t' \geq 0$) such that if r_1 (resp. r_2) with identifier id_1 (resp. id_2) is on node u_1 (resp. u_2) in state s_1 (resp. s_2) at a time t and there exists only one adjacent edge to each position of the robots continuously present from time t to time $t+t'$, then r_1 and/or r_2 moves at time $t+t'$. This lemma holds even if the robots have the same chirality.*

Proof. By contradiction, assume that there exists a self-stabilizing deterministic perpetual exploration algorithm $\mathcal{A}$ in connected-over-time rings of size 4 or more using 2 robots r_1 and r_2 satisfying: there exist two states s_1 and s_2, and two distinct identifiers id_1 and id_2 such that, for any integer t' ($t' \geq 0$), r_1 (resp. r_2) with identifier id_1 (resp. id_2) is on node u_1 (resp. u_2) in state s_1 (resp. s_2) at a time t and there exists only one adjacent edge to each position of the robots continuously present from time t to time $t+t'$ and none of the robots move at time $t+t'$.

Note that the existence of such states and identifiers is not contradictory with the definition of $\mathcal{A}$ if this algorithm has the ability to avoid that robots ever reaches such states in each of its execution. In the following, we are going to prove that it is not the case.

Let G be a ring of size 4 or more. Let $\mathcal{G} = \{G_0, G_1, \ldots\}$ be a connected-over-time ring such that $\forall i, G_i = G$. Consider the evolving graph $\mathcal{G}'$ such that $\mathcal{G}' = \mathcal{G} \backslash \{(\{e\}, \{0, \ldots, +\infty\})\}$,

where $e = \{u, v\}$ is an arbitrary edge of G. Note that $\mathcal{G}'$ is a connected-over-time ring, since it has only one eventual missing edge.

If $u_1 = u_2$, then we define the following configuration γ on $\mathcal{G}'$: r_1 (resp. r_2) with identifier id_1 (resp. id_2) is on node u (resp. u) in state s_1 (resp. s_2).

If $u_1 \neq u_2$, then we define the following configuration γ on $\mathcal{G}'$: r_1 (resp. r_2) with identifier id_1 (resp. id_2) is on node u (resp. v) in state s_1 (resp. s_2).

As $\mathcal{A}$ is self-stabilizing, there exists an execution ε of $\mathcal{A}$ on $\mathcal{G}'$ starting from γ. In both cases above, by construction, there is only one adjacent edge to each position of the robots continuously present from time 0 to $+\infty$, and r_1 and r_2 are respectively in state s_1 and s_2 at time 0. Then, by assumption, r_1 and r_2 do not leave their respective nodes in ε. As $\mathcal{G}'$ counts 4 nodes or more, we obtain a contradiction with the fact that $\mathcal{A}$ is a self-stabilizing algorithm solving deterministically the perpetual exploration problem for connected-over-time rings of size 4 or more using two robots possessing distinct identifiers. □

Theorem 3.1. *There exists no deterministic algorithm satisfying the perpetual exploration specification in a self-stabilizing way on the class of connected-over-time rings of size* 4 *or more with two fully synchronous robots possessing distinct identifiers.*

Proof. By contradiction, assume that there exists a deterministic algorithm $\mathcal{A}$ satisfying the perpetual exploration specification in a self-stabilizing way on any connected-over-time rings of size 4 or more using two robots r_1 and r_2 possessing distinct identifiers.

Consider the connected-over-time graph $\mathcal{G} = \{G_0, G_1, \ldots\}$ whose footprint G is a ring of size 4 or more and such that $\forall i \in \mathbb{N}, G_i = G$.

Consider four nodes u, v, w and x of $\mathcal{G}$, such that node v is the adjacent node of u in the clockwise direction, w is the adjacent node of v in the clockwise direction, and x is the adjacent node of w in the clockwise direction. We denote respectively e_{vr} and e_{vl} the clockwise and counter clockwise adjacent edges of v, e_{wr} and e_{wl} the clockwise and counter clockwise adjacent edges of w, and e_{xr} and e_{xl} the clockwise and counter clockwise adjacent edges of x. Note that $e_{vr} = e_{wl}$, and $e_{wr} = e_{xl}$.

Let $\mathcal{G}'$ be $\mathcal{G}\backslash\{(\{e_{vl}\}, \mathbb{N})\}$. Let ε be the execution of $\mathcal{A}$ on $\mathcal{G}'$ starting from the configuration where r_1 (resp. r_2) is located on node v (resp. w).

Our goal is to construct a sequence of connected-over-time rings denoted $(\mathcal{G}_m)_{m\in\mathbb{N}}$ such that $\mathcal{G}_0 = \mathcal{G}'$ and, for any $i \geq 0$, only nodes among $\{v, w, x\}$ have been visited until time t_i in ε_i (the execution of $\mathcal{A}$ in $\mathcal{G}_i$ starting from the same configuration as ε), $(t_m)_{m\in\mathbb{N}}$ being a strictly increasing sequence with $t_0 = 0$. First, we show in the next paragraph that, if some such $\mathcal{G}_i$ exists and moreover ensures the existence of an integer $t'_i \geq 0$ such that until time $t_i + t'_i + 1$ only nodes among $\{v, w, x\}$ have been visited in ε_i, then we can construct $\mathcal{G}_{i+1}$ as shown on Figure 1. We prove, after that, that our construction guarantees the existence of such a t'_i, implying the well-definition of $(\mathcal{G}_m)_{m\in\mathbb{N}}$.

Since $\mathcal{G}_i$ is a connected-over-time ring, and since $\mathcal{A}$ is a deterministic algorithm solving the perpetual exploration problem in a self-stabilizing way on connected-over-time rings of size 4 or more using 2 robots possessing distinct identifiers, when the configuration γ_i at time t_i is such that there is exactly one adjacent edge present to the location of each of

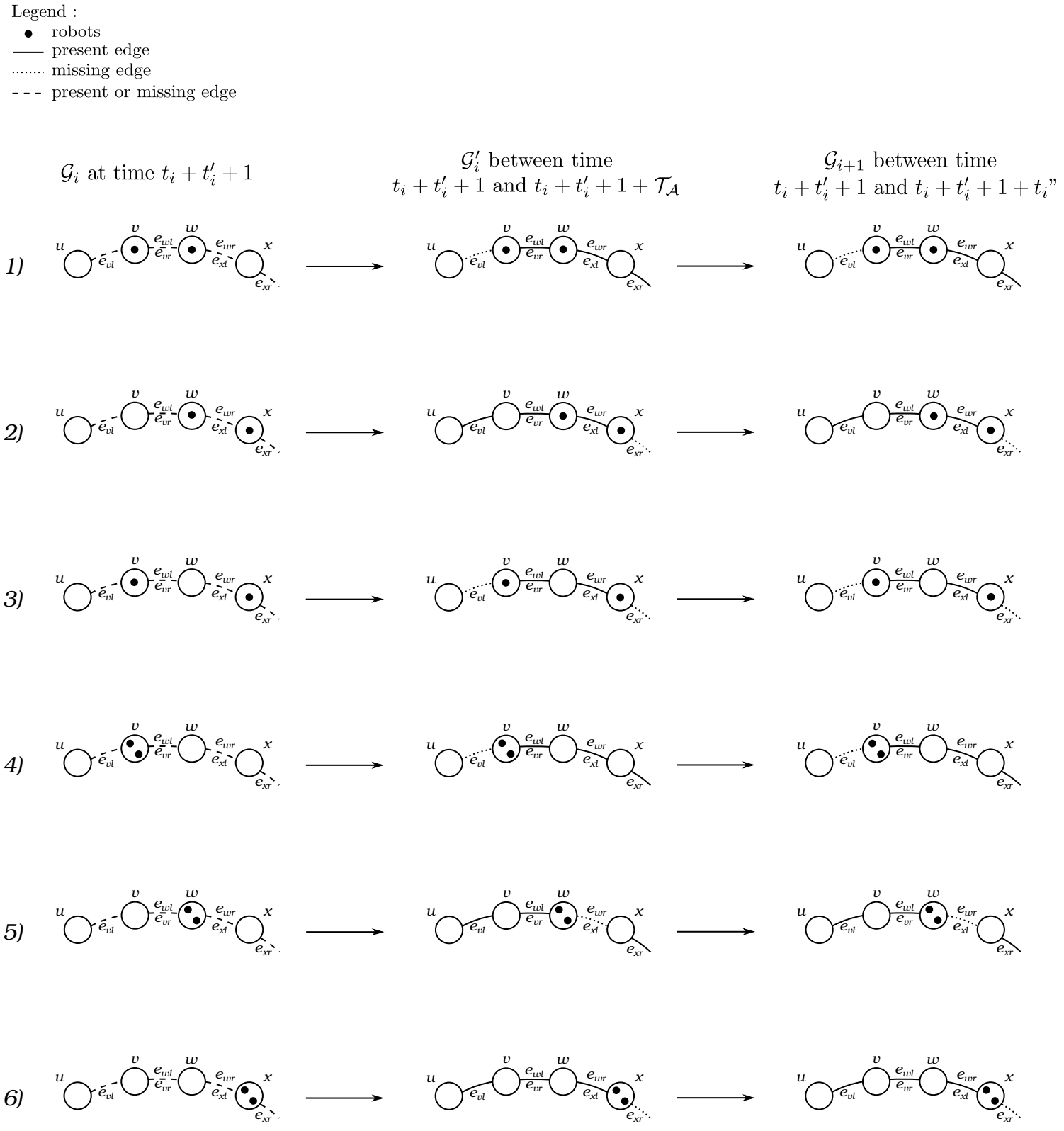

Figure 1: Construction of $\mathcal{G}_{i+1}$ from $\mathcal{G}_i$.

the two robots, we use Lemma 3.1 to exhibit the smallest integer $t'_i \geq 0$ such that if the configuration γ_i last from time t_i to time $t_i + t'_i$, then one or both of the robots move at time $t_i + t'_i$. $(*)$ Similarly, when the configuration γ_i at time t_i is such that there is only one missing edge, and that only one robot is adjacent to this missing edge, then we can also exhibit the smallest integer $t'_i \geq 0$ such that at time $t_i + t'_i$ at least one of the robots move. Indeed, if this configuration lasts from time t_i to time $+\infty$, $\mathcal{G}_i$ is a connected-over-time ring, and if none of the robots move in this configuration, the perpetual exploration cannot be solved. Therefore such an integer t'_i exists. Let $\mathcal{T}_\mathcal{A}$ be the largest integer t'_i whatever the states and identifiers considered by the robots executing $\mathcal{A}$. Since the states reached by the robots executing $\mathcal{A}$ and the identifiers are among a finite set of values, $\mathcal{T}_\mathcal{A}$ exists and is bounded.

In the following we show how we construct the dynamic graph $\mathcal{G}_{i+1}$ in function of $\mathcal{G}_i$, t_i and t'_i. As we assume that in $\mathcal{G}_i$, until time $t_i + t'_i + 1$, only nodes among $\{v, w, x\}$ have been visited, then the following cases are possible. If the two robots are on two distinct nodes in $\mathcal{G}_i$ at time $t_i + t'_i + 1$ then:

1. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, one of the robots is on node v and the other robot is on node w then let $E_i = \{e_{vl}\}$.
2. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, one of the robots is on node w and the other robot is on node x then let $E_i = \{e_{xr}\}$.
3. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, one of the robots is on node v and the other robot is on node x then let $E_i = \{e_{vl}, e_{xr}\}$.

If the two robots are on the same node in $\mathcal{G}_i$ at time $t_i + t'_i + 1$ then:

4. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, the two robots are on node v, then let $E_i = \{e_{vl}\}$.
5. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, the two robots are on node w then let $E_i = \{e_{wr}\}$.
6. if, at time $t_i + t'_i + 1$ in $\mathcal{G}_i$, the two robots are on node x, then let $E_i = \{e_{xr}\}$.

For each case, we define $\mathcal{G}'_i$ such that $\mathcal{G}'_i$ and $\mathcal{G}_i$ have the same footprint and $\mathcal{G}'_i = \mathcal{G}_i \otimes_{t_i+t'_i} (\mathcal{G} \backslash \{(E_i, \{t_i+t'_i+1, \ldots, t_i+t'_i+1+\mathcal{T}_\mathcal{A}\})\})$. Note that $\mathcal{G}'_i$ is a connected-over-time ring since it is indistinguishable from $\mathcal{G}$ after $t_i+t'_i+1+\mathcal{T}_\mathcal{A}$. We can then apply either Lemma 3.1 or $(*)$ to find the smallest integer t_i" (with $\mathcal{T}_\mathcal{A} \geq t_i$" ≥ 0) such that at time $t_i+t'_i+1+t_i$" at least one of the robots move in $\mathcal{G}'_i$. By definition of $\mathcal{T}_\mathcal{A}$, t_i" exists. Finally, we define $\mathcal{G}_{i+1}$ such that $\mathcal{G}_{i+1}$ and $\mathcal{G}_i$ have the same footprint and $\mathcal{G}_{i+1} = \mathcal{G}_i \otimes_{t_i+t'_i} (\mathcal{G} \backslash \{(E_i, \{t_i+t'_i+1, \ldots, t_i+t'_i+1+t_i\text{"}\})\})$.

