# Computing on rings by oblivious robots: a unified approach for different tasks

Gianlorenzo d'Angelo, Gabriele Di Stefano, Alfredo Navarra, Nicolas Nisse, Karol Suchan

# Computing on rings by oblivious robots: a unified approach for different tasks*,**

Gianlorenzo D'Angelo[1], Gabriele Di Stefano[2], Alfredo Navarra[1], Nicolas Nisse[3], and Karol Suchan[4]

[1] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy.
gianlorenzo.dangelo@dmi.unipg.it alfredo.navarra@unipg.it
[2] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Italy.
gabriele.distefano@univaq.it
[3] COATI Project, INRIA/I3S(CNRS/UNSA), France. nicolas.nisse@inria.fr
[4] Facultad de Ingenierìa y Ciencias, Universidad Adolfo Ibàñez, Chile karol.suchan@uai.cl

**Abstract.** A set of autonomous robots have to collaborate in order to accomplish a common task in a ring-topology where neither nodes nor edges are labeled (that is, the ring is anonymous). We present a unified approach to solve three important problems: the exclusive perpetual exploration, the exclusive perpetual clearing, and the gathering problems. In the first problem, each robot aims at visiting each node infinitely often while avoiding that two robots occupy a same node (exclusivity property); in exclusive perpetual clearing (also known as searching), the team of robots aims at clearing the whole ring infinitely often (an edge is cleared if it is traversed by a robot or if both its endpoints are occupied); and in the gathering problem, all robots must eventually occupy the same node. We investigate these tasks in the Look-Compute-Move model where the robots cannot communicate but can perceive the positions of other robots. Each robot is equipped with visibility sensors and motion actuators, and it operates in asynchronous cycles. In each cycle, a robot takes a snapshot of the current global configuration (Look), then, based on the perceived configuration, takes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case it eventually moves to this neighbor (Move). Moreover, robots are endowed with very weak capabilities. Namely, they are anonymous, asynchronous, oblivious, uniform (execute the same algorithm) and have no common sense of orientation. In this setting, we devise algorithms that, starting from an exclusive and rigid (i.e. aperiodic and asymmetric) configuration, solve the three above problems in anonymous ring-topologies.

## 1 Introduction

In the field of robot-based computing systems, we consider $k \geq 1$ robots placed on the nodes of an input graph. Robots are equipped with visibility sensors and motion actuators, and operate in *Look-Compute-Move* cycles in order to achieve a common task (see [19]).

The Look-Compute-Move model considers that in each cycle a robot takes a snapshot of the current global configuration (Look), then, based on the perceived configuration, takes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case it moves to this neighbor (Move). Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move operations is finite but unbounded, and it is decided by the adversary for each robot. Hence, robots that cannot communicate may move based on outdated perceptions. From the practical viewpoint, the Look-Compute-Move model faithfully describes the behavior of some real robots.

* This work has been partially supported by the Research Grant 2010N5K7EB 'PRIN 2010' ARS TechnoMedia (Algoritmica per le Reti Sociali Tecno-mediate)' from the Italian Ministry of University and Research, and by Project ECOS-SUD Chile (Action ECOS C12E03) and the Inria Associated Team AlDyNet.

** Preliminary results concerning this work have been presented in the 15th IEEE IPDPS Workshop on Advances in Parallel and Distributed Computing Models (APDCM) [15].

In the continuous plane, this model is referred in the literature also as the *CORDA* model [28]. The inaccuracy of the sensors used by robots to scan the surrounding environment motivates its discretization. Moreover, robots can model software agents moving on a computer network.

Various problems have been studied in this setting and several algorithms have been proposed for particular topologies such as lines, rings, trees and grids. Here, we propose a unified approach to solve the exclusive perpetual exploration, the exclusive perpetual clearing, and the gathering problems on rings. The relevance of the ring topology is motivated by its completely symmetrical structure. It means that algorithms for rings are more difficult to be devised as they cannot exploit any topological structure, as all nodes look the same. In fact, our algorithms are only based on robots' disposal and not on topology.

We consider a minimalist variant of the Look-Compute-Move model which has very weak hypothesis. Neither nodes nor edges of the graph are labeled and no local memory is available on nodes. Robots are *anonymous*, *asynchronous*, *uniform* (i.e. they all execute the same algorithm), *oblivious* (memoryless) and have no common sense of orientation. Apart for the gathering problem, guided by physical constraints, the robots may also satisfy the *exclusivity property*, according to which at most a node can be occupied by at most one robot [2]. In contrast to the CORDA model in the continuous plane, we assume that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges. Note that, in a discrete asynchronous environment this does not constitute a limitation to the model. In fact, an algorithm cannot take advantages from seeing robots on the edges as the adversary can decide to perform the Look operations only when the robots are on the nodes. On the other hand, if an algorithm takes advantage from the assumption that the robots always occupy nodes, the same algorithm can be applied by adding the rule that if a robot sees another robot on an edge, it just don't move (i.e. it waits until all the robots occupy only nodes). In the following, we denote such model as the *discrete CORDA model*.

The discrete CORDA model received a lot of attention in the recent years. Most of the proposed algorithms consider that the starting configuration is *exclusive*, i.e., any node is occupied by at most one robot, and *rigid*, i.e., asymmetric and aperiodic. An exclusive configuration is called *symmetric* if the ring admits a geometrical *axis of symmetry*, dividing the ring into two specular halves, it is called *periodic* if it is invariable under non-trivial (i.e., non-complete) rotations.

In the following, we review the literature concerning the CORDA model on various graph topologies. For the literature about the three problems under study in different settings, the interest reader can refer to [1, 4, 9, 10, 20, 21, 27].

**Related work.** In the problem of graph exploration with stop [16–18], it is required that each node (or each edge) of the input graph is visited a finite number of times by at least one robot and, eventually, all the robots have to stop. Whereas, the exclusive perpetual graph exploration [2, 3, 7, 8] requires that each robot visits each node of the graph infinitely many times. Moreover, it adds the exclusivity constraint. In [7], first results on $n$-node rings are given. In detail, the paper gives algorithms for $k = 3$ and $n \geq 10$, for $k = n - 5$ (if $n \bmod k \neq 0$), and shows that the problem is infeasible for $k = 3$ and $n \leq 9$, and for some symmetric configurations where $k \geq n - 4$.

Graph clearing (also called graph searching) has been widely studied in centralized [20] and distributed settings (e.g., [21]). The aim is to make the robots clear all the edges of a contaminated graph. An edge is *cleared* if it is traversed by a robot or if both its ends are occupied. However, a cleared edge is recontaminated if there is a path without robots from a contaminated edge to it. The study of graph clearing in the discrete CORDA model when the exclusivity property must be always

satisfied is introduced in [6] where a characterization of the exclusive perpetual graph clearing on tree topologies is given. As far as we know, no results have been proposed in ring topologies for the exclusive perpetual graph clearing problem in the discrete CORDA model.

The gathering problem consists in moving all the robots in the same node and remain there. In [11] and [14], a full characterization of the gathering on grid and tree topologies, respectively, without any multiplicity detection is given. On rings, it has been proven that the gathering is unsolvable if the robots are not empowered by the so-called *multiplicity detection* capability [26], either in its *global* or *local* version. In the former type, a robot is able to perceive whether any node of the graph is occupied by a single robot or more than one (i.e., a *multiplicity* occurs). In the latter type, a robot is able to perceive the multiplicity only if it is part of it. Using the global multiplicity detection capability, in [26], some impossibility results have been proven. Then, several algorithms have been proposed for different kinds of exclusive initial configurations in [12, 25, 26]. These papers left open some cases which have been closed in [13] where a unified strategy for all the gatherable configurations has been provided. With local multiplicity detection capability, an algorithm starting from rigid configurations where the number of robots $k$ is strictly smaller than $\lfloor \frac{n}{2} \rfloor$ has been designed in [22]. In [23], the case where $k$ is odd and strictly smaller than $n-3$ has been solved. In [24], the authors provide an algorithm for the case where $n$ is odd, $k$ is even, and $10 \leq k \leq n-5$. Papers [23] and [24] do not assume that the initial configuration is rigid. The remaining cases with local multiplicity detection are left open and a the design of a unified algorithm for all the cases is still not known.

**Contribution.** In this work, we provide a unified approach for solving different tasks in the discrete CORDA model on ring topologies. Namely, starting from any rigid configuration, we solve the exclusive perpetual exploration, the exclusive perpetual clearing, and the gathering with local multiplicity detection capability. Our algorithms consist of two phases. The first phase is common to all problems and allows $k > 2$ robots to achieve a particular rigid exclusive configuration, denoted below by $\mathcal{C}^*$, in an $n$-node ring, $k < n-2$. The second phase depends on the task. We present an algorithm that, starting from configuration $\mathcal{C}^*$, solves both the exclusive perpetual exploration and the exclusive perpetual clearing problems, for any team of $k$ robots in $n$-node rings, $n > 9$, $5 \leq k < n-3$ (but for $k = 5$ and $n = 10$). Moreover, we design a specific algorithm that, starting from any rigid configuration, solves the exclusive perpetual clearing problem using $n-3$ robots in any $n$-node ring, $n > 9$. Finally, we provide some impossibility results for the exclusive perpetual clearing problem, showing that for $3 \leq n \leq 9$ and $k < n$, or $k \in \{1, 2, 3, n-2, n-1\}$ and $n > 4$, the problem cannot be solved in a $n$-node ring with $k$ robots. All together, we obtain an almost full characterization of exclusive perpetual clearing in rings, leaving only open the cases $(k = 4, n > 9)$ and $(k = 5, n = 10)$. Concerning the gathering problem, we design an algorithm that starting from configuration $\mathcal{C}^*$ solves the problem with local multiplicity detection for any team of $k$ robots in $n$-node rings, $2 < k < n-2$ (note that, if $n = 2$ or $k \geq n-2$, no rigid configuration exists). It is worth noting that for the exclusive perpetual exploration and for the gathering problems, besides providing a unified approach, we solve some open cases.

**Outline.** In the next section we define the notation used in the paper and describe the discrete CORDA model. In Section 3, we propose an algorithm to achieve the special configuration $\mathcal{C}^*$. Exclusive perpetual clearing is formally defined and studied in Section 4. We note that the algorithms given in this section also solve the exclusive perpetual exploration problem. The gathering problem

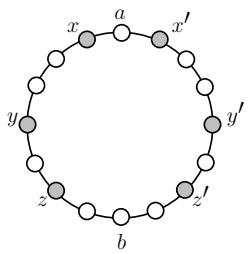


**Fig. 1.** A configuration $\mathcal{C}$ in a ring with 16 nodes. The occupied nodes are depicted in grey.

is considered in Section 5. We then conclude the paper by Section 6 with some possible future research directions.

## 2 Model and Notations

We consider a team of $k \geq 1$ robots located in an $n$-node ring, $n \geq 3$. The ring is *anonymous*, that is its nodes and edges are undistinguishable, and no orientation is provided.

A *configuration* consists of the set of nodes that are occupied by a robot. Note that, it does not take into account the number of robots in each node. A configuration is said *exclusive* if each node is occupied by at most one robot. For $2 \leq k < n-2$, we denote by $\mathcal{C}^*$ the configuration that consists of $k-1$ consecutive occupied nodes, one empty node, one occupied node, and at least further two consecutive empty nodes. An *interval* in a configuration is an inclusion-maximal (possibly empty) subset of consecutive empty nodes, i.e., a subpath of empty nodes that stands between two occupied nodes. For instance, in $\mathcal{C}^*$, there are $k-2$ intervals of length 0, one interval of length 1, and one interval of length $n-k-1 > 1$.

In a configuration $\mathcal{C}$, a *view* from some occupied node $r \in \mathcal{C}$ is a sequence of integers $W(r) = (q_0, q_1, \ldots, q_j)$, $j < k$, that represents the sequence of the lengths of the intervals met when traversing the ring in one direction (clockwise or anti-clockwise) starting from $r$. Abusing the notation, for any $i \leq j$, we refer to $q_i$ as the corresponding interval and its length. Note that, if $\mathcal{C}$ is exclusive, then $j = k-1$ and $\sum_{0 \leq i < j} q_j = n-k$. Moreover, a node $r$ may have 2 distinct views, depending on the direction. Unless differently specified, we refer to $W(r) = (q_0, q_1, \ldots, q_j)$ as the view at $r$ that is minimum in the lexicographical order.

For instance, given the configuration depicted in Figure 1, the possible views of the robot at node $x$ are $(2,1,3,1,2,1)$ and $W(x) = (1,2,1,3,1,2)$.

Let $W(\mathcal{C})$ be the set of the at most $2k$ views (at most two views per occupied node) in the configuration $\mathcal{C}$. The *supermin configuration view* $W^{\mathcal{C}}_{min}$ of the configuration $\mathcal{C}$ is the minimal view in $W(\mathcal{C})$ in the lexicographical order. Note that, in $W^{\mathcal{C}}_{min}$, no interval has length strictly smaller than $q_0$, and, moreover, if $k < n$, then $q_{k-1} > 0$. For instance, $W^{\mathcal{C}^*}_{min} = (q_0, \ldots, q_{k-2}, q_{k-1})$ with $q_0 = \ldots = q_{k-3} = 0$, $q_{k-2} = 1$ and $q_{k-1} = n-k-1$.

For any view $W = (q_0, q_1, \ldots, q_j)$ in a configuration $\mathcal{C}$, we set $\overline{W} = (q_0, q_j, q_{j-1}, \ldots, q_1)$, and $W_i = (q_i, q_{(i+1) \bmod (j+1)}, \ldots, q_{(i+j) \bmod (j+1)})$ denotes the view obtained by reading $W$ starting from $q_i$ as first interval. Note that $W(\mathcal{C}) = \{W_i, \overline{W_i}, \mid 0 \leq i \leq j\}$. Let $I_{\mathcal{C}}$ be the set of intervals $q_i$ such that $W_i$ or $\overline{W_i}$ are equal to $W^{\mathcal{C}}_{min}$. The intervals in $I_{\mathcal{C}}$ are the *supermins* of $\mathcal{C}$. E.g., $|I_{\mathcal{C}^*}| = 1$.

An exclusive configuration is called *symmetric* if the ring admits a geometrical *axis of symmetry*, dividing the ring into two specular halves. An exclusive configuration is called *periodic* if it is invariable under non-trivial (i.e., non-complete) rotations. A configuration which is aperiodic and asymmetric is called *rigid*.

In Figure 1, the intervals (consisting of one node) between occupied nodes $y$ and $z$ and between $y'$ and $z'$ are the supermins of $\mathcal{C}$ and $W^{\mathcal{C}}_{min} = (1, 2, 1, 2, 1, 3)$. $|I_{\mathcal{C}}| = 2$ and $\mathcal{C}$ is aperiodic and has one unique axis of symmetry passing through $a$ and $b$.

We now give some useful properties that are proved in [13]. In particular, Lemma 1 is used to detect possible symmetry or periodicity of a configuration.

*Property 1 ([13]).* Given a view $W$ of a configuration $\mathcal{C}$,

- there exists $0 < i \leq j$ such that $W = W_i$ iff $\mathcal{C}$ is periodic;
- there exists $0 \leq i \leq j$ such that $W = \overline{W_i}$ iff $\mathcal{C}$ is symmetric;
- $\mathcal{C}$ is aperiodic and symmetric iff there exists one unique axis of symmetry.

It follows that if a configuration is rigid, then each occupied node has a view which is different from any other occupied node.

**Lemma 1 ([13]).** *Given a configuration $C$,*

- $|I_{\mathcal{C}}| = 1$ *iff* $\mathcal{C}$ *is either rigid or it admits only one axis of symmetry passing through the supermin;*
- $|I_{\mathcal{C}}| = 2$ *iff* $\mathcal{C}$ *is either aperiodic and symmetric with the axis not passing through any supermin or it is periodic with period* $\frac{n}{2}$*;*
- $|I_{\mathcal{C}}| > 2$ *iff* $\mathcal{C}$ *is periodic, with period at most* $\frac{n}{3}$*.*

We consider a discrete variant of the CORDA model introduced in [28] where the robots have no explicit way of communicate to each other (e.g., they cannot exchange messages). However, they are endowed with visibility sensors allowing each robot to perceive their own position in the graph and the positions of all the other robots. However, when the exclusivity property does not hold, and more than one robot reside at a same node, a robot only perceives a so called *multiplicity*, without the information of the exact number of robots composing it.

The robots proceed by cycles of three phases *Look-Compute-Move*. In the Look-phase, a robot at some node $r$ accesses a *snapshot* of the network that consists of the view $W(r)$. In the Compute-phase, the robot decides its action based on the information it received during the Look-phase. Finally, during the Move-phase, the robot executes its action, i.e., it moves to a neighboring node or stays idle. The environment is fully asynchronous which, in particular, means that the Compute-phase may be executed based on an out-dated view of the network.