$\mathcal{G}_{i+1}$ is a connected-over-time ring (since it is indistinguishable from $\mathcal{G}$ after $t_i + t'_i + 1 + t_i$"). Note that $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ are indistinguishable for robots until time $t_i + t'_i$. This implies that, at time $t_i + t'_i$, r_1 and r_2 are on the same node in ε_i and in ε_{i+1}. By construction of t'_i, either r_1 or r_2 or both of the two robots move at time $t_i + t'_i$ in ε_{i+1}. Moreover, even if one or both of the robots move during the Move phase of time $t_i + t'_i$, at time $t_i + t'_i + 1$ the robots are still on nodes among $\{v, w, x\}$, by assumption on $\mathcal{G}_i$ and construction of $\mathcal{G}_{i+1}$, since from time t_i to time $t_i + t'_i$ the edges permitting a robot to go on a node other than the nodes among $\{v, w, x\}$ are missing.

Let $t_{i+1} = t_i + t'_i + 1$ and t_i" $= t'_{i+1}$. Then we can construct recursively each dynamic ring of $(\mathcal{G}_m)_{m\in\mathbb{N}}$ by applying the argues above on all the possible configurations reached by the movements of the robots at time $t_{i+1} + t'_{i+1} + 1$ on $\mathcal{G}_{i+1}$.

Note that the recurrence can be initiated, since $\mathcal{G}_0$ exists, is a connected-over-time ring and from time $t_0 = 0$ to time $t'_0 + 1$ only nodes among $\{v, w, x\}$ have been visited. In other words, $(\mathcal{G}_m)_{m\in\mathbb{N}}$ is well-defined.

We can then define the evolving graph $\mathcal{G}_\omega$ such that $\mathcal{G}_\omega$ and $\mathcal{G}_0$ have the same footprint and for all $i \in \mathbb{N}$, $\mathcal{G}_\omega$ shares a common prefix with $\mathcal{G}_i$ until time $t_i + t'_i$. As the sequence $(t_m + t'_m)_{m\in\mathbb{N}}$ is increasing by construction, this implies that the sequence $(\mathcal{G}_m)_{m\in\mathbb{N}}$ converges to $\mathcal{G}_\omega$.

Note that $\mathcal{G}_\omega$ is a connected-over-time ring. Indeed, among the cases presented, only the dynamic graph $\mathcal{G}_{i+1}$ built in case 3 possesses two simultaneous missing edges between time t_{i+1} and $t_{i+1} + t'_{i+1}$. However, even if at some point such a dynamic ring is used to construct $(\mathcal{G}_m)_{m\in\mathbb{N}}$, note that since at least one of the two robots move at time $t_{i+1} + t'_{i+1}$ then $\mathcal{G}_{i+2}$ cannot be built thanks to case 3. Therefore, there is only one missing edge in $\mathcal{G}_{i+2}$ between time t_{i+2} and $t_{i+2} + t'_{i+2}$. In conclusion, there is at most one eventual missing edge and $\mathcal{G}_\omega$ is a connected-over-time ring.

Applying the theorem of [6], we obtain that, until time $t_i + t'_i$, the execution of $\mathcal{A}$ on $\mathcal{G}_\omega$ is identical to the one on $\mathcal{G}_i$. This implies that, executing $\mathcal{A}$ on $\mathcal{G}_\omega$ (whose footprint is a ring of size 4 or more), r_1 and r_2 only visit the nodes among $\{v, w, x\}$. This is contradictory with the fact that $\mathcal{A}$ satisfies the perpetual exploration specification on connected-over-time rings of size 4 or more using 2 robots possessing distinct identifiers. □

3.2. Highly Dynamic Rings of Size 3 *or More*

In [5], the authors prove (in Theorem V.1) that a single anonymous and synchronous robot cannot perpetually explore connected-over-time rings of size 3 or more in a fault-free setting. We can do two observations. First, any fault-free synchronous execution is possible in a self-stabilizing setting. Second, in the case of a *single* robot, the anonymous and the identified model are equivalent.

These observations are sufficient to directly state the following result:

Theorem 3.2. *There exists no deterministic algorithm satisfying the perpetual exploration specification in a self-stabilizing way on the class of connected-over-time rings of size* 3 *or more using one robot possessing an identifier.*

4. Sufficiency of Three Robots for $n \geq 4$

In this section, we present our self-stabilizing deterministic algorithm for the perpetual exploration of any connected-over-time ring of size greater than 4 with three robots. In this context, the difficulty to complete the exploration is twofold. First, in connected-over-time graphs, robots must deal with the possible existence of some eventual missing edge (without the guarantee that such edge always exists). Note that, in the case of a ring, there is at most one eventual missing edge in any execution (otherwise, we have a contradiction with the connected-over-time property). Second, robots have to handle the arbitrary initialization of the system (corruption of variables and arbitrary position of robots).

4.1. Presentation of the algorithm

Principle of the algorithm. The main idea behind our algorithm is that a robot does not change its direction (arbitrarily initialized) while it is isolated. This allows robots to perpetually explore connected-over-time rings with no eventual missing edge regardless of the initial direction of the robots.

Obviously, this idea is no longer sufficient when there exists an eventual missing edge since, in this case, at least two robots will eventually be stuck (*i.e.,* they point to an eventual missing edge that they are never able to cross) forever at one end of the eventual missing edge. When two (or more) robots are located at the same node, we say that they form a tower. In this case, our algorithm ensures that at least one robot leaves the tower in a finite time. In this way, we obtain that, in a finite time, a robot is stuck at each end of the eventual missing edge. These two robots located at two ends of the eventual missing edge play the role of "sentinels" while the third one (we call it a "visitor") visits other nodes of the ring in the following way. The "visitor" keeps its direction until it meets one of these "sentinels", they then switch their roles: After the meeting, the "visitor" still maintains the same direction (becoming thus a "sentinel") while the "sentinel" robot changes its direction (becoming thus a "visitor" until reaching the other "sentinel").

In fact, robots are never aware if they are actually stuck at an eventual missing edge or are just temporarily stuck on an edge that will reappear in a finite time. That is why it is important that the robots keep their directions and try to move forward while there is no meeting in order to track a possible eventual missing edge. Our algorithm only guarantees convergence in a finite time towards a configuration where a robot plays the role of "sentinel" at each end of the eventual missing edge if such an edge exists. Note that, in the case where there is no eventual missing edge, this mechanism does not prevent the correct exploration of the ring since it is impossible for a robot to be stuck forever.

Our algorithm easily deals with the initial corruption of its variables. Indeed, all variables of a robot (with the exception of a counter and the variable *dir* whose initial respective values have no particular impact) store information about the environment of this robot in the previous round it was edge-activated. These variables are updated each time a robot is edge-activated. Since we consider connected-over-time rings, there can only exist one eventual missing edge, therefore all robots are infinitely often edge-activated. The initial values of these variables are hence reset in a finite time. The main difficulty to achieve self-stabilization is to deal with the arbitrary initial position of robots. In particular, the robots may initially form towers. In the worst case, all robots of a tower may be stuck at an eventual missing edge and be in the same state. They are then unable to start the "sentinels"/"visitor" scheme explained above. Our algorithm needs to "break" such a tower in a finite time (*i.e.,* one robot must leave the node where the tower is located). In other words, we tackle a classical problem of symmetry breaking. We achieve this by providing each robot with a function that returns, in a finite number of invocations, different global directions to two robots of the tower based on the private identifier of the robot and without any communication among the robots. More precisely, this is done thanks to a transformation of the robot identifier: each bit of the binary representation of the identifier is duplicated and we add the bits "010"

at the end of the sequence of these duplicated bits. Then, at each invocation of the function, a robot reads the next bit of this transformed identifier. If the robot reads zero, it tries to move to its left. Otherwise, it tries to move to its right. Doing so, in a finite number of invocation of this function, at least one robot leaves the tower. If necessary, we repeat this "tower breaking" scheme until we are able to start the "sentinels"/"visitor" scheme.

The main difficulty in designing this algorithm is to ensure that these two mechanisms ("sentinels"/"visitor" and "tower breaking") do not interfere with each other and prevent the correct exploration. We solve this problem by adding some waiting "at good time", especially before starting the procedure of tower breaking by identifier to ensure that robots do not prematurely turn back and "forget" to explore some parts of the ring.

Formal presentation of the algorithm. Before presenting our algorithm formally, we need to introduce the set of constants (*i.e.,* variables assumed to be not corruptible) and the set of variables of each robot. We also introduce three auxiliary functions.

As stated in the model, each robot has an unique identifier. We denote it by id and represent it in binary as $b_1 b_2 \dots b_{|id|}$. We define, for the purpose of the "breaking tower" scheme, the constant $TransformedIdentifier$ by its binary representation $b_1 b_1 b_2 b_2 \dots b_{|id|} b_{|id|} 010$ (each bit of id is duplicated and we add the three bits 010 at the end). We store the length of the binary representation of $TransformedIdentifier$ in the constant ℓ and we denote its ith bit by $TransformedIdentifier[i]$ for any $1 \leq i \leq \ell$.

In addition to the variable dir defined in the model, each robot has the following three variables: (i) the variable $i \in \mathbb{N}$ corresponds to an index to store the position of the last bit read from $TransformedIdentifier$; (ii) the variable $NumberRobotsPreviousEdgeActivation \in \mathbb{N}$ stores the number of robots that were present at the node of the robot during the Look phase of the last round where it was edge-activated; and (iii) the variable $HasMovedPreviousEdgeActivation \in \{true, false\}$ indicates if the robot has crossed an edge during its last edge-activation.

Our algorithm makes use of a function Update that updates the values of the two last variables according to the current environment of the robot each time it is edge-activated. We provide the pseudo-code of this function in Algorithm 1. Note that this function also allows us to deal with the initial corruption of the two last variables since it resets them in the first round where the robot is edge-activated.

We already stated that, whenever robots are stuck forming a tower, they make use of a function to "break" the tower in a finite time. The pseudo-code of this function GiveDirection appears in Algorithm 2. It assigns the value $left$ or $right$ to the variable dir of the robot depending on the ith bit of the value of $TransformedIdentifier$. The variable i is incremented modulo ℓ (that implicitly resets this variable when it is corrupted) to ensure that successive calls to GiveDirection will consider each bit of $TransformedIdentifier$ in a round-robin way. As shown in the next section, this function guarantees that, if two robots are stuck together in a tower and invoke repeatedly their own function GiveDirection, then two distinct global directions are given in finite time to the two robots regardless of their chirality. This property allows the algorithm to "break" the tower since at least one robot is then able to leave the node where the tower is located.

Algorithm 1 Function Update

```
1: function UPDATE
2:     if ExistsAdjacentEdge() then
3:         NumberRobotsPreviousEdgeActivation ← NumberOfRobotsOnNode()
4:         HasMovedPreviousEdgeActivation ← ExistsEdgeOnCurrentDirection()
5:     end if
6: end function
```

Finally, we define the function OPPOSITEDIRECTION that simply affects the value $left$ (resp. $right$) to the variable dir when $dir = right$ (resp. $dir = left$).

There are two types of configurations in which the robots may change the direction they consider. So, our algorithm needs to identify them. We do so by defining a predicate that characterizes each of these configurations.

The first one, called $WeAreStuckInTheSameDirection()$, is dedicated to the detection of configurations in which the robot must invoke the "tower breaking" mechanism. Namely, the robot is stuck since at least one edge-activation with at least another robot and the edge in the direction opposite to the one considered by the robot is present. More formally, this predicate is defined as follows:

$$\begin{aligned} &WeAreStuckInTheSameDirection() \equiv \\ &\quad (NumberOfRobotsOnNode() > 1) \\ &\wedge\ (NumberOfRobotsOnNode() = NumberRobotsPreviousEdgeActivation) \\ &\wedge\ \neg ExistsEdgeOnCurrentDirection() \\ &\wedge\ ExistsEdgeOnOppositeDirection() \\ &\wedge\ \neg HasMovedPreviousEdgeActivation \end{aligned}$$

The second predicate, called $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$, is designed to detect configurations in which the robot must transition from the "sentinel" to the "visitor" role in the "sentinel"/"visitor" scheme. More precisely, such configuration is characterized by the fact that the robot is edge-activated, stuck during its previous edge-activation, and there are strictly more robots located at its node than at its previous edge-activation. More formally, this predicate is defined as follows:

$$\begin{aligned} &IWasStuckOnMyNodeAndNowWeAreMoreRobots() \equiv \\ &\quad (NumberOfRobotsOnNode() > NumberRobotsPreviousEdgeActivation) \\ &\wedge\ \neg HasMovedPreviousEdgeActivation \\ &\wedge\ ExistsAdjacentEdge() \end{aligned}$$

Now, we are ready to present the pseudo-code of the core of our algorithm (called $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, see Algorithm 3). The basic idea of the algorithm is the following. The function GIVEDIRECTION is invoked when $WeAreStuckInTheSameDirection()$ is true (to try to "break" the tower after the appropriate waiting), while the function OPPOSITEDIRECTION is called when $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ is true (to implement the "sentinel"/"visitor" scheme). Afterwards, the function UPDATE is called (to update the state of the robot according to its environment).