We consider a minimalist variant of the model, where the robots have very weak abilities. Robots are *anonymous*, i.e., they do not have identifiers, *uniform*, i.e., they all run the same algorithm, *oblivious*, i.e., memoryless, and they have no *sense of direction*, i.e., they do not agree on a common orientation of the ring. Unless differently specified, two or more robots cannot occupy the same node (*exclusivity property*). When the exclusivity property is not imposed (e.g. for solving the gathering problem), the robots have the so called *local multiplicity detection* capability that is, a robot is able to detect whether the node where it resides is occupied by more than one robot or only by itself, but it is not able to detect the exact number of robots occupying the node. Note that this is the weakest assumption that has to be made to solve the gathering since it has been shown that the gathering is impossible if no multiplicity detection capability is allowed [26].

In contrast to the CORDA model in the continuous plane, we assume that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges. We remark that, in a discrete asynchronous environment this does not constitute a limitation to the model. We call such model the *discrete CORDA model*.

Our goal is to investigate the feasibility of several collaborative tasks with these weak hypothesis. We assume that the starting configuration is rigid and exclusive.

## 3 Reaching configuration $\mathcal{C}^*$

In this section, we propose an algorithm, called ALIGN, in the discrete CORDA model that allows to reach configuration $\mathcal{C}^*$ starting from any exclusive rigid configuration. Algorithm ALIGN will be used in next sections to achieve the configurations suitable for the exclusive perpetual exploration, clearing, or gathering problems. We first describe the algorithm that allows to reach configuration $\mathcal{C}^*$. Then, we prove its correctness.

### 3.1 Algorithm ALIGN

The idea at the basis of Algorithm ALIGN is to exploit the initial rigidity and exclusivity properties. In so doing, we can ensure that one single robot moves at time. The moves performed aim to (lexicographically) reduce the unique supermin configuration view of a rigid configuration in a way that the obtained configuration is still rigid and exclusive, until configuration $\mathcal{C}^*$ is achieved.

By rigidity and exclusivity, the starting configuration has a unique supermin interval and each node has a unique supermin configuration view (see Property 1 and Lemma 1). Therefore, the snapshots provided to the robots allow them to agree on a common view (the unique minimum one) where each robot can identify its position. This ensures that a single robot will move and that the next configuration is still exclusive. Given a configuration $\mathcal{C}$, four rules, called REDUCTION$_i(\mathcal{C})$, $i \in \{-1, 0, 1, 2\}$, are defined below where, for each rule, a single robot is asked to move to an empty node. REDUCTION$_0(\mathcal{C})$ is executed only if the supermin has length at least one. If the supermin has null length, REDUCTION$_1(\mathcal{C})$ is executed if the corresponding move does not create any symmetry. Otherwise, REDUCTION$_2(\mathcal{C})$ is executed if it does not create any symmetry, and REDUCTION$_{-1}(\mathcal{C})$ is executed otherwise. We prove that, starting from any rigid configuration, the move resulting from this algorithm achieves a new rigid configuration. The only exception is configuration $\mathcal{C}^s$ such that $W_{min}^{\mathcal{C}^s} = (0, 1, 1, 2)$. In fact, from such a configuration, any single move would generate either a symmetric configuration or configuration $\mathcal{C}^s$ itself. In this case, we first perform REDUCTION$_1(\mathcal{C}^s)$, obtaining the symmetric configuration $\mathcal{C}'$ such that $W_{min}^{\mathcal{C}'} = (0, 0, 2, 2)$, then we perform REDUCTION$_1(\mathcal{C}')$ which leads to $\mathcal{C}^*$. In any case, in the entire algorithm, only one robot is allowed to move at one time. Moreover, we prove that REDUCTION$_i(\mathcal{C})$, $i \in \{0, 1, 2\}$ strictly decreases the supermin. Finally, from some configuration $\mathcal{C}$, applying REDUCTION$_{-1}(\mathcal{C})$ may lead to a configuration $\mathcal{C}'$ with a greater supermin configuration view. However, we prove that, in this case, the next move will reach a new configuration whose supermin configuration view is strictly smaller that the one of $\mathcal{C}$. Since, clearly, $\mathcal{C}^*$ is the rigid configuration with smallest supermin configuration view, this will prove that executing Algorithm ALIGN eventually achieves $\mathcal{C}^*$.

We now formally define the four rules mentioned above. Let $\mathcal{C}$ be any exclusive and rigid configuration and let $W_{min}^{\mathcal{C}} = (q_0, q_1, \ldots, q_{k-1})$ be its unique supermin configuration view. Let $\ell_1$ be the smallest integer such that $q_{\ell_1} > 0$ and let $\ell_2$ be the smallest integer such that $q_{\ell_2} > q_{\ell_1}$. That is, if $\ell_1 > 0$ and $\ell_2 > \ell_1 + 1$, $W_{min}^{\mathcal{C}} = (0, \ldots, 0, q_{\ell_1}, 0, \ldots, 0, q_{\ell_2}, q_{\ell_2+1}, \ldots, q_{k-1})$. Let $a, b, c$ and $d$

be the nodes between the intervals $q_0$ and $q_{k-1}$, $q_{\ell_1}$ and $q_{\ell_1+1}$, $q_{\ell_2}$ and $q_{\ell_2+1}$, and $q_{k-2}$ and $q_{k-1}$, respectively.

- **REDUCTION$_0(\mathcal{C})$:** The robot at $a$ moves to its neighbor in the interval $q_0 > 0$. Then, the new configuration is $(q_0 - 1, q_1, \ldots, q_{k-2}, q_{k-1} + 1)$;
- **REDUCTION$_1(\mathcal{C})$:** The robot at $b$ moves to its neighbor in the interval $q_{\ell_1} > 0$. Then, the new configuration is $(q_0, q_1, \ldots, q_{\ell_1-1}, q_{\ell_1} - 1, q_{\ell_1+1} + 1, \ldots, q_{k-1})$;
- **REDUCTION$_2(\mathcal{C})$:** The robot at $c$ moves to its neighbor in the interval $q_{\ell_2} > 0$. Then, the new configuration is $(q_0, q_1, \ldots, q_{\ell_2-1}, q_{\ell_2} - 1, q_{\ell_2+1} + 1, \ldots, q_{k-1})$;
- **REDUCTION$_{-1}(\mathcal{C})$:** The robot at $d$ moves to its neighbor in the interval $q_{k-1} > 0$. Then, the new configuration is $(q_0, q_1, \ldots, q_{k-2} + 1, q_{k-1} - 1)$.

The pseudo-code of algorithm ALIGN is formally given in Figure 2 and described later. It is clear from the definition of the rules that, from an exclusive rigid configuration, only one robot can execute a move and that the reached configuration is still exclusive. Note that, in the case that the configuration is $\mathcal{C}^s$ (i.e. $W^{\mathcal{C}^s}_{min} = (0, 1, 1, 2)$), any REDUCTION move creates a symmetric configuration. In this case, we perform REDUCTION$_1$ which produces the symmetric configuration $\mathcal{C}$ such that $W^{\mathcal{C}}_{min} = (0, 0, 2, 2)$. After this, REDUCTION$_1$ is again performed and it leads to $\mathcal{C}^*$ (i.e $W^{\mathcal{C}^*}_{min} = (0, 0, 1, 3)$). As $\mathcal{C}$ is symmetric, the supermin configuration view can be obtained by reading the ring in both possible directions (i.e. $W^{\mathcal{C}}_{min} = \overline{(W^{\mathcal{C}}_{min})}$). However robot $b$ is unequivocally identified as the single robot on the axis of symmetry and REDUCTION$_1$ corresponds to moving $b$ in an arbitrary direction. In any case $\mathcal{C}^*$ is achieved. In next subsection, we formally prove that $\mathcal{C}^*$ is eventually achieved and that, except for the case of $\mathcal{C}^s$, the obtained intermediate configurations are always rigid.

**Pseudo-code of ALIGN.** The pseudo-code of ALIGN is given in Figure 2 and it is performed by a generic robot $r$. It makes use of procedure REDUCTION$_i$ whose pseudo-code is given in Figure 3 and described later.

Let $q_{min}$ be the first interval of $W^{\mathcal{C}}_{min}$. If $q_{min} > 0$, the algorithm performs REDUCTION$_0$ (lines 2–3). Otherwise, it first tries to perform REDUCTION$_1$ by computing the configuration $C'$ that would be obtained (line 5) and by checking whether $C'$ is symmetric (line 6). In the negative case, REDUCTION$_1$ is performed (line 7), Otherwise, the algorithm tries to perform REDUCTION$_2$ (lines 9–11) and then REDUCTION$_{-1}$ (lines 13–15). If the configuration obtained is still symmetric, then it must be $\mathcal{C}^s$ such that $W^{\mathcal{C}^s}_{min} = (0, 1, 1, 2)$. In this case, REDUCTION$_1$ is performed at line 17. The configuration obtained is $\mathcal{C}$ such that $W^{\mathcal{C}}_{min} = (0, 0, 2, 2)$. At the next step, REDUCTION$_1$ is again performed at line 7.

We now describe the pseudo-code of REDUCTION$_i$ which is given in Figure 3 as performed by each robot. Let $W = (q_0, q_1, \ldots, q_{k-1})$ be the view of $\mathcal{C}$ read by the robot $r$ which performs the procedure and let $W^{\mathcal{C}}_{min} = (\bar{q}_0, \bar{q}_1, \ldots, \bar{q}_{k-1})$. At lines 1–6, the algorithm moves the last robot in a supermin configuration view, that is it performs REDUCTION$_{-1}$. If $(W^{\mathcal{C}}_{min})_{k-2} \geq (W^{\mathcal{C}}_{min})_{k-1}$, then the robot has to move if and only if $W_1 = W^{\mathcal{C}}_{min}$ (line 2), that is, $q_0 = \bar{q}_{k-1}$, $q_1 = \bar{q}_0$, $\ldots$, $q_{k-1} = \bar{q}_{k-2}$, and it has to move towards $q_0$ (line 3) in order to reduce $\bar{q}_{k-1}$ by enlarging $\bar{q}_{k-2}$. If $(W^{\mathcal{C}}_{min})_{k-2} \leq (W^{\mathcal{C}}_{min})_{k-1}$, then the robot has to move if and only if $\overline{W_{k-2}} = W^{\mathcal{C}}_{min}$ (line 5) and it has to move towards $q_{k-1}$ (line 6). Lines 7–9 implement REDUCTION$_0$ which consists in reducing the supermin interval by moving the robot on the largest side of such interval, that is the robot whose view is the supermin one. At lines 10–15 the algorithm performs REDUCTION$_i$ for $i \in \{1, 2\}$.

**Algorithm**: ALIGN
**Input**: Rigid and exclusive configuration $\mathcal{C}$ with view $W = (q_0, q_1, \ldots, q_{k-1})$ as seen from a robot $r$

```
1  Let q_min be the first interval of W^C_min;
2  if q_min > 0 then
3  |  Apply REDUCTION_0(C, W);
4  else
5  |  Let C' be the configuration obtained after REDUCTION_1(C, W);
6  |  if not SYMMETRIC(C') then
7  |  |  Apply REDUCTION_1(C, W);
8  |  else
9  |  |  Let C'' be the configuration obtained after REDUCTION_2(C, W);
10 |  |  if not SYMMETRIC(C'') then
11 |  |  |  Apply REDUCTION_2(C, W);
12 |  |  else
13 |  |  |  Let C''' be the configuration obtained after REDUCTION_-1(C, W);
14 |  |  |  if not SYMMETRIC(C''') then
15 |  |  |  |  Apply REDUCTION_-1(C, W);
16 |  |  |  else
17 |  |  |  |  Apply REDUCTION_1(C, W);
```

**Fig. 2.** Algorithm ALIGN.

If $(W^{\mathcal{C}}_{min})_{\ell_i} \geq (W^{\mathcal{C}}_{min})_{\ell_i+1}$, then a robot has to move if and only if there exists an integer $m$ such that $\bar{q}_0 = q_m$, $\bar{q}_1 = q_{m-1}$, $\ldots$, $\bar{q}_{\ell_i} = q_0$, that is if and only if $\overline{W_m} = W^{\mathcal{C}}_{min}$ and $m = \ell_i$ (line 11). In this case, such robot has to move towards $q_0$ (line 12). If $(W^{\mathcal{C}}_{min})_{\ell_i} \leq (W^{\mathcal{C}}_{min})_{\ell_i+1}$, then a robot has to move if and only if there exists an integer $m$ such that $\bar{q}_0 = q_m$, $\bar{q}_1 = q_{m+1}$, $\ldots$, $\bar{q}_{\ell_i} = q_{k-1}$, that is if and only if $W_m = W^{\mathcal{C}}_{min}$ and $k - 1 - m = \ell_i$ (line 14). In this case, such robot has to move towards $q_{k-1}$ (line 15).

### 3.2 Correctness

We consider a rigid exclusive configuration $\mathcal{C}$ with unique (by Lemma 1) supermin configuration view $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$. We prove that, when one of the four rules is applied by Algorithm ALIGN, the resulting configuration $\mathcal{C}'$ is still rigid. Moreover, in the case of the first three rules, the supermin configuration view of $\mathcal{C}'$ is strictly smaller than $W^{\mathcal{C}}_{min}$. In the case of REDUCTION$_{-1}$, we must consider the next move to strictly reduce the supermin configuration view.

Since $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$ is the supermin configuration view, no interval has length smaller than $q_0$ and $q_1 \leq q_{k-1}$. Therefore, if $q_0 > 0$ and REDUCTION$_0$ is applied, the view $(q_0 - 1, q_1, \ldots, q_{k-2}, q_{k-1} + 1)$ is clearly the unique supermin configuration view of the resulting configuration $\mathcal{C}'$. By Lemma 1, we obtain:

*Property 2.* The configuration $\mathcal{C}'$ obtained by applying REDUCTION$_0$ in the rigid exclusive configuration $\mathcal{C}$ with $q_0 > 0$ is rigid. Moreover, $W^{\mathcal{C}}_{min} > W^{\mathcal{C}'}_{min}$ (in lexicographical order).

Algorithm ALIGN performs REDUCTION$_0$ until it reaches a rigid exclusive configuration $\mathcal{C}$ with supermin configuration view $W^{\mathcal{C}}_{min} = (0, q_1, \ldots, q_{k-1})$ (i.e., $q_0 = 0$). In this case, REDUCTION$_0$ cannot be applied as otherwise there would be a collision, that is, a multiplicity is created but at this stage we want to avoid it. Therefore REDUCTION$_1$, REDUCTION$_2$ or REDUCTION$_{-1}$ are applied

**Procedure**: REDUCTION$_i$
**Input**: Rigid and exclusive configuration $\mathcal{C}$ with view $W = (q_0, q_1, \ldots, q_{k-1})$ as seen from a robot $r$

```
if i = −1 then
    if W_1 = W^C_min then
        move towards q_0;
    else
        if the reverse of W_{k−2} = (W^C_min) then
            move towards q_{k−1};
if i = 0 then
    if W = W^C_min then
        move towards q_0;
if i ∈ {1, 2} then
    if for some m, the reverse of W_m = W^C_min and m = ℓ_i then
        move towards q_0;
    else
        if for some m, W_m = W^C_min and k − 1 − m = ℓ_i then
            move towards q_{k−1};
```

**Fig. 3.** Procedure REDUCTION.

depending on the configuration $\mathcal{C}$. In particular, REDUCTION$_1$ is applied if it does not create any symmetry. If $q_0 = 0$, by performing REDUCTION$_1$ we cannot obtain a symmetry except for some particular configurations given in the next lemma.

**Lemma 2.** *Let $\mathcal{C}$ be a rigid exclusive configuration with supermin configuration view $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$, with $2 < k < n - 2$ and $q_0 = 0$. Then, the configuration $\mathcal{C}'$ resulting from the application of* REDUCTION$_1$ *is aperiodic. Moreover, $\mathcal{C}'$ is symmetric if and only if the following conditions hold:*

$$q_i = 0, \text{ for each } i = 0, 1, \ldots, \ell_1 - 1; \tag{1}$$

$$q_{\ell_1} = 1; \tag{2}$$

$$q_{\ell_1+1} + 1 = q_{k-1}; \tag{3}$$

$$\text{the sequence } q_{\ell_1+2}, q_{\ell_1+3}, \ldots q_{k-2} \text{ is symmetric.} \tag{4}$$

*Proof.* By rigidity of $\mathcal{C}$, only one robot can perform REDUCTION$_1$ and then $\mathcal{C}'$ is well defined and admits a view $W = (q'_0, q'_1, \ldots, q'_{k-1}) = (q_0, q_1, \ldots, q_{\ell_1} - 1, q_{\ell_1+1} + 1, \ldots, q_{k-1})$.

If $\mathcal{C}'$ is periodic, by Property 1, there must exist $j > 0$ such that $(q'_{j \bmod k}, q'_{(j+1) \bmod k}, \ldots, q'_{(j+\ell_1) \bmod k}) = (q_0, q_1, \ldots, q_{\ell_1} - 1) = (0, \ldots, 0, q_{\ell_1} - 1)$. Note that, as $q'_{\ell_1+1} = q_{\ell_1+1} + 1 > 0$, then $j > \ell_1 + 1$. Hence, in that case, the view $(q_j, \ldots, q_{k-1}, q_0, \ldots, q_{j-1})$ is a view of $\mathcal{C}$ strictly smaller than $W^{\mathcal{C}}_{min}$, a contradiction. Therefore, $\mathcal{C}'$ is aperiodic.

If equations 1–4 hold, then $\mathcal{C}'$ has a view $W = (0, \ldots, 0, q_{\ell_1+1} + 1, q_{\ell_1+2}, q_{\ell_1+3}, \ldots q_{k-2}, q_{\ell_1+1} + 1)$ which is symmetric with the axis of symmetry passing through the middle of the sequences $q_0, q_1, \ldots, q_{\ell_1} - 1$ and $q_{\ell_1+2}, q_{\ell_1+3}, \ldots q_{k-2}$.

We now show the only if statement. Note that Condition 1 is always satisfied by the hypothesis that $q_0 = 0$ and the definition of $\ell_1$. Let us assume that $\mathcal{C}'$ is symmetric and let $W = (q_0, q_1, \ldots, q_{\ell_1-1}, q_{\ell_1} - 1, q_{\ell_1+1} + 1, \ldots, q_{k-1}) = (0, 0, \ldots, 0, q_{\ell_1} - 1, q_{\ell_1+1} + 1, \ldots, q_{k-1})$.

For the sake of contradiction, let us assume that $q_{\ell_1} > 1$. Then, since $q_{\ell_1} \leq q_{k-1}$ and $q_{\ell_1} - 1 > 0$, it is easy to check that $W$ is the supermin configuration view of $\mathcal{C}'$, and $W < W^{\mathcal{C}}_{min}$. Hence, $q_0$ must be the unique supermin of $\mathcal{C}'$ since otherwise, a supermin interval different from $q_0$ would have been a supermin interval in $\mathcal{C}$, contradicting the fact that $W^{\mathcal{C}}_{min}$ is the supermin minimum view of $\mathcal{C}$. By Lemma 1, since $|I_{\mathcal{C}'}| = 1$ and $\mathcal{C}'$ is symmetric, the (unique) axis of symmetry of $W$ passes through the edge corresponding to $q_0$. However, since $q_{\ell_1} - 1 < q_{k-1}$, $\mathcal{C}'$ is not symmetric, a contradiction. It follows that $q_{\ell_1} = 1$. In this case, the first $\ell_1$ elements of $W$ are 0 and, as before, this sequence is unique and the possible axis of symmetry of $\mathcal{C}'$ passes through the middle of such unique sequence. This implies that $\mathcal{C}'$ is symmetric only if $q_{\ell_1+1}+1 = q_{k-1}$ and that the sequence $q_{\ell_1+2}, q_{\ell_1+3}, \ldots, q_{k-2}$ is symmetric. □

It follows that if $W^{\mathcal{C}}_{min}$ does not satisfy Conditions 1–4, the application of REDUCTION$_1$ results in a rigid configuration. Otherwise, REDUCTION$_2$ is applied unless it creates symmetries. The following Lemma 3 shows that actually, when Conditions 1–4 hold, REDUCTION$_2$ can create symmetries only for some specific configurations.

For the next lemmata, we need further notation. A *pattern* is the set of possible configurations admitting a view that fulfills some rules defined by a string of integer numbers and the following symbols. Let $x$ be an integer number: $x^*$ denotes the repetition of $x$ zero or more times; $x^+$ denotes the repetition of $x$ one or more times; $x^{\{n\}}$ denotes the repetition of $x$ exactly $n$ times. Given a configuration $\mathcal{C}$ we say that $\mathcal{C}$ belongs to a pattern $P$ if it has a view $W$ that matches the rules of the pattern. We denote it by $W \in P$. As an example, the configuration $\mathcal{C}$ with a view $(0, 0, 0, 1, \ldots, 1, 2, 2, \ldots, 2)$ belongs to $(0^{\{3\}}, 1^*, 2^+)$.

**Lemma 3.** *Let $\mathcal{C}$ be a rigid exclusive configuration with supermin configuration view $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$, such that $2 < k < n - 2$, $q_0 = 0$, and Conditions 1–4 hold. Then, the configuration $\mathcal{C}'$ resulting from the application of* REDUCTION$_2$ *is aperiodic. Moreover, $\mathcal{C}'$ is symmetric if and only if one of the following conditions hold:*

$$W^{\mathcal{C}}_{min} \in (0, 1, 1^+, 2); \tag{5}$$

$$W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1). \tag{6}$$

*Proof.* By rigidity of $\mathcal{C}$, only one robot can perform REDUCTION$_2$ and then $\mathcal{C}'$ is well defined and admits a view $W = (q'_0, \ldots, q'_{k-1}) = (q_0, q_1, \ldots, q_{\ell_2} - 1, q_{\ell_2+1} + 1, \ldots, q_{k-1})$.

Because $\mathcal{C}$ satisfies Conditions 1–4, it is straightforward to see that $\mathcal{C}'$ is aperiodic.

If $W^{\mathcal{C}}_{min} \in (0, 1, 1^+, 2)$, by performing REDUCTION$_2$ we obtain either $W = (0, 1, 0, 3)$ or $W = (0, 1, 0, 2, 1^*, 2)$. In the first case, $\mathcal{C}'$ is symmetric with the axis of symmetry passing through the intervals of size 1 and 3. In the second case, $\mathcal{C}'$ is symmetric with the axis of symmetry passing through the single node of interval $q_1$ and either in the middle of the sequence $1^*$ or in the occupied node which separates the two intervals of size 2.

If $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$, by performing REDUCTION$_2$ we obtain either $W \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}}, 1, 0^{\{\ell_1-2\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^*, 0^{\{\ell_1-2\}}, 1)$ or $W \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}}, 1, 0^{\{\ell_1-3\}}, 1)$. In both cases $\mathcal{C}'$ is symmetric with the axis of symmetry passing through the single node of interval $q_1$ and either in the middle of the sequence $1, \{0^{\{\ell_1-1\}}, 1\}^*$ in the first case, or in the middle of the sequence $0^{\{\ell_1-3\}}$ in the second case.

Let us assume that $\mathcal{C}'$ is symmetric. We prove the only if statement by case analysis on $q_{\ell_1+1}$.

- $q_{\ell_1+1} > 0$. Let us first assume that $\ell_1 + 2 < k - 1$. The hypothesis $q_{\ell_1+1} > 0$, implies that $\ell_2 = \ell_1 + 1$ and hence, $W \in (0^{\{\ell_1\}}, 1, q_{\ell_1+1} - 1, q_{\ell_1+2} + 1, S', q_{\ell_1+1} + 1)$, for some sequence $S'$.

Note that $S'$ may be empty if $\ell_1 + 2 = k - 2$, otherwise we set $S' = (S, q_{k-2})$ (where $S$ may be an empty sequence).
The possible axis of symmetry cannot pass through the middle of the initial sequence of 0s because, under the hypothesis that $q_{\ell_1+1} > 0$, we have that $q_{k-1} = q_{\ell_1+1} + 1 > 1 = q_{\ell_1}$ and hence $q_{\ell_1} \neq q_{k-1}$.
Therefore, $W$ contains a subsequence $(q'_j, \ldots, q'_{j+\ell_1}) = (1, 0^{\{\ell_1\}})$ where the sequence of $\ell_1$ 0s is disjoint from the initial sequence of 0s, i.e., $\ell_1 < j$ and $j + \ell_1 < k - 1$. If $\ell_1 + 1 < j$, then the view $W^{\mathcal{C}}_{min}$ has to contain $(q'_j, \ldots, q'_{j+\ell_1})$ or $(q'_j - 1, \ldots, q'_{j+\ell_1})$ (the second case occurs only if $j = \ell_1 + 2$) as a subsequence disjoint from $(q_0, \ldots, q_{\ell_1-1})$. This would constitute another supermin, smaller than or equal to the original one, contradicting the rigidity of $\mathcal{C}$. Therefore, $j = \ell_1 + 1$ and, thus, $q_{\ell_1+1} = 1$. By similar arguments, we show that $\ell_1$ must be equal to 1.
Therefore, $W = (0, 1, 0, q_{\ell_1+2} + 1, S', q_{\ell_1+1} + 1)$, and the axis in $\mathcal{C}'$ passes through the single node of $q_1$ and the middle of sequence $S'$ which thus is symmetric. Hence, $q_{\ell_1+2} + 1 = q_{k-1}$ and, as $q_{\ell_1+1} = 1$ and $q_{k-1} = q_{\ell_1+1} + 1$, then $q_{k-1} = 2$ and $q_{\ell_1+2} = 1$. Since sequence $S'$ is symmetric, we have that $q_{\ell_1+2+m} = q_{k-1-m}$, for all $m = 1, 2, \ldots, \lfloor \frac{k-1-\ell_1-4}{2} \rfloor$. Moreover, by Condition 4, $q_{\ell_1+1+m} = q_{k-1-m}$, for all $m = 1, 2, \ldots, \lfloor \frac{k-1-\ell_1-3}{2} \rfloor$. As $q_{\ell_1+2} = 1$, this implies that $(q_{\ell_1+2}, q_{\ell_1+3}, \ldots q_{k-2}) \in (1^+)$. In conclusion, $W^{\mathcal{C}}_{min} \in (0, 1, 1, 1^+, 2)$.
If $q_{\ell_1+1} > 0$ and $\ell_1 + 2 = k - 1$, we have that $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, q_{\ell_1+1}, q_{\ell_1+1} + 1)$ and $W \in (0^{\{\ell_1\}}, 1, q_{\ell_1+1} - 1, q_{\ell_1+1} + 2)$. By similar arguments as above, $\mathcal{C}'$ is symmetric only if $\ell_1 = 1$ and $q_{\ell_1+1} - 1 = 0$ which implies that $W^{\mathcal{C}}_{min} = (0, 1, 1, 2)$.
Summarizing if $q_{\ell_1+1} > 0$ and $\mathcal{C}'$ is symmetric, then $W^{\mathcal{C}}_{min} \in (0, 1, 1^+, 2)$.

– $q_{\ell_1+1} = 0$. In this case $q_{k-1} = q_{\ell_1+1} + 1 = 1$ and then $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, 0, S, 1)$, where, by Condition 4, $S$ is a symmetric sequence. We first show that the possible axis of symmetry cannot pass through the sequence $0^{\{\ell_1\}}$. Let $W = (q'_0, q'_1, \ldots, q'_{k-1}) \in (0^{\{\ell_1\}}, 1, 0, S', 1)$, for some sequence $S'$, and note that $q'_i = q_i$ for all $i \in \{0, 1, \ldots, k-1\} \setminus \{\ell_2, \ell_2 + 1\}$. If the axis passes through the sequence $0^{\{\ell_1\}}$, then the sequence $(q'_{\ell_1+1}, q'_{\ell_1+2}, \ldots, q'_{k-2}) = (0, S')$ is symmetric. Therefore, since $q'_{\ell_1+1} = 0$, then $q'_{k-2} = 0$. Since $q'_{\ell_2+1} \geq 1$, it follows that $q'_{k-2} \neq q'_{\ell_2+1}$, that is $j - 1 \neq \ell_2 + 1$ and then $q_{k-2} = q'_{k-2} = 0$. Moreover, since also $S$ is symmetric, $q_{\ell_1+2} = 0$ and $\ell_1 + 2 \neq \ell_2$, which implies that $q'_{\ell_1+2} = q_{\ell_1+2} = 0$. By iterating these arguments, we have that $q'_i = q_i = 0$ for all $i \in \{\ell_1 + 1, \ldots, k-2\}$ which implies that $k = n - 2$, a contradiction.
Let us assume that there is an axis not passing through the sequence of $0^{\{\ell_1\}}$. This implies that $W$ contains a subsequence $(q'_j, \ldots, q'_{j+\ell_1}) = (1, 0^{\{\ell_1\}})$ where the sequence of $\ell_1$ zeros is disjoint from the initial sequence of zeros, i.e., $\ell_1 < j$ and $j + \ell_1 < k - 1$.
Three cases may arise:

- REDUCTION$_2$ creates a sequence $0^{\{\ell_1+1\}}$ (i.e., there was in $W^{\mathcal{C}}_{min}$ a sequence $0^{\{\ell_1\}}$ distinct from the initial one). In this case, the axis of symmetry of $\mathcal{C}'$ has to pass through the middle of the unique sequence $0^{\{\ell_1+1\}}$.
  This implies that $W \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1+1\}}, 1, 0^{\{\ell_1\}}, S'')$, where $S''$ is a symmetric sequence. Therefore, $W^{\mathcal{C}}_{min} = (0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}}, 1, 0^{\{\ell_1+1\}}, S'')$ which is a contradiction to the fact that $W^{\mathcal{C}}_{min}$ is the supermin configuration view as there is a sequence of $\ell_1 + 1$ of zeros.
- REDUCTION$_2$ creates a sequence $0^{\{\ell_1\}}$ (disjoint from the initial one). Under this hypothesis, $q_{\ell_2} = 1$, either $W^{\mathcal{C}}_{min} = (0^{\{\ell_1\}}, 1, 0^{\{\ell_1-1\}}, 1, 1)$ or
  $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, 0^{\ell_1-1}, 1, q_{\ell_2+1}, S'', 1)$ where $S''$ is a sequence that may be empty. Because $W^{\mathcal{C}}_{min}$ satisfies Conditions 1–4, the first case may occur only for $\ell_1 = 2$, and in that case, $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$.

Assume that $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, 0^{\ell_1-1}, 1, q_{\ell_2+1}, S'', 1)$. In that case, $W \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}}, q_{\ell_2+1}+1, S'', 1)$.

We first show that the possible axis of symmetry passes through the middle of the initial subsequence $(0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}})$. By contradiction, let us assume that the axis of symmetry passes through another interval which implies that there exists an index $m \geq \ell_2 + 1$ such that $W = \overline{W_m}$ (see Property 1). However, $W < W^{\mathcal{C}}_{min}$ while $\overline{W_m} > \overline{(W^{\mathcal{C}}_{min})_m}$ (because $q_{\ell_2+1}$ increased) and $\overline{(W^{\mathcal{C}}_{min})_m} > W^{\mathcal{C}}_{min}$ (because $W^{\mathcal{C}}_{min}$ is the unique supermin). Therefore $W < \overline{W_m}$, a contradiction.

It follows that the axis of symmetry passes through the middle of the initial subsequence $(0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}})$ and therefore, $q_{\ell_2+1} = 0$ and $S''$ is a symmetric sequence. Summarizing, $W \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1\}}, 1, S'', 1)$ and $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, 0^{\{\ell_1-1\}}, 1, 0, S'', 1) = (0^{\{\ell_1\}}, 1, 0, 0^{\{\ell_1-2\}}, 1, 0, S'', 1)$, where $S''$ is symmetric and, by Condition 4, $(0^{\{\ell_1-2\}}, 1, 0, S'')$ is also symmetric. By the latter symmetry, we have that $S''$ ends with $(0, 1, 0^{\{\ell_1-2\}})$ and by the former one it follows that $S''$ starts with $(0^{\{\ell_1-2\}}, 1, 0)$.

By iterating these arguments, we obtain $S'' \in (0^{\{\ell_1-2\}}, 1, 0, 0^{\{\ell_1-2\}}, 1, 0, \ldots, 0, 1, 0^{\{\ell_1-2\}}, 0, 1, 0^{\{\ell_1-2\}}) = (0^{\{\ell_1-2\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^*, 0^{\{\ell_1-2\}})$ and hence, by plugging $S''$ into $W^{\mathcal{C}}_{min}$, $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$.

- REDUCTION$_2$ does not create a sequence $0^{\{x\}}$, for any $x \geq \ell_1$. In this case, the sequence $(1, 0^{\{\ell_1\}})$ is contained also in $W^{\mathcal{C}}_{min}$. Let $m$ be the position of the first 0 of this sequence in $\overline{W}$. Note that, such sequence does not contain neither $q_{\ell_2}$ nor $q_{\ell_2+1}$. Hence $W \in (0^{\{\ell_1\}}, 1, 0, \ldots, q_{\ell_2}-1, q_{\ell_2+1}+1, \ldots, 1, 0^{\{\ell_1\}}, \ldots, 1)$. Moreover $W < W^{\mathcal{C}}_{min}$ while $\overline{W}_m > \overline{(W^{\mathcal{C}}_{min})_m}$. Hence $\overline{W}_m$ cannot be equal to $W$. It follows that no such axis of symmetry can exist.

In conclusion, if $q_{\ell_1+1} = 0$ and $\mathcal{C}'$ is symmetric, then
$W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$. □

It follows that we can use REDUCTION$_2$ in all the configurations which satisfy Conditions 1–4 but not Conditions 5–6. The next lemma shows that in the remaining cases we can use REDUCTION$_{-1}$, still ensuring that the resulting configuration is rigid.

**Lemma 4.** *Let $\mathcal{C}$ be a rigid exclusive configuration with supermin configuration view $W^{\mathcal{C}}_{min}$. If either $W^{\mathcal{C}}_{min} \in (0, 1, 1, 1^+, 2)$ or $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$, then, the configuration $\mathcal{C}'$ resulting from the application of* REDUCTION$_{-1}$ *is rigid.*

*Proof.* By rigidity of $\mathcal{C}$, only one robot can perform REDUCTION$_{-1}$ and then $\mathcal{C}'$ is well defined.