Algorithm 2 Function GiveDirection

```
1: function GiveDirection
2:     i ← i + 1 (mod ℓ) + 1
3:     if TransformedIdentifier[i] = 0 then
4:         dir ← left
5:     else
6:         dir ← right
7:     end if
8: end function
```

Algorithm 3 𝕊𝔼𝕃𝔽-𝕊𝕋𝔸𝔹_ℙ𝔼𝔽_3

```
1: if WeAreStuckInTheSameDirection() then
2:     GiveDirection
3: end if
4: if IWasStuckOnMyNodeAndNowWeAreMoreRobots() then
5:     OppositeDirection
6: end if
7: Update
```

4.2. Preliminaries to the Correctness Proof

First, we introduce some definitions and preliminary results that are extensively used in the proof.

We saw previously that the notion of tower is central in our algorithm. Intuitively, a tower captures the simultaneous presence of all robots of a given set on a node at each time of a given interval. We require either the set of robots or the time interval of each tower to be maximal. Note that the tower is not required to be on the same node at each time of the interval (robots of the tower may move together without leaving the tower).

We distinguish two kinds of towers according to the agreement of their robots on the global direction to consider at each time there exists an adjacent edge to their current location (excluding the last one). If they agreed, the robots form a long-lived tower while they form a short-lived tower in the contrary case. This implies that a short-lived tower is broken as soon as the robots forming the tower are edge-activated, while the robots of a long-lived tower move together at each edge-activation of the tower (excluding the last one).

Definition 4.1 (Tower)**.** *A tower T is a couple (S, θ), where S is a set of robots ($|S| > 1$) and $\theta = [t_s, t_e]$ is an interval of $\mathbb{N}$, such that all the robots of S are located at the same node at each instant of time t in θ and S or θ are maximal for this property. Moreover, if the robots of S move during a round $t \in [t_s, t_e[$, they are required to traverse the same edge.*

Definition 4.2 (Long-lived tower)**.** *A long-lived tower $T = (S, [t_s, t_e])$ is a tower such that there is at least one edge-activation of all robots of S in the time interval $[t_s, t_e[$.*

Definition 4.3 (Short-lived tower)**.** *A short-lived tower T is a tower that is not a long-lived tower.*

For $k > 1$, a long-lived (resp., a short-lived) tower $T = (S, \theta)$ with $|S| = k$ is called a k-long-lived (resp., a k-short-lived) tower.

As there are only three robots on our system, and that in each round each of them consider a global direction, we can make the following observation.

Observation 4.1. *There are at least two robots having the same global direction at each instant time.*

In the remainder of this section, we consider an execution $\mathcal{E}$ of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ executed by three robots r_1, r_2, and r_3 on a connected-over-time ring $\mathcal{G}$ of size $n \in \mathbb{N}$, with $n \geq 4$, starting from an arbitrary configuration.

We say that a robot r has a coherent state at time t, if the value of its variable $NumberRobotsPreviousEdgeActivation(r,t)$ corresponds to the value of its predicate $NumberOfRobotsOnNode()$ at its previous edge-activation and the value of its variable $HasMovedPreviousEdgeActivation(r,t)$ corresponds to the value of its predicate $ExistsEdgeOnCurrentDirection()$ at its previous edge-activation. The following lemma states that, for each robot, there exists a suffix of the execution in which the state of the robot is coherent.

Lemma 4.1. *For any robot, there exists a time from which its state is always coherent.*

Proof. Consider a robot r performing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$.

$\mathcal{G}$ belongs to the class of connected-over-time rings, hence at least one adjacent edge to each node of $\mathcal{G}$ is infinitely often present. This implies that r is infinitely often edge-activated, whatever its location is. Let t be the first time at which r is edge-activated.

Variables can be updated only during Compute phases of rounds. When executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, the variables $NumberRobotsPreviousEdgeActivation$ and $HasMovedPreviousEdgeActivation$ of r are updated with the current values of its predicates $NumberOfRobotsOnNode()$ and $ExistsEdgeOnCurrentDirection()$ only when it is edge-activated.

Therefore from time $t+1$, r is in a coherent state. □

Let t_1, t_2, and t_3 be respectively the time at which the robots r_1, r_2, and r_3, respectively are in a coherent state. Let $t_{max} = max\{t_1, t_2, t_3\}$. From Lemma 4.1, the three robots are in a coherent state from t_{max}. In the remainder of the proof, we focus on the suffix of the execution after t_{max}.

The two following lemmas (in combination with Lemma 4.5 and Corollary 4.1) aim at showing that, regardless of the chirality of the robots and the initial values of their variables i, a finite number of synchronous invocations of the function GiveDirection by two robots of a long-lived tower returns them a distinct global direction. This property is shown by looking closely to the structure of the binary representation of the transformed identifiers of the robots.

Indeed, the round-robin reading of its transformed identifier by a robot r can be seen as an infinite binary word $U(r)$ (the infinite concatenation of this transformed identifier). When robots of a long-lived tower are stuck on a node, we want them to break the tower by considering simultaneously two opposite global directions in finite time. The robots of a long-lived tower that are stuck on a node have their predicate $WeAreStuckInTheSameDirection()$ true and hence use the function GiveDirection. Two robots that possess the

same chirality and that call the function GiveDirection must consider two distinct bits at the same time to consider two opposite global directions at this time. Lemma 4.2 shows that any common factor of $U(r_1)$ and $U(r_2)$ is finite for any pair of robots r_1 and r_2, allowing us to latter (see Corollary 4.1) state that, in finite time, if the predicate $WeAreStuckInThe\text{-}SameDirection()$ of the robots are infinitely often true then the long-lived tower they are involved in is broken. Lemma 4.3 shows a similar result when robots of the long-lived tower does not have the same chirality.

To state formally these lemmas, we need to introduce some vocabulary and definitions from combinatorics on words. We consider words as (possibly infinite) sequence of letters from the alphabet $A = \{0, 1\}$. Given a word u, we refer to its i-th letter by $u[i]$. The length of a word u (denoted $|u|$) is its number of letters. Given two finite words $u = u[1] \dots u[k]$ and $v = v[1] \dots v[\ell]$ (with $k = |u|$ and $\ell = |v|$), the concatenation of u and v (denoted $u.v$) is the word $u[1] \dots u[k]v[1] \dots v[\ell]$ (with $|u.v| = k + \ell$). Given a finite word u, the word u^1 is u itself and the word u^z ($z > 1$) is the word $u.u^{z-1}$. Given a finite word u, the word u^ω is the infinite word $u.u.u. \dots$. A prefix u_1 of a word u is a word such that there exists a word u_2 satisfying $u = u_1.u_2$. A suffix u_2 of a word u is a word such that there exists a word u_1 satisfying $u = u_1.u_2$. A factor u_2 of a word u is a word such that there exists a prefix u_1 and a suffix u_3 of u satisfying $u = u_1.u_2.u_3$. The factor of u starting from the i^{th} bit of u and ending to the j^{th} bit of u included is denoted $u[i \dots j]$. A circular permutation of a word u is a word of the form $u_2.u_1$ where $u = u_1.u_2$.

Lemma 4.2. *Let u and v be two distinct transformed identifiers. If u^ω and v^ω share a common factor X, then X is finite.*

Proof. Consider two transformed identifiers u and v such that $u \neq v$.

By definition, the transformed identifier u is either equal to 00.010 or to $11.\Pi_{d=1}^{\alpha(u)}(\Pi_1^{\beta(u,d)}00.\Pi_1^{\gamma(u,d)}11).010$ $(*)$ with $\alpha(u)$ a function giving the number of blocks $(\Pi_1^{\beta(u,d)}00.\Pi_1^{\gamma(u,d)}11)$ contained in u, $\beta(u, d)$ a function giving the number of pair of bits 00 contained in the d^{th} block of u, and $\gamma(u, d)$ a function giving the number of pair of bits 11 contained in the d^{th} block of u.

Similarly, by definition v is either equal to 00.010 or to $11.\Pi_{d=1}^{\alpha(v)}(\Pi_1^{\beta(v,d)}00.\Pi_1^{\gamma(v,d)}11).010$ $(**)$.

Let $U = u^\omega$ and $V = v^\omega$.

Assume by contradiction that U and V share a common factor X of infinite size. Hence $U = x.X$ and $V = y.X$, with x (respectively y) the prefix of U (respectively of V). We have $X = \tilde{u}^\omega$, where $\tilde{u}$ is a circular permutation of the word u, and $X = \tilde{v}^\omega$, where $\tilde{v}$ is a circular permutation of the word v.

By definition of a common factor we have $\forall h \in \mathbb{N}^*, U[|x| + h] = V[|y| + h]$ $(***)$.

Let $k \in \mathbb{N}^*$ such that $U[|x| + k] = 0$, $U[|x| + k + 1] = 1$ and $U[|x| + k + 2] = 0$. By $(*)$ and since $U = x.X = x.\tilde{u}^\omega$, k exists. By $(*)$ and by construction of U, we know that $U[|x| + k + 3 \dots |x| + k + |u| + 2]$ is equal to u and $U[|x| + k + 3 \dots |x| + k + |u| - 1]$ is either equal to 00 or to $11.\Pi_{d=1}^{\alpha(u)}(\Pi_1^{\beta(u,d)}00.\Pi_1^{\gamma(u,d)}11)$.

By $(***)$, we have $V[|y|+k]=0$, $V[|y|+k+1]=1$ and $V[|y|+k+2]=0$. By $(**)$ and by construction of V, we know that $V[|y|+k+3\ldots|y|+k+|v|+2]$ is equal to v and $V[|y|+k+3\ldots|y|+k+|v|-1]$ is either equal to 00 or to $11.\Pi_{d=1}^{\alpha(v)}(\Pi_1^{\beta(v,d)}00.\Pi_1^{\gamma(v,d)}11)$.

Case 1: $|\mathbf{u}|=|\mathbf{v}|$.

If $|u|=|v|$, then by $(***)$ we have $U[|x|+k+3\ldots|x|+k+|u|+2]=V[|y|+k+3\ldots|y|+k+|v|+2]$. This implies that $u=v$, which leads to a contradiction with the fact that u and v are distinct.

Case 2: $|\mathbf{u}|\neq|\mathbf{v}|$.

Without lost of generality assume that $|u|<|v|$. We have $U[|x|+k+|u|]=0$, $U[|x|+k+|u|+1]=1$ and $U[|x|+k+|u|+2]=0$. Therefore by $(***)$ we have $V[|y|+k+|u|]=0$, $V[|y|+k+|u|+1]=1$ and $V[|y|+k+|u|+2]=0$.

Note that $|u|=2w+3$ with $w\in\mathbb{N}^*$. Similarly $|v|=2z+3$, with $z\in\mathbb{N}^*$, and $z>w$ since $|u|<|v|$. Since $V[|y|+k+3\ldots|y|+k+|v|+2]$ is equal to v, this implies that $V[|y|+k+3]=v[1]$, and $V[|y|+k+|u|]=V[|y|+k+2w+3]=v[i]$ where i is odd and such that $1\leq i\leq 2z$. Hence by $(**)$, necessarily $V[|y|+k+|u|]=V[|y|+k+|u|+1]$, which leads to a contradiction with the fact that $V[|y|+k+|u|]=0$ and $V[|y|+k+|u|+1]=1$.

□

Let us introduce the notation $\overline{w}$ which given a word w is defined such that $\overline{w}=\prod_{i\in\{1,\ldots,|w|\}}\overline{w[i]}$ where if $w[i]=1$ then $\overline{w[i]}=0$, and if $w[i]=0$ then $\overline{w[i]}=1$.

Lemma 4.3. *Let u and v be two distinct transformed identifiers. If u^ω and $\overline{v}^\omega$ share a common factor X, then X is finite.*

Proof. Consider two transformed identifiers u and v such that $u\neq v$.

By definition, the transformed identifier u is either equal to 00.010 or to $11.\Pi_{d=1}^{\alpha(u)}(\Pi_1^{\beta(u,d)}00.\Pi_1^{\gamma(u,d)}11).010$ $(*)$ with $\alpha(u)$ a function giving the number of blocks $(\Pi_1^{\beta(u,d)}00.\Pi_1^{\gamma(u,d)}11)$ contained in u, $\beta(u,d)$ a function giving the number of pair of bits 00 contained in the d^{th} block of u, and $\gamma(u,d)$ a function giving the number of pair of bits 11 contained in the d^{th} block of u.

Similarly, by definition v is either equal to 00.010 or to $11.\Pi_{d=1}^{\alpha(v)}(\Pi_1^{\beta(v,d)}00.\Pi_1^{\gamma(v,d)}11).010$. Call $w=\overline{v}$. This implies that $|w|=|v|$ and w is either equal to 11.101 or to $00.\Pi_{d=1}^{\alpha(v)}(\Pi_1^{\beta(v,d)}11.\Pi_1^{\gamma(v,d)}00).101$ $(**)$. Note that u and w are distinct. Indeed, if $|u|\neq|v|$ then, w and u are distinct since $|w|=|v|$. If $|u|=|v|$ then, since the suffix of size 3 of u is the word 010, and the suffix of size 3 of w is the word 101, then u and w are distinct.

Let $U=u^\omega$ and $W=w^\omega$.

Assume by contradiction that U and W share a common factor X of infinite size. Hence $U=x.X$ and $W=y.X$, with x (respectively y) the prefix of U (respectively of W). We

have $X = \tilde{u}^\omega$, where $\tilde{u}$ is a circular permutation of the word u, and $X = \tilde{w}^\omega$, where $\tilde{w}$ is a circular permutation of the word w.

By definition of a common factor we have $\forall h \in \mathbb{N}^*, U[|x|+h] = W[|y|+h]$ $(***)$.

Let $k \in \mathbb{N}^*$ such that $U[|x|+k] = 0$, $U[|x|+k+1] = 1$ and $U[|x|+k+2] = 0$. By $(*)$ and since $U = x.X = x.\tilde{u}^\omega$, k exists. By $(*)$ and by construction of U, we know that $U[|x|+k+3 \dots |x|+k+|u|+2]$ is equal to u.

By $(***)$, we have $W[|y|+k] = 0$, $W[|y|+k+1] = 1$ and $W[|y|+k+2] = 0$. By $(**)$ and by construction of W, we know that either $W[|y|+k+4 \dots |y|+k+|w|+3] = w$ (in the case where $W[|y|+k+1] = w[|w|-2]$) or $W[|y|+k+2 \dots |y|+k+|w|+1] = w$ (in the case where $W[|y|+k+1] = w[|w|]$).

Case 1: $\mathbf{W[|y|+k+4 \dots |y|+k+|w|+3] = w}$.

In this case $W[|y|+k+3] = 1$, then necessarily by $(***)$ $U[|x|+k+3] = 1$. By $(*)$, and since $U[|x|+k+3 \dots |x|+k+|u|+2] = u$, this implies that $U[|x|+k+4] = 1$. Therefore by $(***)$, necessarily $W[|y|+k+4] = 1$. Since $W[|y|+k+4 \dots |y|+k+|w|+3] = w$, and by $(**)$, this implies that $w = 11.101$, otherwise $W[|y|+k+4] = 0$, which leads to a contradiction with the fact that $U[|x|+k+4] = 1$.

This implies by $(***)$, that $U[|x|+k+3 \dots |x|+k+8] = 111101$. Therefore by $(*)$, necessarily $U[|x|+k+9] = 0$. However by construction of W, since $W[|y|+k+4 \dots |y|+k+|w|+3] = w$, and since $|w| = 5$, we have $W[|y|+k+9 \dots |y|+k+13] = w$. This implies that $W[|y|+k+9] = 1$ since $w = 11.101$, which leads to a contradiction with $(***)$ since $U[|x|+k+9] = 0$.

Case 2: $\mathbf{W[|y|+k+2 \dots |y|+k+|w|+1] = w}$.

In this case, since $W[|y|+k+2] = 0$, this implies by $(**)$ that $W[|y|+k+3] = 0$. Therefore by $(***)$ we have $U[|x|+k+3] = 0$. Hence, since $U[|x|+k+3 \dots |x|+k+|u|+2] = u$, then by $(*)$, we have $u = 00.010$, otherwise $U[|x|+k+3] = 1$ which leads to a contradiction with the fact that $W[|y|+k+3] = 0$.

This implies by $(***)$, that $W[|y|+k+2 \dots |y|+k+7] = 000010$. Therefore by $(**)$, necessarily $W[|y|+k+8] = 1$. However by construction of U, since $U[|x|+k+3 \dots |x|+k+|u|+2] = u$, and since $|u| = 5$, we have $U[|x|+k+8 \dots |x|+k+12] = u$ which implies that $U[|x|+k+8] = 0$ since $u = 00.010$, which leads to a contradiction with $(***)$ since $W[|y|+k+8] = 1$.

□

4.3. Tower Properties

We are now able to state a set of lemmas that show some interesting technical properties of towers under specific assumptions during the execution of our algorithm. These properties are extensively used in the main proof of our algorithm.

This first lemma is a preliminary result used only in the proof of the two following ones (Lemmas 4.5 and 4.6).

Lemma 4.4. *The robots of a long-lived tower $T = (S, [t_s, t_e])$ consider the same global direction at each time between the Look phase of round t_s and the Look phase of round t_e included.*

Proof. Consider a long-lived tower $T = (S, [t_s, t_e])$.

Call t_{act} the first time in $[t_s, t_e[$ at which the robots of S are edge-activated. Since T is a long-lived tower, t_{act} exists.

When executing $\mathbb{SELF\text{-}STAB\_PEF}\_3$, a robot can change the global direction it considers only when it is edge-activated. Moreover a robot does not change the global direction it considers if it has moved during its previous edge-activation. Besides, during the Look phase of a time t a robot considers the same global direction than the one it considers during the Move phase of time $t-1$.

Therefore, during the Look phase of time t_s the robots of S consider the same global direction. Indeed, if this was not the case; *i.e.,* if the robots of S consider different global directions during the Move phase of time $t_s - 1$; they necessarily move during the Move phase of time $t_s - 1$ (otherwise T is not formed at time t_s), therefore they separate during the Move phase of time t_{act}. This leads to a contradiction with the fact that T is a long-lived tower.

Consider a time $t \in]t_s, t_e[$. If at time t the robots of S are not edge-activated, then during the Move phase of time t the robots of S do not change the global direction they consider.

T is a long-lived tower from time t_s to time t_e included. Therefore if at time $t \in]t_s, t_e[$ the robots of S are edge-activated, then, by definition of a long-lived tower, during the Move phase of time t, the robots of S consider the same global direction.

Since at time t_s the robots of S consider the same global direction, using the two previous arguments by recurrence on each time $t \in]t_s, t_e[$ and the fact that robots change the global directions they consider only during Compute phases, we can conclude that the robots of S consider the same global direction from the Look phase of time t_s to the Look phase of time t_e. □

The following lemma is used to prove, in combination with Lemmas 4.2 and 4.3, the "tower breaking" mechanism since it proves that robots of a long-lived tower synchronously invoke their GiveDirection function after their first edge-activation.

Lemma 4.5. *For any long-lived tower $T = (S, [t_s, t_e])$, any (r_i, r_j) in S^2, and any t less or equal to t_e, we have $WeAreStuckInTheSameDirection()(r_i, t) = WeAreStuckInTheSameDirection()(r_j, t)$ and $IWasStuckOnMyNodeAndNowWeAreMoreRobots()(r_i, t) = IWasStuckOnMyNodeAndNowWeAreMoreRobots()(r_j, t)$ if all robots of S have been edge-activated between t_s (included) and t (not included).*

Proof. Consider a long-lived tower $T = (S, [t_s, t_e])$. Let t_{act} be the first time in $[t_s, t_e[$ where the robots of S are edge-activated. By definition of a long-lived tower, this time exists.

By definition of a long-lived tower and by lemma 4.4, from the Look phase of time t_s to the Look phase of time t_e included, all the robots of S are on the same node and consider

the same global direction. Therefore the values of their respective predicates $NumberOf$-$RobotsOnNode()$, $ExistsEdgeOnCurrentDirection()$, $ExistsEdgeOnOppositeDirection()$ and $ExistsAdjacentEdge()$ are identical from the Look phase of time t_s to the Look phase of time t_e included.

When executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot updates its variables $NumberRobotsPreviousEdgeActivation$ and $HasMovedPreviousEdgeActivation$ respectively with the values of its predicates $NumberOfRobotsOnNode()$ and $ExistsEdgeOnCurrentDirection()$, only during Compute phases of times where it is edge-activated. By the observation made at the previous paragraph, this implies that from the Compute phase of time t_{act} to the Look phase of time t_e included, the robots of S have the same values for their variables $NumberRobots$-$PreviousEdgeActivation$ and $HasMovedPreviousEdgeActivation$.

Then, by construction of the predicates $WeAreStuckInTheSameDirection()$ and I-$WasStuckOnMyNodeAndNowWeAreMoreRobots()$, the lemma is proved.

□

From Lemmas 4.5, 4.2, and 4.3, we can deduce the following corollary stating that there exists a time from which the predicate $WeAreStuckInTheSameDirection()$ of the robots of any infinite long-lived tower are always false. This corollary is useful to prove that our algorithm solves the perpetual exploration problem in the particular case of the presence of an infinite long-lived tower. Indeed, it implies that, if there is an infinite long-lived tower, then it exists a time from which its robots are never stuck forever. Note that, in the particular case where the long-lived tower is composed of the 3 robots of the system, this implies that the 3 robots never change the direction they consider (since these robots cannot have their predicate $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ true) making them able to perpetually explore the highly dynamic ring.

Corollary 4.1. *For any long-lived tower $T = (S, [t_s, t_e])$, if $t_e = +\infty$ then the predicates $WeAreStuckInTheSameDirection()$ of the robots of S cannot be infinitely often true.*

Proof. First, note that if two robots possess two distinct identifiers, then their transformed identifiers are also distinct.

Consider a long-lived tower $T = (S, \theta)$ with $\theta = [t_s, +\infty[$.

Call $t_{act} \geq t_s$ the first time after t_s where the robots of S are edge-activated. By definition of a long-lived tower, t_{act} exists. By Lemma 4.5, after time t_{act}, the robots of S consider the same values of predicates $WeAreStuckInTheSameDirection()$ and $IWas$-$StuckOnMyNodeAndNowWeAreMoreRobots()$.

Assume by contradiction that after t_{act} the predicates $WeAreStuckInTheSameDirection()$ of the robots of S are infinitely often true. Then by construction of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, after time t_{act}, all the robots of S call the function GiveDirection infinitely often and at the same instants of times.

If among the robots of S two have the same chirality, to keep forming T they need to consider the same values of bits each time the function GiveDirection is called. Here the robots have to consider the same values of bits infinitely often (since the two robots call the function GiveDirection infinitely often). Each time a robot executes the function

GiveDirection it reads the bit next (in a round robin way) to the bit read during its previous call to the function GiveDirection. Call i_1 and i_2 the two respective transformed identifiers of two robots forming T such that these two robots possess the same chirality. By the previous observations, to keep forming T, i_1^ω and i_2^ω must share an infinite common factor. However according to Lemma 4.2 this is not possible. Therefore there exists a time t_{end} at which these two robots consider two different bits. When the robots call the function GiveDirection, they are edge-activated (by definition of the predicate $WeAreStuckInTheSameDirection()$), therefore at time t_{end}, T is broken.

Similarly, if among the robots of S two have not the same chirality, to keep forming T they need to consider different values of bits each time the function GiveDirection is called. Here the robots have to consider different values of bits infinitely often (since the two robots call the function GiveDirection infinitely often). Each time a robot executes the function GiveDirection it reads the next bit (in a round robin way) of the bit read during its previous call to the function GiveDirection. Call j_1 and j_2 the two respective transformed identifiers of two robots forming T such that these two robots possess a different chirality. By the previous observations, to keep forming T, j_1^ω must possess an infinite suffix S such that an infinite suffix of j_2^ω is equal to $\overline{S}$. This is equivalent to say that j_1^ω and $\overline{j_2^\omega}$ must possess an infinite common factor. However according to Lemma 4.3 this is not possible. Therefore there exists a time t_{end} at which these two robots consider two identical bits. When the robots call the function GiveDirection, they are edge-activated, therefore at time t_{end}, T is broken.

Hence in both cases the long-lived tower T is broken, which leads to a contradiction with the fact that $\theta = [t_s, +\infty[$. □

The next lemma state one of the more fundamental properties of our algorithm: If there exists an eventual missing edge, then it is not possible for all the robots to be stuck forever on one or both of the extremities of this edge.

Lemma 4.6. *If there exists an eventual missing edge, then all long-lived towers have a finite duration.*

Proof. Consider that there exists an edge e of $\mathcal{G}$ which is missing forever from time $t_{missing}$. Consider the execution from time $t_{missing}$.