If $W^{\mathcal{C}}_{min} \in (0, 1, 1, 1^+, 2)$, then $\mathcal{C}'$ admits a view $W \in (0, 1, 1, 1^*, 2, 1)$ which is always rigid. Indeed, there is only one interval of size 0 and only one interval of size 2 which implies that a possible axis can pass only through these two intervals. However, the number of nodes between theses two intervals on one side is different from that on the other side.

If $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$, then $\mathcal{C}'$ admits a view $W \in (0^{\{\ell_1+1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-3\}}, 1)$ which is rigid. Indeed, the axis of symmetry can pass only through the middle of the initial sequence $0^{\{\ell_1+1\}}$ but the two sides of such sequence are different. □

By the above lemma, it follows that if we apply REDUCTION$_{-1}$ to a supermin configuration view $W^{\mathcal{C}}_{min}$ fulfilling Condition 5 or 6, the only case in which the obtained configuration can be symmetric is when $W^{\mathcal{C}}_{min} = (0, 1, 1, 2)$. The correctness of ALIGN then follows from next theorem.

**Theorem 1.** *Let $2 < k < n-2$ robots standing in an $n$-node ring and forming a rigid exclusive configuration, Algorithm* ALIGN *eventually terminates achieving configuration $\mathcal{C}^*$ and all intermediate configurations obtained are exclusive and either rigid or such that the supermin view is $(0,0,2,2)$.*

*Proof.* As ALIGN starts from a rigid exclusive configuration, by Lemma 1, there exists a unique supermin in the initial configuration. Hence exactly one robot moves at one time. Moreover, all the performed movements reduce an interval which is strictly greater than 0 and hence the obtained configuration is exclusive.

First, we assume that the initial configuration is not $\mathcal{C}^s$.

In a current rigid exclusive configuration $\mathcal{C}$ with unique supermin configuration view $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$, we prove that the next move is unique and result in a rigid exclusive configuration.

If $q_0 > 0$, the algorithm performs REDUCTION$_0$. This involves a unique robot and the resulting configuration is rigid by Property 2.

If $q_0 = 0$, a unique robot executes REDUCTION$_1$ if the resulting configuration is rigid. Otherwise, by Lemma 2, $W^{\mathcal{C}}_{min}$ satisfies Conditions 1–4. In that case, a unique robot executes REDUCTION$_2$ if the resulting configuration is rigid. Otherwise, by Lemma 3, $W^{\mathcal{C}}_{min} \in (0,1,1^+,2)$ or $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}},1,\{0^{\{\ell_1-1\}},1\}^+,0^{\{\ell_1-2\}},1)$. In this case, a unique robot executes REDUCTION$_{-1}$. By Lemma 4, as the initial configuration is different from $\mathcal{C}^s$, this results in a rigid configuration.

Since configuration $\mathcal{C}^*$ is the configuration with the smallest supermin configuration view, it only remains to show that each movement reduces the supermin. Hence, in the following, we show that each movement (or each two movements) of ALIGN reduces the supermin.

Let us denote by $W = (q'_0, q'_1, \ldots, q'_{k-1})$ the view of the configuration $\mathcal{C}'$ obtained after the movement. $W$ is the view of $\mathcal{C}'$ at the same node and in the same direction as $W^{\mathcal{C}}_{min}$. Let $W^{\mathcal{C}'}_{min}$ be the supermin configuration view of $\mathcal{C}'$. If the movement is REDUCTION$_0$, then $q'_0 = q_0 - 1$ and hence $W^{\mathcal{C}'}_{min} \leq W < W^{\mathcal{C}}_{min}$. If the movement is REDUCTION$_i$, $i \in \{1,2\}$ then $W = (q_0, q_1, \ldots, q_{\ell_i} - 1, q_{\ell_i+1}+1, \ldots, q_{k-1}) < W^{\mathcal{C}}_{min}$ and therefore $W^{\mathcal{C}'}_{min} \leq W < W^{\mathcal{C}}_{min}$. If the movement is REDUCTION$_{-1}$ it follows that $W^{\mathcal{C}}_{min} \in (0,1,1,1^+,2)$ or $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}},1,\{0^{\{\ell_1-1\}},1\}^+,0^{\{\ell_1-2\}},1)$. In the latter case, $W \in (0^{\{\ell_1+1\}},1,\{0^{\{\ell_1-1\}},1\}^+,0^{\{\ell_1-3\}},1)$ and hence $W^{\mathcal{C}'}_{min} \leq W < W^{\mathcal{C}}_{min}$. In the former case, $W \in (0,1,1,1^*,2,1)$ and hence $W > W^{\mathcal{C}}_{min}$. However, $\mathcal{C}'$ is rigid and does not satisfy Conditions 1–4 and hence the movement performed in $\mathcal{C}'$ is REDUCTION$_1$. Therefore, the configuration $\mathcal{C}''$ obtained after performing REDUCTION$_1$ on $\mathcal{C}'$ is $W'' \in (0,0,2,1^*,2,1)$. Therefore, $W'' < W^{\mathcal{C}}_{min}$.

Let us now assume that the initial configuration is $\mathcal{C}^s$. Note that, this is the only initial configuration with $k = 4$ and $n = 8$ which is rigid and different from $\mathcal{C}^*$. From $\mathcal{C}^s$, REDUCTION$_1$ is performed and the symmetric configuration $\mathcal{C}$ such that $W^{\mathcal{C}}_{min} = (0,0,2,2)$ is achieved. The next movement performed is again REDUCTION$_1$ which leads to $\mathcal{C}^*$ (i.e. $W^{\mathcal{C}^*}_{min} = (0,0,1,3)$) independently from the supermin view. In fact, even if configuration $\mathcal{C}$ is symmetric, robot $b$ is unequivocally identified as the single robot on the axis of symmetry and REDUCTION$_1$ corresponds to moving $b$ in an arbitrary direction. In any case $\mathcal{C}^*$ is achieved. □

## 4 Clearing and Exploring a ring

In this section, we study the exclusive perpetual clearing and exploration problems in the discrete CORDA model. In detail, we study the exclusive perpetual clearing and note that any algorithm provided to this respect also solves the exclusive perpetual exploration and hence in the following we only refer to the clearing.

Let us consider an $n$-node ring ($n \geq 3$) and a team of $1 \leq k \leq n$ robots forming a rigid and exclusive configuration. In the case, $4 < k < n-3$ and $n > 9$ (or $n > 10$ if $k = 5$), we propose an algorithm that makes use of Algorithm Align presented in the previous section. We then propose a specific algorithm for the case $k = n-3$ and $n > 9$. On the impossibility side, we show that for $k \in \{1, 2, 3, n-2, n-1\}$ and $n > 3$, or for $3 \leq n \leq 9$ and $k < n$, there is no algorithm that solves the problem, even if the initial configuration is rigid. The cases $k = 4$ and $(k = 5, n = 10)$ are left as open problems.

### 4.1 Exclusive perpetual clearing

Given a graph $G$ where all edges are *contaminated*, the graph clearing problem consists in coordinating a team of robots to eventually *clear* all edges. The robots occupy the nodes of $G$ and a robot can move along an edge from its current position to a neighboring node. An edge is *cleared* when it is traversed by a robot or if both its endpoints are simultaneously occupied by some robots. However, a cleared edge is instantaneously *recontaminated* if there is a path from one of its endpoint to the endpoint of a contaminated edge and no node of this path is occupied by some robot. This variant of graph clearing is classically referred as *mixed graph searching* [5]. Motivated by physical constraints and following [6], we moreover impose the *exclusivity constraint*, i.e., a node can be occupied by at most one robot.

A *clearing strategy* using $1 \leq k \leq n$ robots consists of choosing a set of $k$ nodes, the *initial positions*, and a sequence of moves of the robots, sliding the robots along the edges to empty neighbors, that eventually clear all edges. For instance, there is no clearing strategy that clears a $n$-node ring using one robot. On the other hand, a possible strategy using two robots is the following: first place two robots at adjacent nodes $u$ and $v$, then slide the robot at $u$ along the empty nodes of the ring until it reaches the other neighbor $w$ of $v$.

In this section, we consider the graph clearing problem in $n$-node rings in the discrete CORDA model. More precisely, we aim at designing algorithms that allow robots to clear a $n$-node ring starting from any rigid exclusive configuration. As our algorithms ensure that all met configurations are rigid and exclusive, and as the robots are oblivious of the cleared edges, the resulting strategies clear the ring *perpetually*, i.e., the whole ring is cleared infinitely often. Moreover, we study the exclusive perpetual exploration. Exclusive perpetual clearing and exclusive perpetual exploration are not equivalent. For instance, one robot always moving clockwise will perpetually explore a ring without clearing it. On the other hand, the above clearing strategy using two robots perpetually clears a ring (one robot is at $v$ and the other one alternate its move from $u$ to $w$ and then from $w$ to $u$) but does not perpetually explore it since the robot at $v$ never moves. The algorithms we propose in the sequel both perpetually explore and clear the rings, exclusively.

### 4.2 Impossibility results

In this section, we show that for $k \in \{1, 2, 3, n-2, n-1\}$ or for $n \leq 9$, no algorithm in the discrete CORDA model allows to clear an $n$-node ring using $k$ robots. For these results we do not assume that the initial configurations are rigid, that is the impossibility results hold on a stronger model. We start with a simple result.

**Lemma 5.** *For any $n > 2$ and for any exclusive configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual clearing problem in a $n$-node ring using $n-1$ robots starting from $\mathcal{C}$.*

*Proof.* In any configuration with $n-1$ occupied nodes, only two robots may move without violating the exclusivity property: the two robots adjacent to the empty node. Since these two robots have the same view of the network, whatever be the algorithm in the discrete CORDA model, they are forced by the adversary to take the same decision. Either they never move and the ring cannot be cleared, or both decide to move to their empty neighbor. In the latter case, their moves can be scheduled (due to the asynchronicity) such that they collide, hence violating the exclusivity property. □

Let us consider the case of two robots in a ring with at least three nodes. Two nodes $u$ and $v$ of an $n$-node ring are called *diametral* if either $n$ is even and there are two shortest paths between $u$ and $v$; or $n$ is odd and the length of the two paths from $u$ to $v$ differ by one. We say that two robots occupy a diametral configuration if they occupy two diametral nodes.

We show that any algorithm for exclusive perpetual clearing on rings with two robots needs to reach a configuration where the two robots occupy two diametral nodes. Then, we show that once the two robots occupy two diametral nodes they cannot break the symmetry and hence they cannot clear the ring. The next theorem follows.

**Theorem 2.** *For any $n > 2$ and for any initial configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual graph clearing problem in a $n$-node ring using $k \leq 2$ robots starting from $\mathcal{C}$.*

*Proof.* Since there is no strategy to exclusively and perpetually clear a ring using one robot, it follows that at least two robots are necessary.

We first give general remarks on the clearing of a ring with two robots, independently of the distributed model of computation. Let us assume only two robots are occupying the nodes of a $n$-node ring, $n > 2$, all edges of which are initially contaminated. If the two robots never occupy adjacent nodes, then the ring will never be cleared. Therefore, consider the first time that such a situation occurs. Let $u$ and $v$ be the two neighbors occupied by the robots at this step. Then, all edges but $\{u, v\}$ are contaminated at this step. Moreover, for the ring to be eventually cleared, there must be a later step such that, up to a symmetry, the robot that was occupying $v$ is now at $w \neq v$ and the other robot reaches $w'$ the neighbor of $w$ on the path between $u$ and $w$ not containing $v$. In particular, this proves that, at some step of any clearing strategy of the ring, the two robots are occupying diametral nodes.

In what follows, we show that no algorithm in the discrete CORDA model can ensure the above properties because the symmetry cannot be broken when the robots pass through diametral nodes. This will prove the theorem.

It is worth noting that in symmetric configurations, any robot allowed to move either resides on the axis of symmetry or it admits a symmetric robot also allowed to move. If an algorithm allows to move two symmetric robots, it might happen that while one robot moves, its symmetric one has only performed its Look phase. This results in a possible *pending move*, that is, the symmetric robot will perform the scheduled move, eventually. We consider an adversarial scheduler that always alternates the moves of the two robots until it reaches a diametral configuration for the first time. That is, it first makes one robot do its *Look-Compute-Move* cycle, and then do the same with the second robot, and so on. By the above remarks, if a diametral configuration is never reached, then the ring cannot be cleared. Moreover, when a diametral configuration is reached, then there are no pending moves. Now there are two cases depending on the parity of $n$. In what follows, the robots always are in diametral configuration. Therefore, whatever be the algorithm used, both robots are forced to move when they look such a configuration.

– Assume first that $n$ is even. Then, since there are no pending moves, the adversarial scheduler can synchronize the two robots such that after their respective moves the configuration has not changed and there still are no pending moves. Indeed, the two robots look and decide before any move and then both of them move before the next look. Going on this way, the robots remain in a diametral configuration and the ring cannot be clear.
– Now consider the case when $n$ is odd. Consider the path between the two robots with an odd number of nodes and let $v$ be the node on this path at same distance from both robots. Then, the adversarial scheduler makes the two robots do their *Look-Compute* actions and then their *Move* action. Since they are in a symmetrical configuration (with axis passing through $v$), they move symmetrically. Doing so, after each move of both robots, i.e., each time they are looking, the node $v$ remains at equal distance from both robots. Therefore, it cannot be reached unless both robots collide in it. Hence, the ring cannot be cleared. □

Let us now consider the case of three robots in a ring with at least four nodes. For ease of presentation, we give identifiers to the robots. Of course, the robots are anonymous in the sense that they are not aware of these identifiers and that no algorithm for clearing the ring can make use of them. However, the adversarial scheduler will use them. Hence, let us call the three robots by $r, r''$ and $r'''$. At any step $s$, we denote by $dist_s(x, y)$ the distance (i.e., the number of consecutive edges) between the nodes occupied by robots $x$ and $y$ at this step (if there is no ambiguity, the subscript will be omitted). Let $\mathcal{C}_c$ be the configuration where the three robots occupy three consecutive nodes. Given any algorithm ALG for exclusively and perpetually clearing a ring with 3 robots, we say that a configuration $\mathcal{C}$ is *bad* if, in this configuration, $dist(r', r'') \leq dist(r'', r''')$ and there exists a robot such that, if this robot executes ALG in configuration $\mathcal{C}$, then the configuration reached after its move is such that $dist(r', r'') > dist(r'', r''')$.

In what follows, we show that any algorithm for exclusively and perpetually clearing a ring with three robots must always avoid the configuration $\mathcal{C}_c$. Then, we show that such an algorithm cannot avoid to reach a bad configuration. Finally, we show that from any bad configuration, it is possible to schedule the three robots such that either they reach the configuration $\mathcal{C}_c$, or (1) each robot is scheduled at least once; (2) this reaches a configuration such that $dist(r', r'') \leq dist(r'', r''')$ and $r''$ has been adjacent to $r'''$ in the meantime; and (3) if the new configuration is not $\mathcal{C}_c$, then from this new configuration, ALG will reach another bad configuration before $r''$ is adjacent to $r'''$.

Since any algorithm for exclusively and perpetually clearing the ring must ensure that $r''$ is infinitely many times adjacent to $r'''$, this proves that such an algorithm cannot exist.

**Theorem 3.** *For any $n > 3$ and for any initial configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual graph clearing problem in a $n$-node ring using 3 robots starting from $\mathcal{C}$.*

*Proof.* First, if $n = 4$, the single node that is not occupied cannot be reached without collision. Therefore, let us assume that $n > 4$.

For purpose of contradiction, let us consider any algorithm ALG that exclusively and perpetually clears the ring with 3 robots. Let us consider the periodic infinite sequence $\mathcal{S}$ of moves of the robots following ALG, subject to a scheduler that alternate the robots, i.e., first $r'$ makes its *Look-Compute-Move* actions, then $r''$, then $r'''$ and so on. The goal of considering such a scheduler is to be able to analyze the behavior of ALG in some particular configurations (without pending moves). Then, taking use of a more clever scheduler, ALG can fail. There are two cases to be considered.

**Case 1.** Let us first assume that $\mathcal{S}$ contains the configuration $\mathcal{C}_c$ where the three robots are occupying three consecutive nodes. Considering the moves just before and just after this configuration, we

can derive two facts. First, in the configuration where two robots are adjacent and the third one is at distance two of the closest of the others, if it is the turn of the third one, it will get closer to the other robots and reached the configuration $\mathcal{C}_c$. Second, in the configuration $\mathcal{C}_c$, if it is the turn of a robot not in the middle, then it moves to its empty neighbor.
Then, let us consider the following adversarial scheduler against ALG. First, it schedules the robots alternatively until they reach configuration $\mathcal{C}_c$. W.l.o.g., say $r''$ is in the middle. From this step, the adversarial schedules twice $r'$, then $r''$ and then twice $r'''$. That is, $r'$ moves to its empty neighbor, then comes back; then $r''$ cannot move, and finally $C$ goes and back. Clearly, ALG cannot exclusively and perpetually clear the ring against the proposed adversarial, a contradiction.