Call u and v the two adjacent nodes of e, such that v is the adjacent node of u in the clockwise direction.

By contradiction assume that there exists a long-lived tower $T = (S, \theta)$ such that $\theta = [t_s, +\infty[$. Exactly 3 robots are executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, so $|S|$ is either equals to 2 or 3. We want to prove that all the robots of T have their predicates $WeAreStuckInTheSameDirection()$ infinitely often true. By contradiction, assume that there exists a robot r_i of S, such that it exists a time t_i in θ such that for all time t greater or equal to t_i its predicate $WeAreStuckInTheSameDirection()$ is false.

Call $t_{act} \geq t_s$, the first time after time t_s where the robots are edge-activated. Since T is a long-lived tower, t_{act} exists. By Lemma 4.5, from time $t_{act}+1$ the robots of S possess the same values of predicates $WeAreStuckInTheSameDirection()$. By assumption of contradiction,

from time $t_{false} = max\{t_{act} + 1, t_i\}$ the predicates $WeAreStuckInTheSameDirection()$ of all the robots of S are false.

We recall that by definition of a long-lived tower and by Lemma 4.4 all the robots of S are on the same node and consider the same global direction from the Look phase of time t_s to the Look phase of time t_e included.

Case 1: $|S| = 3$. From time t_s the predicates $NumberOfRobotsOnNode()$ of the robots of S are equal to 3. When executing $\mathbb{SELF\text{-}STAB\_PEF\_3}$, a robot updates its variables $NumberRobotsPreviousEdgeActivation$ with the value of its predicate $NumberOfRobotsOnNode()$, only during Compute phases of times where it is edge-activated. Therefore, from time t_{false}, the robots of S have their variables $NumberRobotsPreviousEdgeActivation$ equal to 3. Hence, from time t_{false} their predicates $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ are false, since the condition $NumberOfRobotsOnNode() > NumberRobotsPreviousEdgeActivation$ is false.

Since from time t_{false}, the predicates $WeAreStuckInTheSameDirection()$ of the robots of S are also false, then from time t_{false} the robots of S always consider the same global direction.

Without lost of generality, assume that, from time t_{false}, the robots of S consider the clockwise direction. All the edges of $\mathcal{G}$ except e are infinitely often present, therefore the robots of S reach node u in finite time. However e is missing forever, hence in finite time, the predicates $WeAreStuckInTheSameDirection()$ of all the robots are true. This leads to a contradiction.

Case 2: $|S| = 2$. Assume, without lost of generality, that T is formed of r_1 and r_2.

If, after t_{false}, the 2-long-lived tower does not meet $\mathbf{r_3}$, then by similar arguments than the one used for the case 1 we prove that there is a contradiction.

Now consider the case where the 2-long-lived tower meets $\mathbf{r_3}$. If at a time $t' > t_{false}$, the robots of S meet r_3 it is either because the two entities (the tower and r_3) move during the Move phase of time $t'-1$ while considering two opposed global directions or because the two entities consider the same global direction but one of the entity cannot move (an edge is missing in its direction) during the Move phase of round $t'-1$. Let $t'_{act} \geq t'$ be the first time after time t' included where the three robots are edge-activated. All the edges of $\mathcal{G}$ except e are infinitely often present therefore t'_{act} exists. In both cases, thanks to the update at time $t'-1$ of the variables $HasMovedPreviousEdgeActivation$ and $NumberRobotsPreviousEdgeActivation$ of the robots, during the Move phase of time t'_{act} the robots of the two entities consider opposed global directions. The two entities separate them during the Move phase of this time. Moreover, from this separation, as long as r_3 is alone on its node it does not change the global direction it considers. Similarly, from this separation, as long as the robots of S do not meet r_3, their predicates $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ are false, and since from time t_{false} their predicates $WeAreStuckInTheSameDirection()$ are false, they do not change the global direction they consider.

Hence, in finite time after time t'_{act} the two entities are located respectively on the two extremities of e. However e is missing forever, therefore in finite time, the predicates $WeAreStuckInTheSameDirection()$ of the robots of T are true. This leads to a contradiction.

In both cases a contradiction is highlighted. Therefore, after t_{false} all the robots of S have their predicates $WeAreStuckInTheSameDirection()$ infinitely often true. Then we can use the contrapositive of Corollary 4.1 to prove that T is broken, which leads to a contradiction with the fact that $\theta = [t_s, +\infty[$. □

While proving the correctness of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, we decompose the proof in multiple cases depending on the presence/absence of k-long-lived tower in the execution. The following lemma states that, in the case where there is no long-lived tower in the execution, then the execution contains only configurations with either three isolated robots or one 2-short-lived tower and one isolated robot.

Lemma 4.7. *No execution containing only configurations without a long-lived tower reaches a configuration where three robots form a tower.*

Proof. Assume that there is no long-lived tower in the execution. The robots can cross at most one edge at each round. Each node has at most 2 adjacent edges in $\mathcal{G}$. Moreover each robot considers at each instant time a direction. Assume, by contradiction that 3 robots form a tower T at a time t. Let $t' \geq t$ be the first time after time t where the robots of T are edge-activated. There is no 3-long-lived tower in the execution, therefore during the Move phase of time t', the robots of T consider two opposed global directions. However there are three robots, and two different global directions, hence, during the Move phase of time t', two robots of T consider the same global direction. Therefore there exists a 2-long-lived tower, which leads to a contradiction. □

The following lemma is a technical preliminary used in the proof of Lemmas 4.9 and 4.10.

Lemma 4.8. *In every execution, if a tower involving 3 robots is formed at time t, then at time $t-1$ a 2-long-lived tower is present in ε.*

Proof. Assume that a tower T of 3 robots is formed at time t.

First note that if there exists a 2-long-lived tower $T' = (S, [t_s, t_e])$ such that $t-1 \in [t_s, t_e[$, it is possible for T to be formed.

Now we prove that if there is no 2-long-lived tower at time $t-1$ then T cannot be formed at time t. Assume that at time $t-1$ there is no 2-long-lived tower. Let us consider the three following cases.

Case 1: There is a tower $\mathbf{T'}$ of 3 robots at time $\mathbf{t-1}$. The tower T' must break at time $t-1$, otherwise there is a contradiction with the fact that T is formed at time t. Hence the robots of T' are edge-activated at time $t-1$. While executing

$\mathbb{SELF\text{-}STAB\_PEF\_3}$ the robots consider a direction at each round. There are only two possible directions. Therefore, for the tower T' to break at time $t-1$, two robots of T' consider the same global direction, while the other robot of T' considers the opposite global direction. This implies that the three robots cannot be present on the same node at time t, since $n \geq 4$.

Case 2: There is a 2-short-lived tower T' at time $t-1$. For the three robots to form T at time t, they must be edge-activated at time $t-1$. By definition of a 2-short-lived tower, the two robots of T' consider two opposed global directions during the Move phase of time $t-1$. Since the robots can cross at most one edge at each round, it is not possible for the three robots to be on the same node at time t, which leads to a contradiction with the fact that T is formed at time t.

Case 3: There are 3 isolated robots at time $t-1$. For the three robots to form T at time t, they must be edge-activated at time $t-1$. The robots can cross at most one edge at each round. Each node has at most 2 adjacent edges present in $\mathcal{G}$. Moreover each robot considers at each instant time a direction. Therefore it is not possible for the three robots to be on the same node at time t, which leads to a contradiction with the fact that T is formed at time t.

□

The following lemma shows that our algorithm ensures that the absence of 3-long-lived tower is a closed property.

Lemma 4.9. *Every execution starting from a configuration without a 3-long-lived tower cannot reach a configuration with a 3-long-lived tower.*

Proof. Assume that $\mathcal{E}$ starts from a configuration which does not contain a 3-long-lived tower. By contradiction, let γ be the first configuration of $\mathcal{E}$ containing a 3-long-lived tower $T = (S, [t_s, t_e])$.

Let $t_{act} \geq t_s$ be the first time after time t_s where the 3 robots of T are edge-activated. By definition of a long-lived tower, t_{act} exists.

Lemma 4.8 implies that the configuration at time $t_s - 1$ contains a 2-long-lived tower. Hence, since γ contains the first 3-long-lived tower of $\mathcal{E}$, at time t_s a 2-long-lived tower and a robot meet to form T. The meeting between these two entities can happen either because both of them move in opposed global directions during the Move phase of time $t_s - 1$, or because, during the Move phase of time $t_s - 1$, the two entities consider the same global direction but one of the entity cannot move (an edge is missing in its direction). In both cases; thanks to the update of the variables $HasMovedPreviousEdgeActivation$ and $NumberRobotsPreviousEdgeActivation$ at time $t_s - 1$; during the Move phase of time t_{act} the two entities consider opposed global directions. Hence, the two entities separate during the Move phase of time t_{act}, therefore there is a contradiction with the fact that T is a 3-long-live tower. □

The following lemma is a technical lemma used commonly by Lemmas 4.11 and 4.12.

Lemma 4.10. *Consider an execution $\mathcal{E}$ without any 3-long-lived tower. If a 2-long-lived tower T is formed at a time t_s, then during the Look phase of time $t_s - 1$, a tower T' of 2 robots involving only one robot of T is present. Moreover, during the Move phase of time $t_s - 1$, the robot of T involved in T' does not move while the other robot of T moves.*

Proof. Consider an execution $\mathcal{E}$ without any 3-long-lived tower. Assume that at time t_s a 2-long-lived tower $T = (S, [t_s, t_e])$ is formed.

First note that if there exists a tower T' of 2 robots at time $t_s - 1$, such that only one robot of T' is involved in T and such that this robot does not move during the Move phase of time $t_s - 1$, then it is possible for T to be formed. Now we prove that T can be formed at time t_s only in this situation.

Assume, by contradiction, that there is no tower of 2 robots during the Look phase of time $t_s - 1$. This implies that, at time $t_s - 1$ either the three robots are involved in a 3-short-lived tower T_3 (case 1) or the three robots are isolated (case 2).

Case 1: Call t, the time of the formation of T_3. At time $t_s - 1$ the robots of T_3 are edge-activated, otherwise T cannot be formed at time t_s. By definition of a 3-short-lived tower, during the Move phase of time $t_s - 1$ the robots of T_3 separate. While executing SELF-STAB_PEF_3, each robot considers at each instant time a direction. Therefore, during the Move phase of time $t_s - 1$ two robots of T_3 consider the same global direction. These two robots are still on the same node at time t_s, hence only these two robots can be involved in T. However, since these two robots are on the same node at least from time t and consider the same global direction when they are edge-activated during the Move phase of time $t_s - 1$, there is a contradiction with the fact that T is formed at time t_s.

Case 2: At time $t_s - 1$ the robots of T must be edge-activated, otherwise there is a contradiction with the fact that T starts at time t_s.

Since there is no long-lived tower at time $t_s - 1$ then by Lemma 4.8, at time t_s it is not possible to have a tower of 3 robots. Then since at time t_s, T is formed, it exists at time t_s a tower of 2 robots. For two robots to form a tower at time t_s, during the Move phase of time $t_s - 1$, they either both move while considering two opposed global directions or they consider the same global direction but one of the robot cannot move (an edge is missing in its direction). In both cases, thanks to the update of their variables $Number\text{-}RobotsPreviousEdgeActivation$ and $HasMovedPreviousEdgeActivation$ during the Compute phase of time $t_s - 1$, during the Move phase of the first time greater or equal to t_s where these two robots are edge-activated, they consider opposed global directions and separate them. Therefore there is a contradiction with the fact that T is a 2-long-lived tower starting at time t_s.

Therefore there exists a tower of 2 robots T' during the Look phase of time $t_s - 1$. Now assume, by contradiction that the two robots of T' are involved in T. If T' is a 2-long-lived

tower then during the Move phase of time $t_s - 1$ the two robots of T' are edge-activated and consider two opposed global directions, otherwise there is a contradiction with the fact that T starts at time t_s. If T' is a 2-short-lived tower then during the Move phase of time $t_s - 1$ the two robots of T' are edge-activated (otherwise T cannot be a 2-long-lived tower), and they consider two opposite global directions (by definition of a 2-short-lived tower).

A robot can cross only one edge at each instant time. Since $n \geq 4$ whatever the situation (only one of the robots of T moves or both of the robots of T move during the Move phase of time $t_s - 1$) the two robots of T cannot be again on the same node at time t_s. In conclusion, only one robot of T' is involved in T.

Finally, assume by contradiction, that during the Move phase of time $t_s - 1$, either both the robots of T move (in this case, during the Move phase of time $t_s - 1$ the two robots consider two opposed global directions otherwise they cannot meet to form T) or only the robot of T involved in T' moves while the other robot of T does not move (in this case, during the Move phase of time $t_s - 1$ the two robots consider the same global direction otherwise they cannot meet to form T). In both cases, thanks to the update of the variables $HasMoved$-$PreviousEdgeActivation$ and $NumberRobotsPreviousEdgeActivation$ during the Compute phase of time $t_s - 1$, during the Move phase of the first time after time t_s where the robots of T are edge-activated, they consider two opposed global directions. Therefore there is a contradiction with the fact that T is a 2-long-lived tower starting at time t_s. □

The next two lemmas show that the whole ring is visited between two consecutive formations of 2-long-lived towers if these two towers satisfy some properties. Intuitively, this fundamental property of our algorithm is mainly due to the fact that the robots of a long-lived tower, when they detect they are stuck, wait one edge-activation before trying to break this tower.

Indeed, if this mechanism of "tower breaking" was not delayed from one edge-activation, then it is possible to not solve the perpetual exploration problem since a 2-long-lived tower T may then be formed just after another one T' is broken, without having the whole ring explored at least once by the robots. The formation of T is due to the meeting between a robot r executing the sentinel/visitor scheme and a robot that was involved in T'. As long as r is involved in T, it cannot visit anymore the nodes of the ring since it is involved in a 2-long-lived tower, and since the robots of this tower may be stuck on their node and hence try to break it. While being broken, one of the robots of T can meet the robot not involved in it (and which is therefore executing the sentinel/visitor scheme), creating another 2-long-lived tower, *etc.* Therefore, without this delay, it is possible to prevent the robots to execute the sentinel/visitor scheme and hence to explore the ring.

Lemma 4.11. *Consider an execution $\mathcal{E}$ without any 3-long-lived tower but containing a 2-long-lived tower $T = (S, [t_s, t_e])$. If there exists another 2-long-lived tower $T' = (S', [t'_s, t'_e])$, with $t'_s > t_e + 1$ and such that T' is the first 2-long-lived tower after T in $\mathcal{E}$, then all the nodes of $\mathcal{G}$ have been visited by at least one robot between time t_e and time $t'_s - 1$.*

Proof. Consider an execution $\mathcal{E}$ without any 3-long-lived tower but containing a 2-long-lived tower $T = (S, [t_s, t_e])$. Assume that there exists another 2-long-lived tower $T' = (S', [t'_s, t'_e])$,

with $t'_s > t_e + 1$ and such that T' is the first 2-long-lived tower after T in $\mathcal{E}$.

Since by assumption there is no long-lived tower between the Look phase of time $t_e + 1$ and the Look phase of time $t'_s - 1$ included, then by Lemma 4.7, from the Look phase of time $t_e + 1$ to the Look phase of time $t'_s - 1$ included, if some robots meet they only form 2-short-lived towers. Therefore, by Lemma 4.10, at time $t'_s - 1$ there exists a 2-short-lived tower T_{short}.

To form T', by Lemma 4.10, the configuration C reached is such that T_{short} and the robot of T' not involved in T_{short} are on two adjacent nodes, the adjacent edge to the location of T_{short} in the global direction d is missing at time $t'_s - 1$, and the two robots of T' are edge-activated and consider the global direction d during the Move phase of time $t'_s - 1$. During the Move phase of time t_e the configuration C' is such that the two robots of T are on the same node considering two opposed global direction. Moreover, from the Look phase of time $t_e + 1$ to the Look phase of time $t'_s - 1$ included, if two robots meet they separate once they are edge-activated considering two opposed global directions. Besides, while executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot does not change the global direction it considers if it is isolated. All this implies that to reach C from C' all the nodes of $\mathcal{G}$ have been visited by at least one robot between time t_e and time $t'_s - 1$. □

Lemma 4.12. *Consider an execution $\mathcal{E}$ without any 3-long-lived tower, and let $T_i = (S_i, [t_{s\_i}, t_{e\_i}])$ be the i^{th} 2-long-lived tower of $\mathcal{E}$, with $i \geq 2$. If $T_{i+1} = (S_{i+1}, [t_{s\_i+1}, t_{e\_i+1}])$ exists and satisfies $t_{s\_i+1} = t_{e\_i} + 1$, then all the nodes of $\mathcal{G}$ have been visited by at least one robot between time $t_{s\_i} - 1$ and time $t_{s\_i+1} - 1$.*

Proof. Consider an execution $\mathcal{E}$ without any 3-long-lived tower but containing a 2-long-lived tower $T_i = (S_i, [t_{s\_i}, t_{e\_i}])$, with $i \geq 2$. Assume that there exists another 2-long-lived tower $T_{i+1} = (S_{i+1}, [t_{s\_i+1}, t_{e\_i+1}])$, with $t_{s\_i+1} = t_{e\_i} + 1$. By Lemma 4.10, to form T_{i+1}, a tower of 2 robots involving only one robot of T_{i+1} must be present at time $t_{s\_i+1} - 1$. Moreover T_i is a tower of 2 robots which is present in $\mathcal{G}$ from time t_{s_i} to time $t_{s\_i+1} - 1$. Therefore $S_{i+1} \neq S_i$.

To form T_i, by Lemma 4.10, the configuration C reached at time $t_{s\_i} - 1$ is such that there is a tower T of 2 robots involving only one robot of T_i and the other robot of T_i which are on two adjacent nodes

Similarly, by Lemma 4.10, and since $t_{s\_i+1} = t_{e\_i} + 1$, to form T_{i+1}, the configuration C' reached at time $t_{s\_i+1} - 1$ is such that T_i and the robot of T_{i+1} not involved in T_i are on two adjacent nodes, the adjacent edge to the location of T_i in the global direction d is missing at time $t_{s\_i+1} - 1$, and the two robots of T_{i+1} are edge-activated and consider the global direction d during the Move phase of time $t_{s\_i+1} - 1$. Moreover, since there is no 3-long-lived tower in $\mathcal{E}$, from the Look phase of time t_{s_i} to the Look phase of time $t_{s\_i+1} - 1$ included, if T_i meets the other robot of the system, they form a 3-short-lived tower and hence they separate once they are edge-activated considering two opposed global directions. Besides, while executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot does not change the global direction it considers if it is isolated. All this implies that to reach C' from C all the nodes of $\mathcal{G}$ have been visited by at least one robot between time $t_{s\_i} - 1$ and time $t_{s\_i+1} - 1$. □

4.4. Correctness Proof

Upon establishing all the above properties of towers, we are now ready to state the main lemmas of our proof. Each of these three lemmas below shows that after time t_{max} our algorithm performs the perpetual exploration in a self-stabilizing way for a specific subclass of connected-over-time rings.

Lemma 4.13. $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ *is a perpetual exploration algorithm for the class of static rings of arbitrary size using three fully synchronous robots with distinct identifiers.*

Proof. Assume that $\mathcal{G}$ is a static ring. While executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot considers a direction at each round. Moreover, a robot does not change the global direction it considers if its variable $HasMovedPreviousEdgeActivation$ is true. The variables of a robot are updated during Compute phases of times where it is edge-activated. Since $\mathcal{G}$ is static, this implies that in each round all the robots are edge-activated and are able to move whatever the direction they consider. So, after t_{max} their variables $HasMovedPreviousEdgeActivation$ are always true. Hence, the robots never change their directions.

As (i) the robots have a stable direction, (ii) they always consider respectively the same global direction, and (iii) there always exists an adjacent edge to their current locations in the global direction they consider, the robots move infinitely often in the same global direction. Moreover, as $\mathcal{G}$ has a finite size, this implies that all the robots visit infinitely often all the nodes of $\mathcal{G}$. □

Lemma 4.14. $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ *is a perpetual exploration algorithm for the class of edge-recurrent but non static rings of arbitrary size using three fully synchronous robots with distinct identifiers.*

Proof. Assume that $\mathcal{G}$ is an edge-recurrent but non static ring. Let us study the following cases.

Case 1: There exists at least one 3-long-lived tower in $\mathcal{E}$.

Case 1.1: One of the 3-long-lived towers of $\mathcal{E}$ has an infinite duration.

Denote by $T = (S, [t_s, +\infty[)$ the 3-long-lived tower of $\mathcal{E}$ that has an infinite duration. Call $t_{act} \geq t_s$ the first time after time t_s where the robots of T are edge-activated. By definition of a long-lived tower, t_{act} exists. The variables of a robot are updated during Compute phases of times where it is edge-activated. Therefore, since there are three robots in the system, from time $t_{act} + 1$, the condition "$NumberOfRobotsOnNode() > NumberRobotsPreviousEdgeActivation$" is false for the three robots of T. Therefore from time $t_{act} + 1$ the predicate $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ of each robot of T is false. By Corollary 4.1, eventually, the predicates $WeAreStuckInTheSameDirection()$ of the robots of T are always false, otherwise T is broken in finite time, which leads to a contradiction.

Since eventually the predicates $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ and $WeAreStuckInTheSameDirection()$ of the robots of T are always false, then eventually they never change the global direction they consider. $\mathcal{G}$ is edge-recurrent, therefore there exists infinitely often an adjacent edge to the location of T in the global direction considered by the robots of T, then the robots are able to move infinitely often in the same global direction. Moreover, as $\mathcal{G}$ has a finite size, all the robots visit infinitely often all the nodes of $\mathcal{G}$.

Case 1.2: Any 3-long-lived tower of $\mathcal{E}$ has a finite duration.

By Lemma 4.9, once a 3-long-lived tower is broken, it is impossible to have another 3-long-lived tower in $\mathcal{E}$. Then, $\mathcal{E}$ admits an infinite suffix that matches either case 2 or 3.

Case 2: There exists at least one 2-long-lived tower in $\mathcal{E}$.

Case 2.1: There exists a finite number of 2-long-lived towers in $\mathcal{E}$.

Let $T' = (S', [t'_s, t'_e])$ be the last 2-long-lived tower of $\mathcal{E}$.

There is no 3-long-lived tower in $\mathcal{E}$ at time t'_s (otherwise Case 1 is considered), hence by Lemma 4.9 there is no 3-long-lived tower in $\mathcal{E}$. Moreover, if T' has a finite duration, then $\mathcal{E}$ admits an infinite suffix with no long-lived tower, hence matching case 3.

Otherwise, (*i.e.,* T' has an infinite duration), as in Case 1.1, the robots of T' eventually have their predicates $WeAreStuckInTheSameDirection()$ always false, otherwise, T' is broken in finite time. Let t_{false} be the time from which the robots of T' have their predicates $WeAreStuckInTheSameDirection()$ always false. After time t_{false}, the only case when the robots of T' change the global direction they consider, is when they meet the third robot of the system.

Case 2.1.1: The robots of T′ meet the third robot finitely often.

After the time when the last tower of 3 robots is broken, the robots of T' have their predicates $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ always false. Let t_{break} be the time when the last tower of 3 robots is broken. From time $t = max\{t_{break}, t_{false}\} + 1$ the robots of T' have their predicates $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ and $WeAreStuckInTheSameDirection()$ always false, therefore they never change the global direction they consider. Since $\mathcal{G}$ is edge-recurrent, there is infinitely often an adjacent edge to the location of T' in the direction considered by the robots of T'. This implies that they are able to move infinitely often in the same global direction. Moreover, as $\mathcal{G}$ has a finite size, this implies that all the nodes of $\mathcal{G}$ are visited infinitely often.

Case 2.1.2: The robots of T′ meet the third robot infinitely often.

Consider the execution after time t_{false}. The robot not involved in T' does not change its direction while it is isolated. Similarly, the robots of T' maintain their directions at least until they meet the third robot. Moreover, when

the robots of T' meet the third robot of the system, they form a 3-short-lived tower. Therefore once they are edge-activated, they separate them considering opposed global directions. Then, we can deduce that all the nodes of $\mathcal{G}$ are visited between two consecutive meetings of T' and the third robot. As T' and the third robot infinitely often meet, all the nodes of $\mathcal{G}$ are infinitely often visited.

Case 2.2: There exist an infinite number of 2-long-lived towers in $\mathcal{E}$.

By Lemmas 4.11 and 4.12, we know that between two consecutive 2-long-lived towers (from the second one), all the nodes of $\mathcal{G}$ are visited. As there is an infinite number of 2-long-lived towers, the nodes of $\mathcal{G}$ are infinitely often visited.

Case 3: There does not exist a long-lived tower in $\mathcal{E}$.

Then, we know, by Lemma 4.7, that $\mathcal{E}$ contains only configurations with either three isolated robots or one 2-short-lived tower and one isolated robot.

We want to prove the following property. If during the Look phase of time t, a robot r is located on a node u considering the global direction gd, then there exists a time $t' \geq t$ such that, during the Look phase of time t', a robot is located on the node v adjacent to u in the global direction gd and considers the global direction gd.

Let $t" \geq t$ be the smallest time after time t where the adjacent edge of u in the global direction gd is present in $\mathcal{G}$. As all the edges of $\mathcal{G}$ are infinitely often present, $t"$ exists.

(i) If r crosses the adjacent edge of u in the global direction gd during the Move phase of time $t"$, then the property is verified.