**Case 2.** Second, assume that $\mathcal{C}_c$ never occurs in $\mathcal{S}$. Note that, in any infinite sequence of moves that exclusively and perpetually clear the ring, the three robots must be pairwise adjacent infinitely often.
W.l.o.g. (up to a renaming of the robots), since $\mathcal{S}$ does not contain the configuration $\mathcal{C}_c$, there must be a step $s$ such that $r'$ and $r''$ are occupying adjacent nodes and then a further step $s'$ when $r''$ and $r'''$ are occupying adjacent nodes, such that $r'''$ and $r'$ never are never occupying adjacent nodes between steps $s$ and $s'$ (including $s$ and $s'$). Therefore, between steps $s$ and $s'$, there must be a step $s_0$ such that $x = dist_{s_0}(r', r'') \leq dist_{s_0}(r'', r''') = y$ and after the next step, $dist_{s_0+1}(r', r'') > dist_{s_0+1}(r'', r''')$. Let $\mathcal{C}_0$ be the configuration reached at step $s_0$. Note that $\mathcal{C}_0$ is a bad configuration.
The proof is a case-analysis depending on whether $x = y$ or not and on how Algorithm ALG behaves in configuration $\mathcal{C}_0$, i.e., what are the moves that can be done by robots $r'$, $r''$ and $r'''$ in configuration $\mathcal{C}_0$ when executing ALG.

- Let assume first that $x = y$. Then, the three robots are scheduled simultaneously, that is, all three robots performs a *Look-Compute-Move* cycle, following ALG in configuration $\mathcal{C}_0$.
  Moreover, if $r''$ decides to move in configuration $\mathcal{C}_0$, we make it moving towards $r'$. This is possible since $\mathcal{C}_0$ is symmetric from the point of view of $r''$. Similarly, $r'$ and $r'''$ must move symmetrically: either both of them go towards $r''$, or they move to their neighbor on the path between $r'$ and $r'''$ not containing $r''$.
  There cannot be a collision since otherwise ALG would not be a valid algorithm. Therefore, after each robot has moved, the reached configuration $\mathcal{C}_1$ is such that $dist(r', r'') \leq dist(r'', r''')$ and $r''$ has not met $r'''$.
  There are two cases, depending on $\mathcal{C}_1$.
  - Either $\mathcal{C}_1$ is $\mathcal{C}_c$ and then Case 1. ensures that ALG cannot exclusively and perpetually clear the ring, a contradiction.
  - Or, we go back using the scheduler that alternates the three robots until a new bad configuration is reached. The same discussion as above ensures that another bad configuration will be reached before $r''$ mets $r'''$. Note that, possibly Configuration $\mathcal{C}_1$ is a bad configuration. Again, we process as in Case 2. Therefore, we can avoid forever that $r''$ mets $r'''$, which contradicts the correctness of ALG.
- Second, assume that $x < y$. Note that, since only one robot moves during step $s_0 + 1$ and $dist_{s_0+1}(r', r'') > dist_{s_0+1}(r'', r''')$, this implies that the robot that moves during this step is $r''$. Therefore, $y = x + 1$ and the configuration $\mathcal{C}'_0$ reached after step $s_0 + 1$ is symmetric with $\mathcal{C}_0$. In particular, $r''$ cannot distinguish $\mathcal{C}_0$ and $\mathcal{C}'_0$. Note that, $r'$ must move in configuration $\mathcal{C}'_0$, and $r'''$ must move in configuration $\mathcal{C}_0$. Indeed, otherwise, scheduling alternatively $r''$, then $r'$, then $r''$, then $r'''$, and so on, Algorithm ALG would oscillate between configurations

$\mathcal{C}_0$ and $\mathcal{C}'_0$ which cannot clear the ring since $n > 4$. Moreover, in the configuration $\mathcal{C}'_0$, $r'$ acts in the same way as $r'''$ in the configuration $\mathcal{C}_0$ (by symmetry).
There are three cases to be considered.

- First assume that $x = 0$ and, when executing Algorithm ALG in configuration $\mathcal{C}_0$, robot $r'''$ goes to the neighbor of $r''$. In that case, configuration $\mathcal{C}_c$ is reached then Case 1 ensures that ALG cannot exclusively and perpetually clear the ring, a contradiction.
- Second assume that $x > 0$. In that case, $r''$ and $r'''$ first look and compute in configuration $\mathcal{C}_0$, then $r''$ moves. Note that, because $x > 0$ and $y = x + 1$, $r''$ does not meet $r'''$. Then, $r'$ and $r''$ look and compute in configuration $\mathcal{C}'_0$. Then, $r''$ moves and reaches the configuration $\mathcal{C}_0$. Finally, $r''$ and $r'''$ move. As mentioned above, they must move symmetrically, i.e., in opposite orientation. Therefore, after each robot has moved ($r''$ has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(r', r'') \leq dist(r'', r''')$ and $r''$ has not met $r'''$. We get a contradiction as above.
- The last case to be considered is when $x = 0$ and, when executing Algorithm ALG in configuration $\mathcal{C}_0$, $r'''$ goes to its neighbor that is not adjacent with $r''$. In that case, there are two possibilities for $r'$ in configuration $\mathcal{C}_0$.
  - Either it does not move, in which case, the three robots look and compute in Configuration $\mathcal{C}_0$, then $r'''$ and $r'$ move first (but $r'$ remains on its position) and finally $r''$ moves. Therefore, after each robot has moved ($r''$ has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(r', r'') \leq dist(r'', r''')$ and $r''$ has not met $r'''$. We get a contradiction as above.
  - Otherwise, $r'$ looks, computes and moves. Here the configuration is such that $r''$ is at distance 2 from both other robots. Then, $r'$ and $r'''$ looks, computes and moves. Since their views are identical, they must do the same move.
    If they go to the neighbors of $r''$, we have reached the configuration $\mathcal{C}_c$ and then Case 1 ensures that ALG cannot exclusively and perpetually clear the ring, a contradiction.
    Otherwise, $r''$ looks, computes and moves. Since its view is symmetrical, the adversarial can let it move toward $r'$. Therefore, after each robot has moved ($r''$ has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(r', r'') \leq dist(r'', r''')$ and $r''$ has not met $r'''$. We get a contradiction as above. □

By using similar argument as Theorem 2 the next theorem can be shown.

**Theorem 4.** *For any $n > 2$ and for any exclusive initial configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual graph clearing problem in a $n$-node ring using $n - 2$ robots starting from $\mathcal{C}$.*

*Proof.* If $k = n - 2$, then all the nodes of the ring but two are occupied. If $n \leq 4$, then $k \leq 2$ and hence it is impossible to clear the ring. Let us assume that $n \geq 5$. Two cases may arise: either the two empty nodes are consecutive (see Figure 4a) or not (see Figure 4b). In both cases, all the possible configurations are symmetric and an axis of symmetry either passes through the middle of the interval of empty nodes or through the middle of the two intervals of occupied nodes between the empty nodes, respectively.

In the first case there are only two robots which can move without creating a collision: those close to an empty node. The adversary can force to move only one of these two symmetric robots while the other robot does not perform the look phase (it does not wake up). In the obtained configuration (see Figure 4c) the two empty nodes are not consecutive and the ring is not cleared.

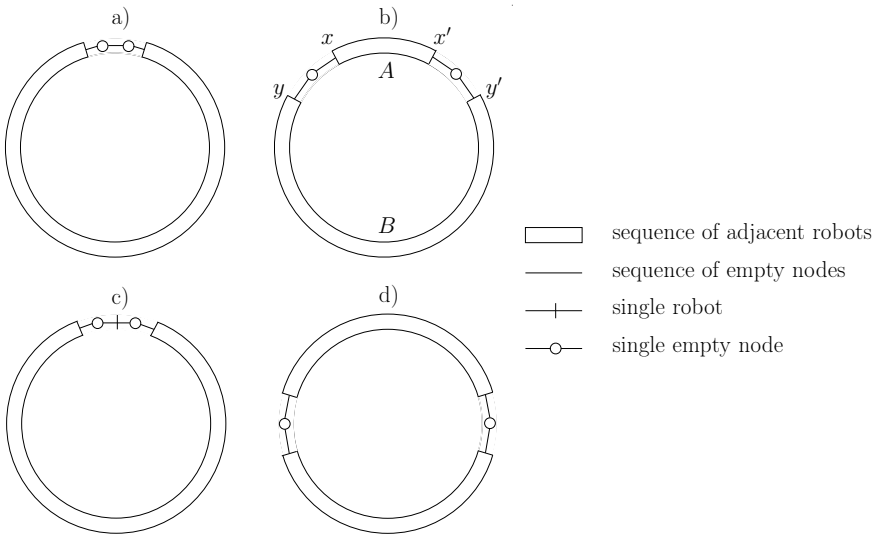


**Fig. 4.** Configurations with $k = n - 2$.

Hence, we can assume without loss of generality that in the initial configuration the two empty nodes are not consecutive.

In such configurations only the three or four robots close to an empty node can move and two cases may arise: the two intervals of occupied nodes between the empty nodes have the same size or not. In the first case, the configuration is periodic with two axis of symmetry and the second axis passes through the two empty nodes (see Figure 4d). In this case, no robot can move as the four robots close to an empty interval are indistinguishable and the adversary can force them to move synchronously causing a collision. Hence let us assume that the two intervals of occupied nodes between the empty nodes have different sizes, which implies that the configuration is symmetric but not periodic.

Let us denote as $A$ and $B$ the sizes of the smallest and the largest intervals of consecutive occupied nodes, respectively. Moreover, we denote ad $x$ and $x'$ the two symmetric robots at the border of $A$ and with $y$ and $y'$ the two symmetric robots at the border of $B$. Note that $x$ ($y$, respectively) cannot be distinguished by $x'$ ($y'$, respectively) and hence they will do the same movements, see Figure 4b for a visualization. There are two possible movements:

1. $x$ and $x'$ move towards $y$ and $y'$, respectively;
2. $y$ and $y'$ move towards $x$ and $x'$, respectively.

By performing the first (second, respectively) movement, $A$ is decreased (increased, respectively) and $B$ is increased (decreased, respectively). We now show that if an algorithm does one of such movements, then it cannot do the other movement in a subsequent step where $A > 1$ and it still holds that $A < B$. By contradiction let us assume that an algorithm first does movement 1 and then movement 2, the other case is symmetric. Let us assume that the adversary moves $x$ towards $y$ but let the symmetric movement of $x'$ towards $y'$ pending, that is $x'$ performs the look and compute phases but it does not move yet. In the obtained configuration if the new $y$ moves towards the new $x$, then, as the configuration is still symmetric, also $y'$ as to move towards $x'$ causing a collision between $x'$ and $y'$ due to the pending move of $x'$. It follows that an algorithm has to do always the same movement: either moving $x$ towards $y$ or moving $y$ towards $x$.

We first analyze the case where an algorithm always do movement 2. If an algorithm moves $y$ towards $x$, then $A$ is increased and $B$ is decreased until either $A = B$ or $A = B - 1$. The case that $A = B$ has been already shown to be impossible. If $A = B - 1$, the adversary can force $y$ to move towards $x$ while the symmetric move of $y'$ remains pending. The configuration obtained is identical to the previous one but the intervals are flipped. Hence the new $y'$ (which is the old $x'$) has to move towards the new $x'$ (which is the old $y'$), causing a collision due to the pending move of the latter. Therefore, an algorithm can only move $x$ towards $y$ (movement 1) until the two empty nodes are adjacent (Figure 4a) or at distance one (Figure 4c). In the case of two empty adjacent nodes, as already discussed, the adversary can force to move only one of two symmetric robots adjacent to an empty node and then the two empty nodes will be at distance one. We then assume that the two empty nodes are at distance one. In this configuration there are three robots that can move: the robot in the middle of the empty nodes which we call $x$ and the two symmetric robots close to an empty node which we call $y$ and $y'$. Two movements are possible: $x$ move towards an arbitrary direction or $y$ and $y'$ move towards $x$. If the first movement is performed, we go back to the configuration with two empty adjacent nodes and from this configuration again the adversary can force to move always the same robot infinitely many times without clearing the ring. If $y$ and $y'$ move towards $x$, the adversary can force to move only one among $y$ and $y'$, let us say $y$. In the obtained configuration, the two empty nodes are at distance two, that is $A = 2 < B$ and, as shown above, the algorithm has to perform movement 1, that is nodes $x$ and $x'$ of the new configuration have to move towards nodes $y$ and $y'$. Note that node $x$ of the new configuration corresponds to node $y$ of the old one. Hence, the adversary can force to move only such node, obtaining again the configuration with the empty nodes at distance one. This two configurations can alternate infinitely many times by moving always the same robot and hence without clearing the ring. □

In the next subsections, we prove that it is possible to clear $n$-node rings using at most $n - 3$ robots for all $n \geq 10$. The next theorem, shows that in all the remaining cases the problem is unsolvable. This is proven by an exhaustive study of the possible configurations and it is based on the following lemmata.

**Lemma 6.** *Let an even number $k$ of robots be in a symmetrical exclusive configuration in an $n$-node ring with $n$ odd. No algorithm starting from (or reaching at some step) such a configuration allows the exclusive perpetual clearing of the ring.*

*Proof.* Indeed, there is a node $v$ that is empty on the axis of symmetry of the initial configuration. Because of the symmetry, the adversarial scheduler can ensure that at each step, two robots occupying symmetrical positions execute the same move. Thus, the axis of symmetry always remains the same. Therefore, if at some step, the vertex $v$ would be occupied, then during the previous step, both its neighbors were occupied and both robots occupying these neighbors would move to $v$. Hence, $v$ cannot be occupied without collision. □

**Theorem 5.** *For any $2 \leq k < n \leq 9$ and for any initial configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual clearing problem in a $n$-node ring using $k$ robots starting from $\mathcal{C}$.*

*Proof.* By the previous results of this section, for any $k \in \{1, 2, 3, n-2, n-1\}$, no algorithm allows $k$ robots to exclusively and perpetually clear an $n$-node ring. Therefore, it only remains to show the theorem for $(k, n) \in \{(4, 7); (4, 8); (5, 8); (4, 9); (5, 9); (6, 9)\}$. We prove it by an exhaustive study of the possible configurations in each case.

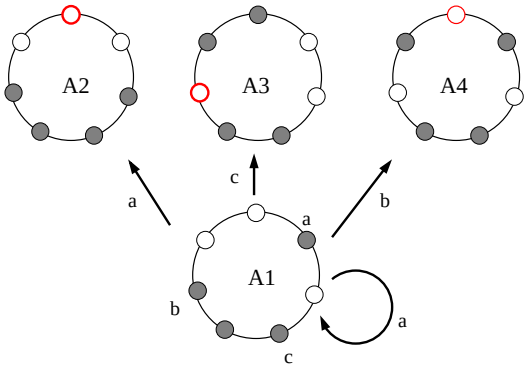


**Fig. 5.** Theorem 5. Case $(k, n) = (4, 7)$. Grey nodes are the occupied ones.

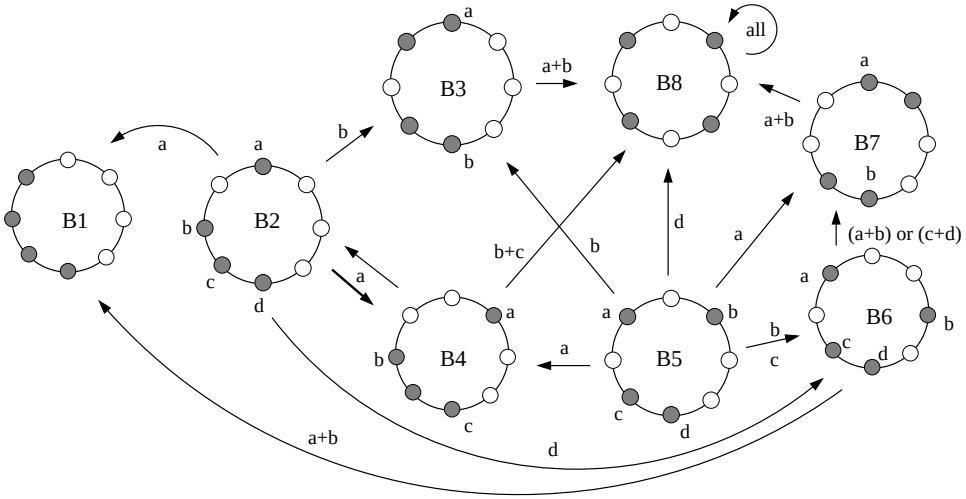


**Fig. 6.** Theorem 5. Case $(k, n) = (4, 8)$. Grey nodes are the occupied ones.

In what follows, we consider any algorithm ALG for exclusive perpetual graph clearing. The adversary schedules the moves sequentially: the adversary chooses a set of robots (generally one robot or two robots having symmetrical positions) that *look*, then they *compute* and *move* simultaneously. In particular, this implies that the robots always look the current configuration, i.e., there is no problem of asynchrony. Of course, in any configuration, any Algorithm must execute some move since otherwise the system would never change and the ring cannot be cleared.

In the proof below, we use the Figures 5-8. In these figures, grey nodes are the occupied nodes. An arc from Configuration $C1$ to Configuration $C2$, with label $a$ means that Robot $a$ in Configuration $C1$ moves such that Configuration $C2$ is reached. Note that the labels of robots in $C1$ and $C2$ may be not consistent because of the symmetries.