(ii) If r does not cross the adjacent edge of u in the global direction gd during the Move phase of time $t"$, this implies that r changes the global direction it considers during the Look phase of time t. While executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot can change the global direction it considers only during Compute phases of times where it is edge-activated and involved in a tower. Let $t_{act} \geq t$ be the first time after time t such that during the Move phase of time t_{act}, r does not consider the global direction gd. Let r' be the robot involved in a tower with r at time t_{act}. Since there are only 2-short-lived towers in the execution, the two robots r and r' consider two opposed global directions during the Move phase of time t_{act}. Therefore during the Move phase of time t_{act}, r' is on node u considering the global direction gd. By applying case (ii) by recurrence, we can say that from the Move phase of time t to the Move phase of time $t"$ there always exists a robot on node u considering the global direction gd. Therefore during the Move phase of time $t"$ a robot moves on node v. Since the robot does not change the global direction they consider during Look phases, during the Look phase of time $t" + 1$ this robot still considers the global direction gd.

This prove the property. By applying recurrently this property to any robot, we prove that all the nodes of $\mathcal{G}$ are infinitely often visited.

Thus, we obtain the desired result in every cases. □

Lemma 4.15. $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ *is a perpetual exploration algorithm for the class of connected-over-time but not edge-recurrent rings of arbitrary size using three fully synchronous robots with distinct identifiers.*

Proof. Consider that $\mathcal{G}$ is a connected-over-time but not edge-recurrent ring. This implies that there exists exactly one eventual missing edge e in $\mathcal{G}$. Denote by $\mathcal{E}^1$ the maximal suffix of $\mathcal{E}$ in which the eventual missing edge never appears. Let $t_{missing}$ the time after which e never appears again. Let us study the following cases.

Case 1: There exists at least one 3-long-lived tower in $\mathcal{E}^1$.

According to Lemma 4.6, this 3-long-liver tower is broken in finite time. Moreover, once this tower is broken, according to Lemma 4.9, it is impossible to have a configuration containing a 3-long-lived tower. Then, $\mathcal{E}^1$ admits an infinite suffix that matches either case 2 or 3.

Case 2: There exists at least one 2-long-lived tower in $\mathcal{E}^1$.

Case 2.1: There exists a finite number of 2-long-lived towers in $\mathcal{E}^1$.

According to Lemma 4.6, the last 2-long-lived tower is broken in finite time. Since by Lemma 4.9, it cannot exist 3-long-lived tower in $\mathcal{E}^1$, then $\mathcal{E}^1$ admits an infinite suffix with no long-lived tower hence matching Case 3.

Case 2.2: There exist an infinite number of 2-long-lived towers in $\mathcal{E}^1$.

By Lemmas 4.11 and 4.12, we know that between two consecutive 2-long-lived towers (from the second one), all the nodes of $\mathcal{G}$ are visited. As there is an infinite number of 2-long-lived towers, all the nodes of $\mathcal{G}$ are infinitely often visited.

Case 3: There does not exist a long-lived tower in $\mathcal{E}^1$.

By Lemma 4.7, all configurations in $\mathcal{E}^1$ contain either three isolated robots or one 2-short-lived tower and one isolated robot.

(1) We want to prove the following property. If during the Look phase of a time t in $\mathcal{E}^1$, a robot considers a global direction gd and is located on a node at a distance $d \neq 0$ in G (G is the footprint of $\mathcal{G}$) from the extremity of e in the global direction gd, then it exists a time $t' \geq t$ such that, during the Look phase of time t', a robot is on a node at distance $d-1$ in G from the extremity of e in the global direction gd and considers the global direction gd. Let v be the adjacent node of u in the global direction gd.

Let t" $\geq t$ be the smallest time after time t where the adjacent edge of u in the global direction gd is present in $\mathcal{G}$. As all the edges of $\mathcal{G}$ except e are infinitely often present and as u is at a distance $d \neq 0$ in G from the extremity of e in the global direction gd, then the adjacent edge of u in the global direction gd is infinitely often present in $\mathcal{G}$. Hence, t" exists.

(i) If r crosses the adjacent edge of u in the global direction gd during the Move phase of time t", then the property is verified.

(ii) If r does not cross the adjacent edge of u in the global direction gd during the Move phase of time t", this implies that r changes the global direction it considers during the Look phase of time t. While executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot can change the global direction it considers only during Compute phases of times where it is edge-activated and involved in a tower. Let $t_{act} \geq t$ be the first time after time t such that during the Move phase of time t_{act}, r does not consider the global direction gd. Let r' be the robot involved in a tower with r at time t_{act}. Since there are only 2-short-lived towers in the execution, the two robots r and r' consider two opposed global directions during the Move phase of time t_{act}. Therefore during the Move phase of time t_{act}, r' is on node u considering the global direction gd. By applying case (ii) by recurrence, we can say that from the Move phase of time t to the Move phase of time t" there always exists a robot on node u considering the global direction gd. Therefore during the Move phase of time t" a robot moves on node v. Since the robot does not change the global direction they consider during Look phases, during the Look phase of time t" $+ 1$ this robot still considers the global direction gd.

This proves the property.

(2) We now want to prove that there exists a time $t_{reachExtremities}$ in $\mathcal{E}^1$ from which one robot is forever located on each extremity of e pointing to e.

First, we want to prove that a robot reaches one of the extremities of e in a finite time after $t_{missing}$ and points to e at this time. If it is not the case at time $t_{missing}$, then there exists at this time a robot considering a global direction gd and located on a node u at distance $d \neq 0$ in G from the extremity of e in the global direction gd. By applying d times the property (1), we prove that, during the Look phase of a time $t_{reach} \geq t_{missing}$, a robot (denote it r) reaches the extremity of e in the global direction gd from u (denote it v and let v' be the other extremity of e), and that this robot considers the global direction gd during the Look phase of time t_{reach}.

Then, we can prove that from time t_{reach} there always exists a robot on node v considering the global direction gd. Indeed, note that no robot can cross e in the global direction gd from time t_{reach} since e is missing from time $t_{missing}$. Moreover while executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$, a robot can change the global direction it considers only during Compute phases of times where it is edge-activated and involved in a tower. Therefore if at a time $t_{change} \geq t_{missing}$, r changes the global direction it considers at time t_{reach} this is because it is involved in a tower. Since there are only 2-short-lived towers in the execution, at time t_{change}, r is involved in a tower with a robot r', and r and r' consider two opposed global directions during the Move phase of time t_{change}. Therefore during the Move phase of time t_{change}, r' is on node v considering the global direction gd. By applying this argument by recurrence, we can say that from time t_{reach} there always exists a robot on node v considering the global direction gd.

Now we prove that this is also true for the extremity v' of e. If there exists at time t_{reach} a robot on node v' considering the global direction $\overline{gd}$, or if it exists a robot considering the global direction $\overline{gd}$ on a node u' at distance $d \neq 0$ in G from v' in the

global direction $\overline{gd}$, then by using similar arguments than the one used for v, we can prove the property (2). If this is not the case, this implies that at time t_{reach} all the robots consider the global direction gd. Then in finite time (after time t_{reach}) by the property (1), a robot reaches node v. Since from time t_{reach} there is always a robot on node v, there is a 2-short-lived tower formed. Then by definition of a 2-short-lived tower, there exists a time at which one of the robots of this tower considers the global direction gd while the other considers the global direction $\overline{gd}$. Then we can use the same arguments as the one used previously to prove the property (2).

(3) It stays to prove that in the Case 3 all the nodes are infinitely often visited. We know that from time $t_{reachExtremities}$ one robot is forever located on each extremity of e pointing to e. Call r" the robot that is not on node v (resp. v') and pointing to e at time $t_{reachExtremities}$. Assume, without lost of generality, that at time $t_{reachExtremities}$, r" is on node u' and considers the global direction gd. Then by applying recurrently the property (1) we can prove that, in finite time, all the nodes between the current node of r" at time $t_{reachExtremities}$ and v in the global direction gd are visited and that r" reaches v. Call $t'_{act} \geq t_{reachExtremities}$, the first time after time $t_{reachExtremities}$ where there are two robots on node v that are edge-activated. At time t'_{act}, the robot that is on node v and pointing to e at time $t_{reachExtremities}$ changes the global direction it considers (hence considers $\overline{gd}$) by construction of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ and since the tower formed is a 2-short-lived tower.

We can then repeat this reasoning (with v and v' alternatively in the role of u' and with v' and v alternatively in the role of v) and prove that all nodes of $\mathcal{G}$ are infinitely often visited.

Thus, we obtain the desired result in every cases. □

To conclude the proof, first note that even if the robots can start in a non coherent state, it exists a time t_{max} from which all the robots of the system are in a coherent state (by Lemma 4.1). Then it is sufficient to observe that a connected-over-time ring is by definition either static, edge-recurrent but non static, or connected-over-time but not edge-recurrent. As we prove the correctness of our algorithm from the time the robots are in a coherent state in these three cases in Lemmas 4.13, 4.14, and 4.15 respectively, we can claim the following final result.

Theorem 4.1. $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_3$ *is a self-stabilizing perpetual exploration algorithm for the class of connected-over-time rings of arbitrary size (greater or equal to four) using three fully synchronous robots with distinct identifiers.*

5. Sufficiency of Two Robots for $n = 3$

In this section, we present $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$, a self-stabilizing algorithm solving deterministically the perpetual exploration problem on connected-over-time rings of size equal to 3, using two robots possessing distinct identifiers.

Algorithm 4 $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$

1: **if** $WeAreStuckInTheSameDirection()$ **then**
2: GiveDirection
3: **end if**
4: **if** $IAmStuckAloneOnMyNode()$ **then**
5: OppositeDirection
6: **end if**
7: Update

This algorithm and its proof of correctness make use of definitions, variables, and predicates defined in Subsections 4.1 and 4.2 and of the following new predicate.

$$\begin{aligned} IAmStuckAloneOnMyNode() \equiv \\ & (NumberOfRobotsOnNode() = 1) \\ & \wedge \neg ExistsEdgeOnCurrentDirection() \\ & \wedge ExistsEdgeOnOppositeDirection() \end{aligned}$$

The pseudo-code of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$ is given in Algorithm 4.

Proof of correctness. We now prove the correctness of this algorithm.

First, note that Lemmas 4.1, 4.2, and 4.3 are also true for $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$.

To show the correctness of $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$, we need to introduce some lemmas. We consider that the two robots executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$ are r_1 and r_2. Let t_1, and t_2 be respectively the time at which the robots r_1 and r_2 are in a coherent state. Let $t_{max} = max\{t_1, t_2\}$. From Lemma 4.1, the two robots are in a coherent state from t_{max}. In the remaining of the proof, we focus on the suffix of the execution after t_{max}. The other notations correspond to the ones introduced in Section 4.

Lemma 5.1. *Every execution starting from a configuration without a 2-long-lived tower cannot reach a configuration with a 2-long-lived tower.*

Proof. Assume that $\mathcal{E}$ starts from a configuration which does not contain a 2-long-lived tower. By contradiction, let C be the first configuration of $\mathcal{E}$ containing a 2-long-lived tower $T = (S, [t_s, t_e])$.

Let $t_{act} \geq t_s$ be the first time after time t_s where the 2 robots of T are edge-activated. By definition of a long-lived tower, t_{act} exists.

For a 2-long-lived tower to be formed at time t_s, r_1 and r_2 must meet at time t_s. While executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$, the two robots can meet at time t_s only because they are moving considering opposed global directions during the Move phase of time $t_s - 1$. Therefore, since the variables of a robot are updated only during Compute phases of time where it is edge-activated, during the Look phase of time t_{act}, the predicates $WeAreStuck$-$InTheSameDirection()$ of the two robots are false (since their variables $HasMovedPreviousEdgeActivation$ are true). Moreover, during the Look phase of time t_{act} the predicates $IAmStuckAloneOnMyNode()$ of the two robots are false (since their predicates $NumberOfRobotsOnNode()$ is not equal to 1). Hence during the Move phase of time t_{act} the two robots still consider two opposed global directions. Therefore T is broken at time

t_{act}, which leads to a contradiction with the fact that T is a 2-long-lived tower. This proves the lemma. □

Let t_{act1} (resp. t_{act2}) be the first time in the execution at which the robot r_1 (resp. r_2) is edge-activated. By definition, we have $t_1 = t_{act1}+1$ and $t_2 = t_{act2}+1$. By Lemma 5.1, if there exists a 2-long-lived tower in $\mathcal{E}$, then this 2-long-lived tower is present in the execution from time $t_0 = 0$. In this case $t_1 = t_2 = t_{max}$ and at time $t_{max} - 1$ the robots are edge-activated for the first time of the execution.

Lemma 5.2. *The robots of a long-lived tower $T = (S, [t_s, t_e])$ consider the same global direction at each time between the Look phase of round t_{max} and the Look phase of round t_e included.*

Proof. Consider a long-lived tower $T = (S, [t_s, t_e])$. We know that $t_s = t_0 = 0$, that $t_1 = t_2 = t_{max}$ and that at time $t_{max} - 1$ the robots are edge-activated for the first time of the execution. During the Move phase of time $t_{max} - 1$, the two robots consider the same global direction, otherwise there is a contradiction with the fact that T is a 2-long-lived tower.

When executing $\mathbb{SELF\text{-}STAB\_PEF}\_2$, a robot can change the global direction it considers only when it is edge-activated. Besides, during the Look phase of a time t a robot considers the same global direction than the one it considers during the Move phase of time $t - 1$.

Consider a time $t \in [t_{max}, t_e[$. If at time t the robots of S are not edge-activated, then during the Move phase of time t the robots of S do not change the global direction they consider.

If at time t the robots of S are edge-activated, then during the Move phase of time t, since $t \neq t_e$, the robots of S consider the same global direction, otherwise there is a contradiction with the fact that T is a long-lived tower from time t_s to time t_e.

Since during the Move phase of time $t_{max} - 1$ the robots of S consider the same global direction using the two previous arguments by recurrence on each time $t \in [t_{max}, t_e[$ and the fact that robots change the global directions they consider only during Compute phases, we can conclude that the robots of S consider the same global direction from the Look phase of time t_{max} to the Look phase of time t_e included. □

Lemma 5.3. *For any long-lived tower $T = (S, [t_s, t_e])$, and any $t \leq t_e$, such that the robots of S have been edge-activated twice between t_s included and t not included, we have $WeAreStuckInTheSameDirection()(r_1, t) = WeAreStuckInTheSameDirection()(r_2, t)$.*

Proof. Consider a long-lived tower $T = (S, [t_s, t_e])$. We know that $t_s = t_0 = 0$, that $t_1 = t_2 = t_{max}$ and that at time $t_{max}-1$ the robots are edge-activated for the first time of the execution. Assume that between t_s included and t_e not included, the robots of T are edge-activated two or more times.

By definition of a long-lived tower and by lemma 5.2, from the Look phase of time t_{max} to the Look phase of time t_e included, all the robots of S are on the same node

and consider the same global direction. Therefore the values of their respective predicates $NumberOfRobotsOnNode()$, $ExistsEdgeOnCurrentDirection()$ and $ExistsEdgeOnOppositeDirection()$ are identical from the Look phase of time t_{max} to the Look phase of time t_e included.

Let $t_{act} \geq t_{max}$ be the first time after t_{max} such that the robots of T are edge-activated. By assumption, t_{act} exists. When executing $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$, a robot updates its variables $HasMovedPreviousEdgeActivation$ and $NumberRobotsPreviousEdgeActivation$ respectively with the values of its predicates $ExistsEdgeOnCurrentDirection()$ and $NumberOfRobotsOnNode()$, only during Compute phases of times when it is edge-activated. Therefore, from the Look phase of time $t_{act}+1$ to the Look phase of time t_e included, the robots of S have the same values for their variables $HasMovedPreviousEdgeActivation$ and $NumberRobotsPreviousEdgeActivation$.

The predicate $WeAreStuckInTheSameDirection()$ depends only on the values of the variables $HasMovedPreviousEdgeActivation$, $NumberRobotsPreviousEdgeActivation$ and on the values of the predicates $NumberOfRobotsOnNode()$, $ExistsEdgeOnCurrentDirection()$, and $ExistsEdgeOnOppositeDirection()$. As seen previously, all these values are identical for all the robots of S from the Look phase of time $t_{act}+1$ until the Look phase of time t_e included. This prove the lemma. □

From the Lemmas 5.3, 4.2 and 4.3, by noticing that the robots of a long-lived tower T cannot have their predicates $IAmStuckAloneOnMyNode()$ true as long as their are involved in T, we can again obtain the corollary 4.1 (the proof is not exactly the same since the predicate $IWasStuckOnMyNodeAndNowWeAreMoreRobots()$ does not exist in $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$, however the proof is very similar, therefore not repeated in this section).

Theorem 5.1. $\mathbb{SELF}$-$\mathbb{STAB}\_\mathbb{PEF}\_2$ *is a deterministic self-stabilizing perpetual exploration algorithm for the class of connected-over-time rings of size equals to* 3 *using 2 fully synchronous robots possessing distinct identifiers.*

Proof. Consider that $\mathcal{G}$ is a connected-over-time ring of size 3. First note that even if the robots can start in a non coherent state, by Lemma 4.1, it exists a time t_{max} from which all the robots are in a coherent state. Let us study the following cases occurring when the robots are in a coherent state.

Case 1 : There exists at least one 2-long-lived tower in $\mathcal{E}$.

By Lemma 5.1, once a 2-long-lived tower is broken, it is not possible to have again a 2-long-lived tower in $\mathcal{E}$. Therefore there exists only one 2-long-lived tower T in $\mathcal{E}$.

If T has a finite duration, then by Lemma 5.1, $\mathcal{E}$ admits an infinite suffix with no long-lived tower hence matching Case 2.

If T has an infinite duration, the robots of T eventually have their predicates $WeAreStuckInTheSameDirection()$ always false, otherwise, by the contrapositive of Corollary 4.1, T is broken in finite time, which leads to a contradiction. Let t_{false} be the

time from which the robots of T have their predicates $WeAreStuckInTheSameDirection()$ always false. After time t_{false} the robots of T never change the global direction they consider (since their predicates $IAmStuckAloneOnMyNode()$ cannot be true). Moreover, after time t_{false} there exists infinitely often an adjacent edge to the location of T in the global direction considered by the robots of T, otherwise there exists a time after t_{false} when the predicates $WeAreStuckInTheSameDirection()$ of the robots of T are true, which is a contradiction. Hence after time t_{false} the robots of T are infinitely often able to move in the same global direction. Since $\mathcal{G}$ has a finite size, all the robots visit infinitely often all the nodes of $\mathcal{G}$.

Case 2: There does not exist a long-lived tower in $\mathcal{E}$.

If there is no long-lived tower, this implies that if a tower is formed, then it is a 2-short-lived tower. By the connected-over-time assumption, each node has at least one adjacent edge infinitely often present. This implies that any short-lived tower is broken in finite time. Two cases are now possible.

Case 2.1: There exists infinitely often a 2-short-lived tower in the execution.

Note that, if a tower is formed at a time t, then the three nodes have been visited between time $t-1$ and time t. Then, the three nodes are infinitely often visited by a robot in the case where there exists infinitely often a 2-short-lived tower in the execution.

Case 2.2: There exists a time $\mathbf{t_{isolated}}$ after which the robots are always isolated.

By contradiction, assume that there exists a time t' after which a node u is never visited. This implies that, after time $max\{t_{isolated}, t'\}$, either the robots are always switching their position or they stay on their respective nodes.

In the first case, during the Look phase of each time greater than $max\{t_{isolated}, t'\}$, the respective variables dir of the two robots contain the direction leading to u (since each robot previously moves in this direction). As at least one of the adjacent edges of u is infinitely often present, a robot crosses it in a finite time, that is contradictory with the fact that u is not visited after t'.

The second case implies that both adjacent edges to the location of both robots are always absent after time $t_{isolated}$ (since an isolated robot moves as soon as it is possible, by definition of the predicate $IAmStuckAloneOnMyNode()$), that is contradictory with the connected-over-time assumption.

Thus, we obtain the desired result in every cases. □

6. Conclusion

In this paper, we addressed the open question: "What is the minimal size of a swarm of self-stabilizing robots to perform perpetual exploration of highly dynamic graphs?". We

give a first answer to this question by exhibiting the necessary and sufficient numbers of such robots to perpetually explore any connected-over-time ring, *i.e.,* any dynamic ring with very weak assumption on connectivity: every node is infinitely often reachable from any another one without any recurrence, periodicity, nor stability assumption. More precisely, we showed that necessary and sufficient numbers of robots proved in [5] in a fault-free setting (2 robots for rings of size 3 and 3 robots for rings of size greater than 4) still hold in the self-stabilizing setting at the price of the loss of anonymity of robots.

In addition to the above contributions, our results improve the state of the art in the robot network literature in a couple of ways. First, with the exception of the algorithms from [5], we give the only algorithms dealing with highly dynamic graphs. All previous solutions made some assumptions on periodicity or on all-time connectivity of the graph. Second, our algorithms are the first self-stabilizing algorithms for the problem of exploration, either for static or for dynamic graphs.

This work opens an interesting field of research with numerous open questions. First, we should investigate the necessity of every assumption made in this paper. In particular, relaxing the synchrony assumption by considering other (non fully synchronous) ATOM models [28] or even the CORDA model [26] seems to be a challenging task.

Second, it would be worthwhile to explore other problems in this rather complicated environment, *e.g.,* gathering, leader election, *etc.*. It may also be interesting to consider other classes of dynamic graphs and other classes of faults, *e.g.,* crashes of robots, Byzantine failures, *etc.*.

Acknowledgements

Funding: This work has been partially supported by the ANR project ESTATE (Ref. ANR-16-CE25-0009-03), supported by French state funds managed by the ANR (Agence Nationale de la Recherche).

The authors want to thank Florence Levé from Université de Picardie-Jules Verne (France) for her help on the proofs of Lemmas 4.2 and 4.3.

References

[1] R Baldoni, F. Bonnet, A. Milani, and M. Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 428–445, 2008.

[2] L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *International Symposium on Distributed Computing (DISC)*, pages 312–327, 2010.

[3] L. Blin, M. Potop-Butucaru, and S. Tixeuil. On the self-stabilization of mobile robots in graphs. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 301–314, 2007.

[4] M. Bournat, A. K. Datta, and S. Dubois. Self-stabilizing robots in highly dynamic environments. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 54–69, 2016.

[5] M. Bournat, S. Dubois, and F. Petit. Computability of perpetual exploration in highly dynamic rings. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, page to appear, 2017.

[6] N. Braud-Santoni, S. Dubois, M.-H. Kaaouachi, and F. Petit. The next 700 impossibility results in time-varying graphs. *International Journal of Networking and Computing*, 6(1):27–41, 2016.

[7] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[8] J. Chalopin, P. Flocchini, B. Mans, and N. Santoro. Network exploration by silent and oblivious robots. In *Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 208–219, 2010.

[9] A. Datta, A. Lamani, L. Larmore, and F. Petit. Ring exploration by oblivious agents with local vision. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 347–356, 2013.

[10] G. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Live exploration of dynamic rings. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 570–579, 2016.

[11] E. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communication of the ACM*, 17(11):643–644, 1974.

[12] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *Journal of Algorithms*, 51(1):38–63, 2004.

[13] S. Dolev. *Self-stabilization*. MIT Press, March 2000.

[14] S. Dubois, M.-H. Kaaouachi, and F. Petit. Enabling minimal dominating set in highly dynamic distributed systems. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 51–66, 2015.

[15] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 105–118, 2007.

[16] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theoretical Computer Science*, 411(14-15):1583–1598, 2010.

[17] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. How many oblivious robots can explore a line. *Information Processing Letters*, 111(20):1027–1031, 2011.

[18] P. Flocchini, B. Mans, and N. Santoro. Exploration of periodically varying graphs. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 534–543, 2009.

[19] D. Ilcinkas, R. Klasing, and A. Mouhamadou Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *Colloquium on Structural Information & Communication Complexity (SIROCCO)*, pages 250–262, 2014.

[20] D. Ilcinkas and A. Mouhamadou Wade. On the power of waiting when exploring public transportation systems. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 451–464, 2011.

[21] D. Ilcinkas and A. Mouhamadou Wade. Exploration of the t-interval-connected dynamic graphs: The case of the ring. In *Colloquium on Structural Information & Communication Complexity (SIROCCO)*, pages 13–23, 2013.

[22] R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 744–753, 2006.

[23] F Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Symposium on the Theory of Computing (STOC)*, pages 513–522, 2010.

[24] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. In *24th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 339–355, 2017.

[25] M. Potop-Butucaru, M. Raynal, and S. Tixeuil. Distributed computing with mobile robots: An introductory survey. In *International Conference on Network-Based Information Systems (NBiS)*, pages 318–324, 2011.

[26] Giuseppe Prencipe. Corda: Distributed coordination of a set of autonomous mobile robots. In *4th European Seminar on Advanced Distributed Systems (ERSADS)*, pages 185–190, 2001.

[27] C. Shannon. Presentation of a maze-solving machine. *8th Conference of the Josiah Macy, Jr. Foundation*, pages 173–180, 1951.

[28] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computers*, 28(4):1347–1363, 1999.

[29] S. Tixeuil. *Algorithms and Theory of Computation Handbook*, chapter Self-stabilizing Algorithms, pages 26.1–26.45. Chapman & Hall. CRC Press, Taylor & Francis Group, November 2009.

[30] B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.