– **Case** $(k, n) = (4, 7)$**.** In that case, there are only four distinct configurations that are depicted in Figure 5. Moreover, Configurations $A2, A3$ and $A4$ are symmetric and satisfy the requirements of Lemma 6. Therefore, by Lemma 6, no algorithm can exclusively and perpetually clear a ring if it reaches one of these three configuration. It only remains to prove that Algorithm ALG cannot

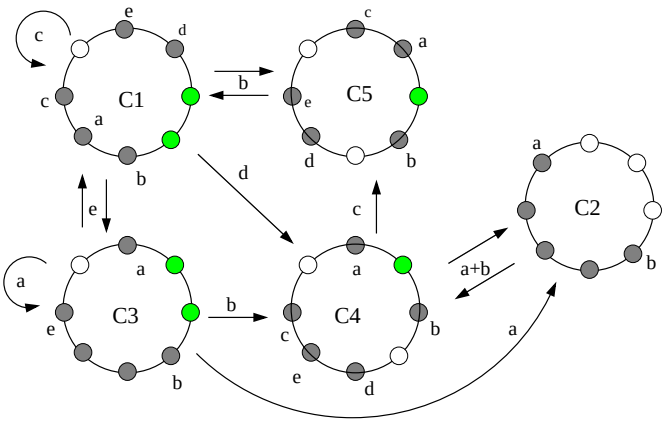


**Fig. 7.** Theorem 5. Case $(k, n) = (5, 8)$. Grey nodes are the occupied ones.

exclusively and perpetually avoid these configurations. Indeed, in Configuration $A1$, moving robot $b$ or $c$, or moving robot $a$ toward $c$ leads to $A4$, $A3$ or $A2$ respectively. The only remaining move consists in moving $a$ toward $b$. However, perpetually executing this move cannot clear the ring.

– **Case** $(k, n) = (4, 8)$**.** In that case, there are 8 distinct configurations that are depicted in Figure 6. If Algorithm ALG reaches Configuration $B1$, then only two robots can move, that is, those with an empty neighbor. In this case, the adversary can force both to move hence obtaining Configuration $B6$. If Configuration $B8$ is reached then the clearing fails since, in such a configuration, all robots have the same view. Therefore, the adversary can schedule all robots simultaneously such that they all move clockwise, which does not clear the ring. Now we show that any Algorithm ALG eventually reaches Configuration $B8$ and thus cannot clear the ring.

  In Configuration $B3$, only robots $a$ and $b$ can move since, otherwise, the adversary could simultaneously move the other two robots (that have the same view) so that they collide in their common empty neighbor. Moreover, since robots $a$ and $b$ have the same view, the adversary can schedule both of them simultaneously such that Configuration $B8$ is reached. In Configuration $B7$, all robots have the same view. Then, the adversary can schedule robots $a$ and $b$ such that Configuration $B8$ is reached. Therefore, if ALG reaches Configurations $B3$ or $B7$, it will eventually reach Configuration $B8$, which, by previous paragraph, cannot clear the ring. Thus, ALG must never reach Configurations $B3$ or $B7$.

  Now, let us consider Configuration $B6$. In such a configuration, robots $a$ and $b$ have the same view and robots $c$ and $d$ have the same view. If ALG allows robots $c$ and $d$ to move, the adversary can move them simultaneously to reach Configuration $B7$. If robots $a$ and $b$ can move, the adversary can move them simultaneously and symmetrically to reach Configuration $B7$. Note that, $B1$ cannot be reached from $B6$ since it would be the opposite move to the only one allowed from $B1$. Thus, ALG must never reach Configuration $B6$.

  In Configuration $B2$, if Algorithm ALG makes $a$ move towards $b$, then the adversary can reach Configuration $B1$, and subsequently $B6$. If robot $b$ (resp. robot $d$) can move to its empty neighbor, then the adversary can reach Configuration $B3$ (resp., Configuration $B6$). Therefore, any Algorithm ALG that exclusively and perpetually clears the ring must never reach Configuration

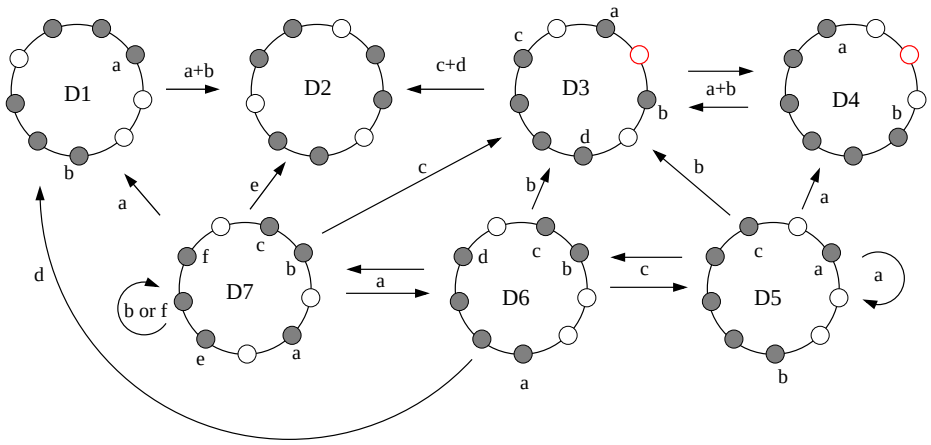


**Fig. 8.** Theorem 5. Case $(k, n) = (6, 9)$. Grey nodes are the occupied ones.

$B2$ or, in such a configuration, only allows $a$ to move to its empty neighbor that is symmetric to $c$.

In Configuration $B4$, robots $b$ and $c$ have the same view. If Algorithm ALG allows them to move, the adversary can move them simultaneously to reach Configuration $B8$. Therefore, only robot $a$ can move. However, in that case, Configuration $B2$ is reached and, by previous paragraph, this would lead to a strategy where only robot $a$ moves, oscillating between Configurations $B4$ and $B2$, which does not clear the ring.

To conclude, in Configuration $B5$, all robots have distinct views, but any move of one of the robot would lead to one of the previous configurations that we prove to be forbidden.

– **Case** $(k, n) = (5, 8)$**.** In that case, there are 5 distinct configurations that are depicted in Figure 7.

If Algorithm ALG reaches Configuration $C2$, then only two robots can move, that is, those with an empty neighbor. In this case, the adversary can force both to move hence obtaining Configuration $C4$. Therefore, in Configuration $C4$, Algorithm ALG must not allow robots $a$ and $b$ (that have the same view) to move since, otherwise, Configuration $C2$ would be reached again or they collide in their common neighbor. Therefore, in Configuration $C4$, only robots $c$ and $d$ (that have the same view) can move to their unique empty neighbor. Hence, any Algorithm ALG that clears the ring and that reaches Configuration $C4$ must reach Configurations $C5$.

We now show that any Algorithm ALG that clears the ring must reach Configurations $C1, C4$ or $C5$. Indeed, otherwise (since $C2$ cannot be reached), it would mean that Algorithm ALG perpetually stays in Configuration $C3$ (the single possibility is then that robot $a$ oscillates and that the neighbor of $b$ is never occupied nor cleared) which clearly does not clear the ring. By previous paragraph, any Algorithm ALG that clears the ring must reach Configurations $C1$ or $C5$.

In Configuration $C5$, robots $c$ and $e$ (that have the same view) cannot move, since otherwise, the adversary could make them collide in their common empty neighbor. Now, we prove that robots $a$ and $d$ (that have the same view) cannot move. Indeed, otherwise, the adversary schedule $c$ and $e$ that cannot move, then $a$ and $d$ that move toward $b$ reaching Configuration $C4$, then $b$ (the robot labelled $b$ in $C5$) that cannot move because now its two neighbors are occupied, and then $a$ and $d$ (that must be able to move by previous paragraph) to go back to Configuration

$C5$. Perpetually executing this schedule is valid since any robot infinitely often executes its cycle. However, the node between $c$ and $e$ is never cleared. Therefore, in Configuration $C5$, any Algorithm Alg that exclusively and perpetually clears the ring can only allow robot $b$ to move. In particular, any Algorithm Alg that clears the ring must reach Configuration $C1$.
Now, assume that, in Configuration $C1$, Algorithm Alg allows robots $b$ to move. Therefore, in Configuration $C1$, the adversary can schedule robot $b$ to reach Configuration $C5$, where robots $a, c, d$ and $e$ are scheduled without being able to move (by previous paragraph). Finally, the adversary schedules robot $b$ to reach back Configuration $C1$. Perpetually executing this schedule is valid since any robot infinitely often executes its cycle. However, the node between $c$ and $e$ is never cleared. Therefore, any Algorithm Alg that exclusively and perpetually clears the ring must not allow robot $b$ to move in Configuration $C1$.
Similarly, let us assume that, in Configuration $C1$, Algorithm Alg allows robots $d$ to move. Therefore, in Configuration $C1$, the adversary can schedule robot $d$ to reach Configuration $C4$ where the adversary schedules robots $a, b$ and $e$ that cannot move by above paragraphs. Then, the adversary schedules robot $c$, which reaches Configuration $C5$ without clearing the neighbor of $b$. Then, by above paragraph, the adversary can schedule robots $a, c, d$ and $e$ without any move, and then schedules robot $b$ to reach back Configuration $C1$, still without clearing the ring. Therefore, any Algorithm Alg that exclusively and perpetually clears the ring must not allow robot $d$ to move in Configuration $C1$.
Since we proved that any Algorithm Alg that exclusively and perpetually clears the ring must reach Configuration $C1$ and that, in such a configuration, neither $d$ nor $b$ can move, then robot $e$ must be able to move. Indeed, otherwise, Alg would reach Configuration $C1$ and would remain in this configuration, robot $c$ being the only one to be able to move, without clearing the ring.
Now, assume that, in Configuration $C3$, Algorithm Alg allows robot $b$ to move. In that case, the adversary can schedule $b$ (in $C3$), then $a, b, d, e$ in $C4$ without any move, then $c$ in $C4$, then $b$ in $C5$, and finally $e$ in $C1$ reaching back $C3$ without clearing the ring the green vertices are never cleared). Therefore, in $C3$, only $e$ may move towards $a$ or $a$ may move towards $e$.
To conclude, it is sufficient to note that, when $C1$ is eventually reached (which must be by above discussion), then the adversary can ensure to oscillate between Configurations $C1$ and $C3$, such that all robots execute a cycle infinitely often (only $e, c$ and $a$ actually may move) without clearing the ring.

– **Case** $(k, n) = (6, 9)$**.** In that case, there are 7 distinct configurations that are depicted in Figure 8.
By Lemma 6, if Algorithm Alg reaches Configuration $D3$ or $D4$, then the clearing fails. If Configuration $D2$ is reached, all robots have the same view, if one of them can move, the one with the same common empty neighbor can also move which would create a collision. Therefore, if Algorithm Alg reaches Configuration $D2$, then the clearing fails. If Configuration $D1$ is reached, only robots $a$ and $b$ may move (if the other two robots with an empty neighbor move, they would collide in it). In that case, robots $a$ and $b$ are scheduled simultaneously which reaches Configuration $D2$ that is forbidden. Therefore, any Algorithm Alg that clears the ring must never reach one of these four configurations.
Therefore, the only possible moves are the following. In Configuration $D5$, robot $a$ can move towards $b$, and $c$ can move. In Configuration $D6$, only robots $c$ and $a$ can move. In Configuration $D7$, $b$ and $f$ can move, and $a$ can move towards $e$. Hence, the adversary executes the following schedule. Arriving (or starting) in $D7$, it sequentially moves $d, e, c$ (that do not move), then$f$ twice, then $a$ and, if $a$ does not move, it moves $b$ twice and goes on. Arriving from $D7$ by moving $a$

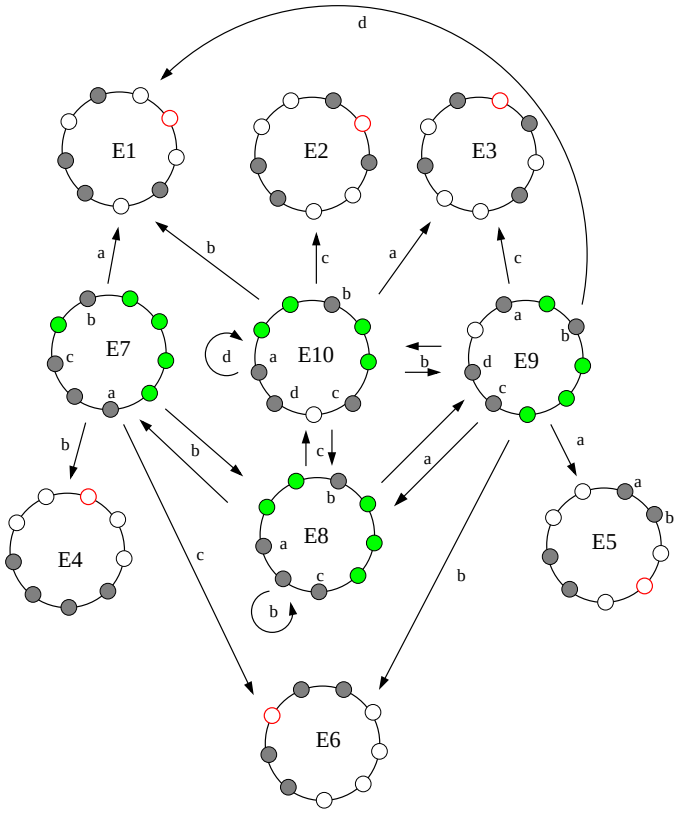


**Fig. 9.** Theorem 5. Case $(k, n) = (4, 9)$. Grey nodes are the occupied ones.

(resp., arriving from $D5$ by moving $c$, resp., starting) at $D6$, the adversary sequentially schedules $b, d, e, f$ (that do not move), then $c$ (resp., $a$), and if $c$ (resp., $a$) does not move, it moves $a$ (resp., $c$) (possibly going to $D7$, resp. to $D5$) and goes on. Arriving from $D6$ by moving $c$ (or starting) at $D5$, the adversary sequentially schedules $b, d, e, f$ (that does not move), then $a$ twice, and then $c$ (possibly going to $D6$) and goes on. It is easy to check that this does not clear the ring.

– **Case** $(k, n) = (4, 9)$**.** In that case, there are 10 distinct configurations that are depicted in Figure 9.

  By Lemma 6, if Algorithm ALG reaches Configuration $E1$ to $E6$, then the clearing fails. Therefore, in particular, any Algorithm ALG that clears the ring never reaches Configuration $E7$ or only $b$ is allowed to move towards $a$ in that configuration. Similarly, in Configuration $E9$, the possible moves are: robot $b$ moving towards $c$ and robot $a$ moving towards $d$. In Configuration $E10$, the possible moves are: robot $b$ moving towards $c$, robot $c$ moving towards $d$, and moving robot $d$. Any movement of robots $a, b, c$ are allowed in Configuration $E8$. It is easy to check that, allowing only these moves cannot clear the green vertices.

– **Case** $(k, n) = (5, 9)$**.** In that case, there are 10 distinct configurations that are depicted in Figure 10. For purpose of contradiction, let us assume that Algorithm ALG clears the ring. We consider several cases depending on the Algorithm ALG.

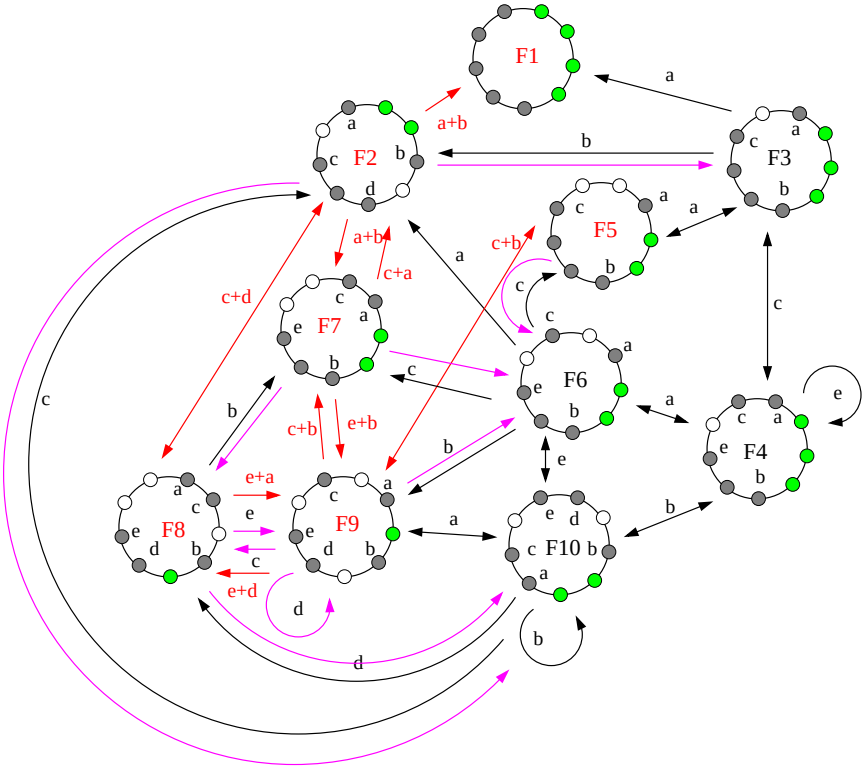


**Fig. 10.** Theorem 5. Case $(k, n) = (5, 9)$. Grey nodes are the occupied ones. In case of symmetrical configurations, there are several cases depending whether the adversary schedules one robot (pink transitions) or two symmetrical robots (red transitions).

If Algorithm Alg reaches Configuration $F1$, then only two robots can move, that is, those with an empty neighbor. In this case, the adversary can force both to move hence obtaining Configuration $F2$.

First, let us assume that, in configuration $F6$, Alg allows other move than moving robot $a$ toward $b$. In that case, the adversary schedules robots one by one. Moreover, it ensures that robot $a$ does not move in Configuration $F6$ by moving another robot first. This case is depicted in Figure 11 where the dotted move is never executed. Hence, in that case, the green vertices are never cleared (of course, the adversary ensures that, in the case of symmetries, e.g. from $F8$ to $F7$, the considered robot moves to the "bad" side). Indeed, we can check that, for any configuration, if the green vertices are initially contaminated, then after any move leading to another configuration, the new green vertices are contaminated. Therefore, such an Algorithm Alg cannot clear the ring.

By previous paragraph, in Configuration $F6$, Algorithm Alg only allows robots $a$ to move towards $b$. Moreover, Configuration $F6$ followed by Configuration $F2$ must be met infinitely often. Therefore, we can assume that $F2$ is the initial configuration and that we have to go back to it via $F6$.

Now, assume that, in Configuration $F9$, Alg allows other move than moving robot $a$ towards $c$ or moving $d$. Such an Algorithm Alg cannot clear the ring since the adversary can ensure that only Configurations $F2, F7, F8$ and $F9$ are perpetually reached, which does not allow to

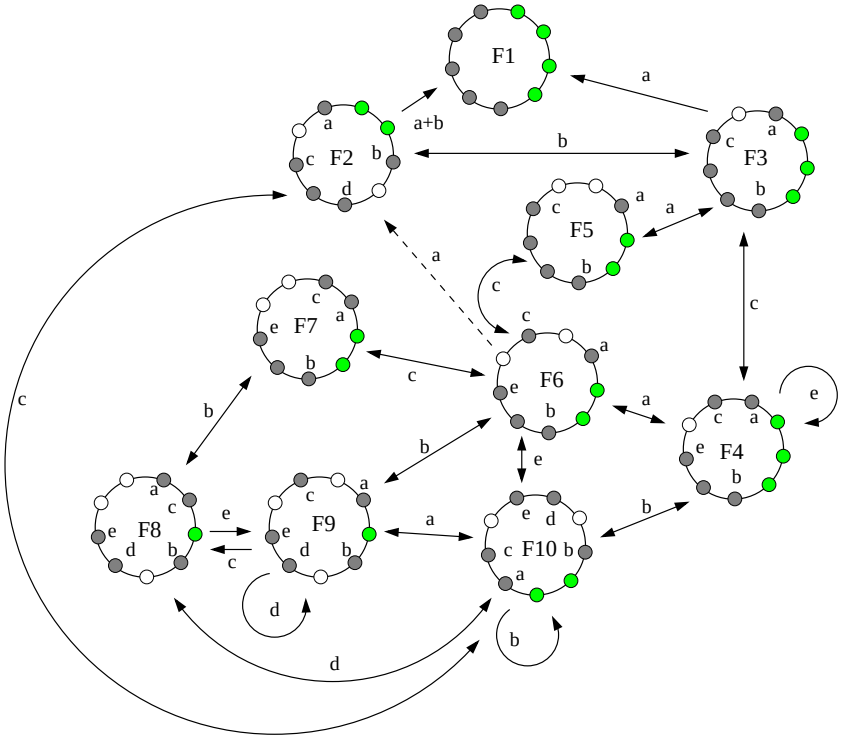


**Fig. 11.** Theorem 5. Case $(k, n) = (5, 9)$. Grey nodes are the occupied ones.

clear the green nodes. Therefore, in Configuration $F9$, Algorithm ALG must allows robots $a$ to move towards $c$ (all other robots decide not to move but $d$ that may move remaining in the same configuration).

We now show that, in Configuration $F2$, Algorithm ALG has to allow robots $c$ and $d$ to move and that these are the single possible moves. Indeed, let us assume that, in Configuration $F2$, Algorithm ALG allows robot $a$ (resp., $b$) decides to move toward $b$ (resp., towards $a$). In that case, the adversary executes the following schedule: in $F2$, it schedules robot $a$ going to Configuration $F6$, then in $F6$, it schedules all robots but $a$, that do not move, and then $a$ going back to Configuration $F2$ without having cleared the ring. Therefore, in Configuration $F2$, Algorithm ALG does not allow robot $a$ to move toward $b$ as otherwise it would lead back to $F1$. Hence, the only possible move is to allow robot $c$ (resp., $d$) to move towards $a$ (resp., towards $b$).

To conclude, we prove that, in Configuration $F10$, robot $e$ must be allowed to move.

First, assume robot $b$ is allowed to move in $F10$. Then, the adversary can schedule robots $c$ and $d$ in $F2$ but only robot $c$ moves, i.e., robot $d$ has looked and computed but not moved yet. Therefore, arriving in Configuration $F10$, the adversary can schedule robot $b$ and makes it moving simultaneously with robot $d$ which results in a collision. Hence, robot $b$ have to decide not to move in Configuration $F10$.

Second, assume for purpose of contradiction, that robot $a$ (resp., robot $d$) is allowed to move in $F10$. Then, the adversary can execute the following schedule: robot $c$ moves in $F2$ to reach $F10$, then robot $a$ moves reaching $F9$ (resp., robot $d$ moves to $F8$ and then robot $e$ moves to $F9$), then all robots but $a$ are scheduled without moving in $F9$ (see above), then robot $a$ move

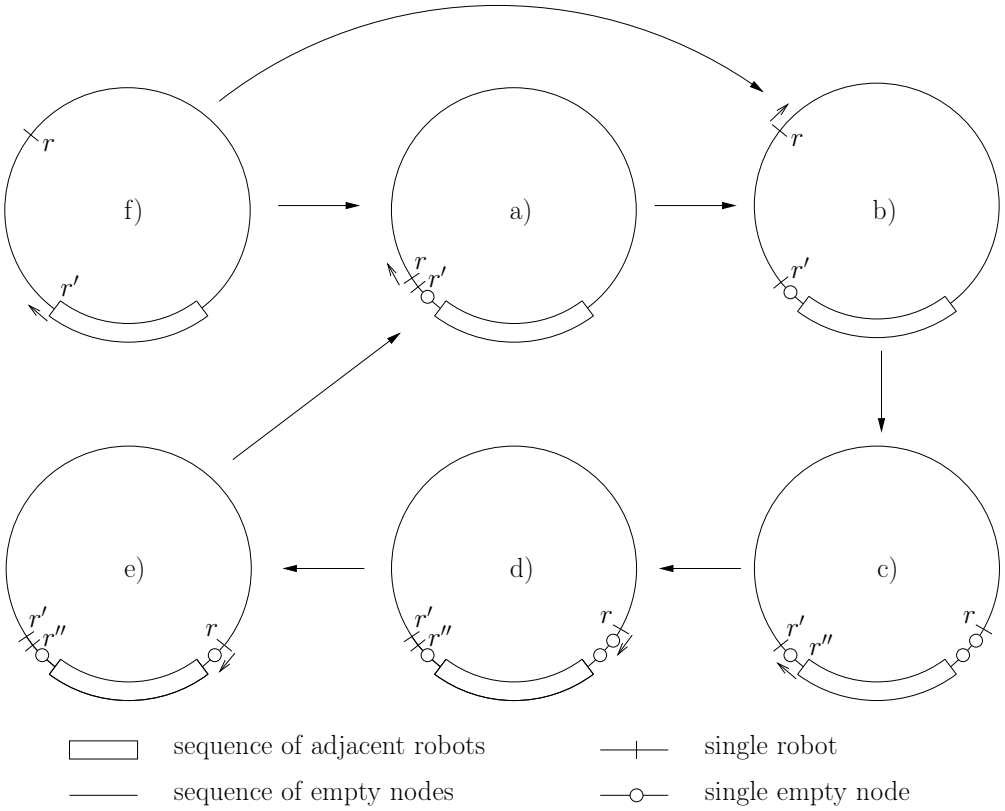


**Fig. 12.** Second phase of the Ring Clear algorithm. The arrows close to the robots indicate the robot that is moving and its direction.

to reach back $F10$ and then, the adversary can alternate Configurations $F9$ and $F10$ (resp., $F8, F9$ and $F10$) without clearing the ring.

Similarly, assume for purpose of contradiction, that robot $c$ is allowed to move in $F10$. Then, the adversary can execute the following schedule: in $F2$, robots $a, b, e$ are scheduled without moving (see above), then robot $c$ moves in $F2$ to reach $F10$, then robot $c$ moves reaching back $F2$, and then this cycle is done again with robot $d$ moving in $F2$ instead of $c$. Thus, the adversary can alternate Configurations $F2$ and $F10$ without clearing the ring.

Therefore, in Configuration $F10$, only moving $e$ and $b$ may be possible. Of course, if moving robot $e$ is not allowed, then the adversary can schedule robot $c$ in $F2$ to reach $F10$, and then, it schedules all robots, where only robot $b$ moves perpetually remaining in Configuration $F10$ which does not clear the ring.

To conclude the proof, the adversary does the following. In Configuration $F2$ (we proved above that we may assume we can start with this configuration), robot $c$ (or $d$) moves and reaches Configuration $F10$, where robot $e$ moves which reaches Configuration $F6$, where all robots but $a$ are scheduled without moving. Finally, robot $a$ is moving reaching back Configuration $F2$. It is easy to check that this does not allow to clear the ring. □

### 4.3 Algorithm Ring Clear

In this section, we give an algorithm, called Algorithm Ring Clear, to clear a ring of $n > 9$ nodes with $4 < k < n - 3$ robots (except for $n = 10$ and $k = 5$) starting from any rigid configuration.

Algorithm Ring Clear works in two phases. In the first phase, Algorithm Align is executed until one configuration in the set of configurations $\mathcal{A}$ (described below and that contains $\mathcal{C}^*$) is reached. Then, the robots execute the algorithm illustrated in Figure 12. The assumption of initial rigidity ensures that, in the entire algorithm, only one robot is allowed to move at one time. Moreover, the set of configurations in the two phases are disjoint and hence the robots can always distinguish which phase is performing.

We denote as $\mathcal{A}$ the set of the following configurations:

$\mathcal{A}$-a: Those with $k-2$ adjacent robots and two adjacent robots separated by one empty node from the first $k-2$ (Figure 12a).

$\mathcal{A}$-b: Those with $k-2$ adjacent robots, one robot separated by an empty node from the first $k-2$, and another robot not adjacent to any other one (Figure 12b).

$\mathcal{A}$-c: Those with $k-2$ adjacent robots, one robot separated by one empty node from the first $k-2$, and another robot separated by two empty nodes from the first $k-2$ on the other side of the first robot. (Figure 12c).

$\mathcal{A}$-d: Those with $k-3$ adjacent robots, two adjacent robots separated by one empty node from the first $k-3$ on one side, and another robot separated by two empty nodes from the first $k-3$ on the other side (Figure 12d).

$\mathcal{A}$-e: Those with $k-3$ adjacent robots, two adjacent robots separated by one empty node from the first $k-3$ on one side, and another robot separated by one empty node from the first $k-3$ on the other side (Figure 12e).

$\mathcal{A}$-f: Asymmetric configurations with $k-1$ adjacent robots and one single robot (Figure 12f).

Note that the configuration $\mathcal{C}^*$ belongs to the set $\mathcal{A}$-f.

The algorithm perpetually cycles among configurations $\mathcal{A}$-a — $\mathcal{A}$-e as depicted in Figure 12. The pseudo-code of Ring Clear is reported in Figure 13. First, at line 2, the algorithm performs Align until a configuration in $\mathcal{A}$ is achieved. Let us refer to $r$, $r'$, and $r''$ as the two robots identified in Figure 12. At line 4, the algorithm identifies a robot $r$ in a configuration in $\mathcal{A}$-a as the one with view $(q_0, q_1, \ldots, q_{k-1}) = (0, 1, 0, 0, \ldots, 0, q_{k-1})$, where $q_{k-1} > 2$. In this case $r$ has to move towards $q_{k-1}$ (line 10). Note that as the configuration is rigid, only $r$ can read such a configuration. Configurations in $\mathcal{A}$-b are identified at lines 5 and 11. In particular, the robot $r$ allowed to move can read the configuration in two directions, depending on the size of the its adjacent intervals. If it reads in clockwise order with respect to Figure 12, then the configuration read is $(q_0, q_1, \ldots, q_{k-1}) = (q_0, 0, 0, \ldots, 0, 1, q_{k-1})$, where $q_0 > 2$ and $q_{k-1} > 0$ (line 11) and $r$ has to move towards $q_0$ (line 15). Otherwise, the configuration read is $(q_0, q_1, \ldots, q_{k-1}) = (q_0, 1, 0, 0, \ldots, 0, q_{k-1})$, where $q_0 > 0$ and $q_{k-1} > 2$ (line 5) and $r$ has to move towards $q_{k-1}$ (line 10). Configurations in $\mathcal{A}$-c, $\mathcal{A}$-d, $\mathcal{A}$-e, and $\mathcal{A}$-f are identified similarly at lines 6, 7 and 12, 13, 8, respectively. Finally, we observe that the conditions at lines 4–13 are pairwise disjoint and, in the same configuration, only one condition is satisfied for exactly one robot.

The next theorem shows that it exclusively and perpetually clears the ring.

**Theorem 6.** *Starting from any exclusive and rigid configuration, Algorithm* Ring Clear *solves both the exclusive perpetual clearing and exploration problems using $k$ robots in any $n$-node ring, $n 9$ and $4 < k < n - 3$ (but for $n = 10$ and $k = 5$).*

**Procedure**: Ring Clear
**Input**: Rigid and exclusive configuration $\mathcal{C}$ with view $W = (q_0, q_1, \ldots, q_{k-1})$ as seen from a robot $r$

**if** $\mathcal{C} \notin \mathcal{A}$ **then**
  Apply Algorithm Align
**else**
  **if** *($q_0 = 0$, $q_1 = 1$, $q_i = 0$ $\forall i \in \{2, 3, \ldots, k-2\}$, $q_{k-1} > 2$)* // $\mathcal{A}$-a
  *OR ($q_0 > 0$, $q_{k-1} > 2$, $q_1 = 1$, $q_i = 0$ $\forall i \in \{2, 3, \ldots, k-2\}$)* // $\mathcal{A}$-b
  *OR ($q_i = 0$ $\forall i \in \{0, 1, \ldots, k-4\}$, $q_{k-3} = 2$, $q_{k-2} > 0$, $q_{k-1} = 1$)* // $\mathcal{A}$-c
  *OR ($q_0 > 0$, $q_1 = 0$, $q_2 = 1$, $q_i = 0$ $\forall i \in \{3, 4, \ldots, k-2\}$, $q_{k-1} > 2$)* // $\mathcal{A}$-d
  *OR ($q_i = 0$ $\forall i \in \{0, 1, \ldots, k-3\}$, $q_{k-2} > q_{k-1} > 0$, $q_{k-2} + q_{k-1} > 3$)* // $\mathcal{A}$-f
  **then**
    move towards $q_{k-1}$;
  **if** *($q_0 > 2$, $q_{k-1} > 0$, $q_i = 0$ $\forall i \in \{1, 2, \ldots, j-2\}$, $q_{k-2} = 1$)* // $\mathcal{A}$-b
  *OR ($q_0 = 2$, $q_i = 0$ $\forall i \in \{1, 2, \ldots, k-4\}$, $q_{k-3} = 1$, $q_{k-2} = 0$, $q_{k-1} > 0$)* // $\mathcal{A}$-d
  *OR ($q_0 = 1$, $q_i = 0$ $\forall i \in \{1, 2, \ldots, k-4\}$, $q_{k-3} = 1$, $q_{k-2} = 0$, $q_{k-1} > 1$)* // $\mathcal{A}$-e
  **then**
    move towards $q_0$;

**Fig. 13.** Algorithm Ring Clear.

*Proof.* By Theorem 1, Algorithm Align eventually achieves configuration $\mathcal{C}^* \in \mathcal{A}$-f. If the configuration is in $\mathcal{A}$-f, let us denote as $r$ the single robot and by $r'$ the robot on the border of the sequence of $k-1$ robots which is the closest to $r$. Note that, as the initial configuration is assumed to be rigid, then we can always distinguish robot $r'$. The algorithm let move $r'$ towards the only direction allowed. The obtained configuration is either $\mathcal{A}$-a or $\mathcal{A}$-b.

In the following, we show that if a configuration is in any of the configurations in $\mathcal{A}$, the algorithm perpetually cycles among them in the sequence ($\mathcal{A}$-a, $\mathcal{A}$-b, $\mathcal{A}$-c, $\mathcal{A}$-d, $\mathcal{A}$-e). Hence the algorithm never goes back to a configuration in $\mathcal{A}$-f and without loss of generality, we can assume that the first configuration is of type $\mathcal{A}$-a.

In this case, we call $S$ the sequence of $k-2$ adjacent robots and $r$ and $r'$ the robot at distance 3 and 2 from $S$, respectively. The algorithm identifies robot $r$ in a configuration in $\mathcal{A}$-a. The view read by $r$ is $(q_0, q_1, \ldots, q_{k-1}) = (0, 1, 0, 0, \ldots, 0, q_{k-1})$, where $q_{k-1} > 2$. In a configuration of type $\mathcal{A}$-a, the edges which are cleared are the internal edges of $S$ and the edge between $r$ and $r'$. The algorithm first clears the edges in the sequence of empty nodes. To this aim, it moves robot $r$ towards the direction opposite to $r'$. Note that as the configuration is rigid, only $r$ can read such a configuration.

The configuration obtained is of type $\mathcal{A}$-b where the distance between the two single robots is 2. In particular, robot $r$ can read the configuration in two directions, depending on the size of the its adjacent intervals.

In this way, $r$ cleared the edge where it passed through. The algorithm keeps on moving $r$ in the same direction until it reaches a configuration of type $\mathcal{A}$-c, in this way the configuration is still $\mathcal{A}$-b and all the edges between $r$ and $r'$ where $r$ passed through are cleared. Note that, the robots are always able to identify the correct direction thanks to the position of robot $r'$. More precisely, robot $r$ can read the configuration in two directions, depending on the size of the its adjacent intervals. If it reads in clockwise order with respect to Figure 12, then the configuration read is $(q_0, q_1, \ldots, q_{k-1}) = (q_0, 0, 0, \ldots, 0, 1, q_{k-1})$, where $q_0 > 2$ and $q_{k-1} > 0$ and $r$ has to move towards $q_0$. Otherwise, the configuration read is $(q_0, q_1, \ldots, q_{k-1}) = (q_0, 1, 0, 0, \ldots, 0, q_{k-1})$, where $q_0 > 0$ and $q_{k-1} > 2$ and $r$ has to move towards $q_{k-1}$.

When the configuration is of type $\mathcal{A}$-c, the only edges which are not cleared are the two edges between $r'$ and $S$ and the three edges between $r$ and $S$. Let $r''$ be the robot on the border of $S$ which is the closest to $r'$. If the configuration is of type $\mathcal{A}$-c, the algorithm moves robot $r''$ towards $r'$, clearing the two edges between $r'$ and $S \setminus \{r''\}$. The obtained configuration is of type $\mathcal{A}$-d, where the only edges which are not cleared are those between $r$ and $S$. Therefore, the algorithm moves $r$ towards $S$ obtaining first a configuration of type $\mathcal{A}$-e and again a configuration of type $\mathcal{A}$-a. Note that, in all the above configurations, it is always possible to distinguish robots $r$, $r'$ and $r''$. Moreover, the direction of the movements can be identified by the robots thanks to the position of robots $r$, $r'$ and $r''$ themselves. Summarizing, the algorithm perpetually cycles among configurations $\mathcal{A}$-a – $\mathcal{A}$-e. □

### 4.4 Clearing an $n$-node ring using $n-3$ robots

In this section, we propose a specific algorithm to clear any $n$-node ring, $n \geq 10$ using $n-3$ robots. Together with the previous algorithm and the impossibility results, this closes all the cases, but for $n = 10$ and $k = 5$.

In any exclusive configuration with $k = n-3$ robots, all the nodes of the rings but three are occupied. In other words, the ring is made of at most three sequences of adjacent occupied nodes. We denote by $A$, $B$ and $C$ the number of nodes in such three sequences. If two empty nodes are adjacent, the corresponding sequence between them has size 0. Note that, as the configuration is rigid, such three sequences are all different and then, we can assume w.l.o.g. that $A < B < C$. In the following, we denote a configuration as $(A, B, C)$. We call *final* configurations the three configurations: $(0, 2, k-2)$, $(0, 3, k-3)$, and $(1, 2, k-3)$. Note that, since $k = n-3 \geq 7$, the final configurations are well defined and distinguishable, that is $B$ is always strictly smaller than $C$. Our algorithm is denoted as NminusThree, its pseudo-code is formally given in Figure 14, and it works in two phases: In the first phase, it creates a final configuration and in the second one it performs the exclusive perpetual clearing.

The first phase is performed if the configuration is not final and it is accomplished by performing the following rules in the priority given by the following ordering.

**R1.1:** If $A > 0$, move towards $C$ the robot on the border of $A$ which is closer to $C$ (line 12);
**R1.2:** If $B = 1$, move towards $B$ the robot on the border of $C$ which is closer to $B$ (line 15);
**R1.3:** If $B > 3$, move towards $C$ the robot on the border of $B$ which is closer to $C$ (line 17).

Rule **R1.1** is executed for $A$ steps until $A = 0$. Afterwards, either Rule **R1.2** or Rule **R1.3** is executed. If $A = 0$ and $B = 1$, then $C = k-1$. It follows that, after one step of Rule **R1.2**, the final configuration $(0, 2, k-2)$ is achieved. If $A = 0$ and $B > 3$, then the configuration is $(0, B, k-B)$ and the final configuration $(0, 3, k-3)$ is achieved after $B-3$ steps or Rule **R1.3**. If $A = 0$ and either $B = 2$ or $B = 3$, the configuration is final. The following lemma follows.

**Lemma 7.** *The first phase of the algorithm eventually achieves a final configuration if* $n \geq 10$ *and* $k = n-3$.

*Proof.* We first prove that, any configuration obtained by applying Rules **R1.1**—**R1.3** is still rigid. Let us denote as $A'$, $B'$ and $C'$ the number of nodes in the sequences of occupied nodes obtained after one movement that corresponds to $A$, $B$ and $C$, respectively. By performing Rule **R1.1**, we obtain that $A' = A-1$, $B' = B$, and $C' = C+1$, therefore $A' < B' < C'$. If Rule **R1.2** is performed,

it follows that $A = 0$, $B = 1$ and $C = k-1$. Therefore, after performing Rule **R1.2**, we have $A' = 0$, $B' = 2$, $C' = k - 2 \geq 5$ that is, $A' < B' < C'$. By performing Rule **R1.3**, we obtain that $A' = 0$, $B' = B - 1$, $C' = C + 1$ and then $A' < B' < C'$.

To conclude the proof, it is enough to observe that if $k \geq 7$ (that is $n \geq 10$) and the configuration is not final, one of the conditions of Rules **R1.1**—**R1.3** is always true. Moreover, by the discussion which follows the rules, the phase one of the algorithm terminates in a finite number of steps in a final configuration. □

The second phase of the algorithm performs the clearing. It starts from any final configuration and performs the following rules.

**R2.1:** If $(A, B, C) = (0, 2, k-2)$, move towards $B$ the robot on the border of $C$ which is closer to $B$ (line 3);
**R2.2:** If $(A, B, C) = (0, 3, k-3)$, move towards $A$ the robot on the border of $B$ which is closer to $A$ (line 6);
**R2.3:** If $(A, B, C) = (1, 2, k-3)$, move the robot of $A$ towards $C$ (line 9).

```
   Procedure: NminusThree
   Input: Rigid and exclusive configuration C with k = n − 3 robots
 1 Let (A, B, C) be the current configuration with 0 ≤ A < B < C
                                                        // Phase 2: clearing the ring
 2 if (A, B, C) = (0, 2, k − 2) then
 3 |   Move towards B the robot of C which is closer to B                  // Rule R2.1
 4 else
 5 |   if (A, B, C) = (0, 3, k − 3) then
 6 |   |   move towards A the robot of B which is closer to A               // Rule R2.2
 7 |   else
 8 |   |   if (A, B, C) = (1, 2, k − 3) then
 9 |   |   |   Move the robot of A towards C                               // Rule R2.3
10 |   |   else
                                          // Phase 1: reaching starting configuration
11 |   |   |   if A > 0 then
12 |   |   |   |   Move towards C the robot of A which is closer to C      // Rule R1.1
13 |   |   |   else
14 |   |   |   |   if B = 1 then
15 |   |   |   |   |   Move towards B the robot of C which is closer to B  // Rule R1.2
16 |   |   |   |   else
17 |   |   |   |   |   Move towards C the robot of B which is closer to C  // Rule R1.3
```

**Fig. 14.** Algorithm NminusThree.

The next theorem states the correctness of the algorithm.

**Theorem 7.** *Starting from any exclusive and rigid configuration, Algorithm* NminusThree *solves both the exclusive perpetual exclusive graph clearing and exploration problems using $n-3$ robots in any $n$-node ring, $n \geq 10$.*

*Proof.* By the hypothesis that $n \geq 10$, it follows that $k \geq 7$, and then the final configurations are well defined and distinguishable. Moreover, by Lemma 7 the first phase of the algorithm always

achieves a final configuration. It remains to show that the second phase exclusively and perpetually performs the clearing. Note that if we perform Rule **R2.**$i$ we obtain the configuration in the condition of Rule **R2.**$((i \bmod 3) + 1)$. It follows that Rules **R2.1**—**R2.3** are performed cyclically infinitely many times. Let us assume that the second phase starts from configuration $(0, 2, k-2)$. In this configuration, the edges inside sequences $B$ and $C$ are cleared. By performing Rule **R2.1**, also the two edges adjacent to $B$ and $C$ are cleared. The non-cleared edges are the three edges between to $B$ and $C$ which pass through $A$. At this point Rules **R2.2** and **R2.3** are performed which in turn clear first the edge between $B$ and $A$ which is adjacent to $B$ and then the two remaining edges. □

## 5 Gathering in a ring

In this section, we devise a strategy to accomplish the *gathering* task on a ring under the discrete CORDA model. The problem requires the robots to reach a common node and remain in there. Hence, more than one robot must be allowed to occupy a node, i.e. a *multiplicity* occurs. We assume that the robots have the *local* multiplicity detection capability. This is necessary as proven in [26].

In accordance to the other tasks previously shown, we make use of procedure ALIGN in order to achieve configuration $\mathcal{C}^*$ starting from any (exclusive) rigid configuration on $n$-node rings with $2 < k < n-2$ robots. In fact, any configuration with $k = 2$, $k = n-1$, or $k = n-2$ robots is symmetric. Hence, the next algorithm provides a full characterization of rigid configurations where the gathering can be accomplished. Before providing the algorithm, we need some more notation.

A configuration is said to be of type $\mathcal{C}^*$ if it is composed by an ordered sequence of $j-2$ intervals of length 0, one interval of length 1 and one interval of length $n-j-1$, with $3 \leq j \leq k$. Consequently, also the nodes of the ring can be considered ordered according to the intervals' order. Hence, the first two nodes of the sequence will constitute interval $q_0 = 0$ in the current configuration. Clearly, $\mathcal{C}^*$ is a $\mathcal{C}^*$-type configuration.

Rule CONTRACTION allows to move any robot occupying the first node of a $\mathcal{C}^*$-type configuration towards the second one. Possibly, such nodes can be occupied by many robots. Whenever a robot $r$ wakes up, it executes the algorithm GATHERING given in Figure 15.

**Algorithm**: GATHERING
**Input**: Rigid and exclusive configuration $\mathcal{C}$

```
1 if C is not a C*-type configuration then
2 |   ALIGN(C);
3 else
4 |   if More than two nodes are occupied then
5 |   |   Apply CONTRACTION(C);
6 |   else
7 |   |   if r is not part of a multiplicity then
8 |   |   |   Move towards the other occupied node;
```

**Fig. 15.** Algorithm GATHERING.

From $\mathcal{C}^*$-type configurations, the algorithm simply applies CONTRACTION until only two nodes are occupied. Note that, at each intermediate step, the current configuration is always a $\mathcal{C}^*$-type,

and the algorithm allows to move the robot(s) from the first node of the current interval $q_0$ towards the second one. Eventually, the number of intervals of length 0 is reduced by one. This is repeated until only two nodes at distance 2 remain occupied. Note that, in such a configuration, $k-1$ robots are gathered on the same node and the other occupied node contains a single robot. From this configuration the robots can distinguish which is the node occupied by a single robot by using the local multiplicity detection. Therefore, only the single robot is allowed to move towards the other occupied node until joining it, while robots composing the multiplicity do not move. We can now state the next theorem.

**Theorem 8.** *There exists an algorithm performing the gathering of $k > 2$ robots on rings of $n > k+2$ nodes when the initial configuration is exclusive and rigid, and the robots are empowered with the local multiplicity detection.*

## 6 Conclusion

In this work, we provided a unified strategy for solving three tasks in the discrete CORDA model on ring topologies when the initial configuration is rigid. Namely we solved the exclusive perpetual clear, the exclusive perpetual exploration and the gathering with local multiplicity detection capability. Moreover, the given algorithms solve some open problems and the impossibility results provided for the exclusive perpetual clearing problem fully characterize any initial rigid configuration. Our work opens two main research direction: use the ALIGN algorithm to solve other problems in rigid configurations and devise similar algorithms to handle symmetric or periodic configurations.

## Bibliography


[1] C. Ambühl, L. Gąsieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. *ACM Trans. Algorithms*, 7(2):17:1–17:21, 2011.

[2] R. Baldoni, F. Bonnet, A. Milani, and M. Raynal. Anonymous graph exploration without collision by mobile robots. *Inf. Process. Lett.*, 109(2):98–103, 2008.

[3] R. Baldoni, F. Bonnet, A. Milani, and M. Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In *12th Int. Conf. On Principles Of Distributed Systems (OPODIS)*, volume 5401 of *Lecture Notes in Computer Science*, pages 428–445. Springer-Verlag, 2008.

[4] E. Bampas, J. Czyzowicz, L. Gąsieniec, D. Ilcinkas, and A. Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *24th Int. Symp. on Distributed Computing (DISC)*, volume 6343 of *Lecture Notes in Computer Science*, pages 297–311, 2010.

[5] D. Bienstock and P. D. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12(2):239–245, 1991.

[6] L. Blin, J. Burman, and N. Nisse. Distributed exclusive and perpetual tree searching. In *26th Int. Symp. on Distributed Computing (DISC)*, volume 7611 of *Lecture Notes in Computer Science*, pages 403–404, 2012.

[7] L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *24th Int. Symp. on Distributed Computing (DISC)*, volume 6343 of *Lecture Notes in Computer Science*, pages 312–327, 2010.

[8] F. Bonnet, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Asynchronous exclusive perpetual grid exploration without sense of direction. In *15th Int. Conf. On Principles Of Distributed Systems (OPODIS)*, volume 7109 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2011.

[9] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. *ACM Trans. Algorithms*, 4(4):42:1–42:18, Aug. 2008.

[10] J. Czyzowicz, L. Gąsieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6–7):481 – 499, 2009.

[11] G. D'Angelo, G. Di Stefano, R. Klasing, and A. Navarra. Gathering of robots on anonymous grids without multiplicity detection. In *19th Int. Coll. on Structural Information and Communication Complexity (SIROCCO)*, volume 7355 of *Lecture Notes in Computer Science*, pages 327–338, 2012.

[12] G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering of six robots on anonymous symmetric rings. In *18th Int. Coll. on Structural Inf. and Com. Complexity (SIROCCO)*, volume 6796 of *Lecture Notes in Computer Science*, pages 174–185, 2011.

[13] G. D'Angelo, G. Di Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. In *26th Int. Symp. on Distributed Computing (DISC)*, volume 7611 of *Lecture Notes in Computer Science*, pages 330–344, 2012.

[14] G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model. In S. Alpern, R. Fokkink, L. Gąsieniec, R. Lindelauf, and V. Subrahmanian, editors, *Search Theory: A Game Theoretic Perspective*, pages 197–222. Springer, 2013.

[15] G. D'Angelo, G. Di Stefano, A. Navarra, N. Nisse, and K. Suchan. A unified approach for different tasks on rings in robot-based computing systems. In *15th IEEE IPDPS Workshop on Advances in Parallel and Distributed Computing Models (APDCM)*, 2013. to appear.

[16] S. Devismes, F. Petit, and S. Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *16th Int. Coll. on Structural Information and Communication Complexity (SIROCCO)*, volume 5869 of *Lecture Notes in Computer Science*, pages 195–208, 2009.

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

[18] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65:562–583, 2013.

[19] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by oblivious mobile robots.* Morgan and Claypool, 2012.

[20] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.

[21] D. Ilcinkas, N. Nisse, and D. Soguet. The cost of monotonicity in distributed graph searching. *Distributed Computing*, 22(2):117–127, 2009.

[22] T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *17th Int. Coll. on Structural Information and Communication Complexity (SIROCCO)*, volume 6058 of *Lecture Notes in Computer Science*, pages 101–113, 2010.

[23] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations. In *18th Int. Coll. on Structural Information and Communication*

*Complexity (SIROCCO)*, volume 6796 of *Lecture Notes in Computer Science*, pages 150–161, 2011.

[24] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *37th Int. Symp. on Math. Foundations of Computer Science (MFCS)*, volume 7464 of *Lecture Notes in Computer Science*, pages 542–553, 2012.

[25] R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci.*, 411:3235–3246, 2010.

[26] R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390:27–39, 2008.

[27] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012.

[28] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theor. Comput. Sci.*, 384:222–231, 2007.