A unified approach for different tasks on rings in robot-based computing systems

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

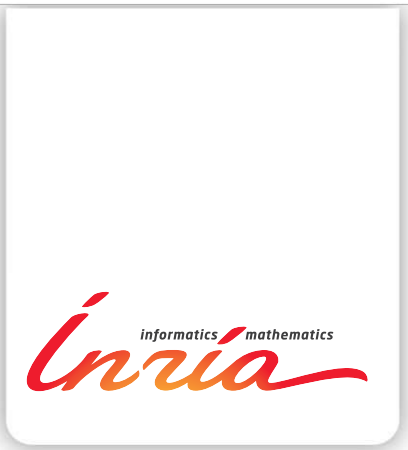
A unified approach for different tasks on rings in robot-based computing systems

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

RESEARCH REPORT
N° 8013

Project-Teams Mascotte

ISSN 0249-6399 ISRN INRIA/RR--8013--FR+ENG



A unified approach for different tasks on rings in robot-based computing systems

Gianlorenzo D'Angelo*, Gabriele Di Stefano†, Alfredo Navarra‡, Nicolas Nisse*, Karol Suchan§

Project-Teams Mascotte

Research Report n° 8013 — — 36 pages

* MASCOTTE Project, INRIA/I3S(CNRS/UNSA), France.
† Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Univ. degli Studi dell'Aquila, Italy.
‡ Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy.
§ Facultad de Ingenierìa y Ciencias, Universidad Adolfo Ibàñez, Chile

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

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. We present a unified approach to solve three important problems in the field: the *exclusive perpetual exploration*, the *exclusive perpetual graph searching* and the *gathering* problems. In the first problem, each robot aims at visiting each node infinitely often; in perpetual graph searching, the team of robots aims at clearing all edges of the network infinitely often; and in the gathering problem, all robots must eventually occupy the same node.

We investigate these tasks in the famous *CORDA* distributed computing model where the robots cannot communicate but can perceive the positions of other robots. More precisely, each robot is equipped with visibility sensors and motion actuators, and it operates in *Look-Compute-Move* 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*, *oblivious*, *uniform* (execute the same algorithm) and have *no common sense of orientation*. For the first two problems, the *exclusivity constraint* must also be satisfied, i.e., a node can be occupied by at most one robot. Finally, the robots have the *local multiplicity detection capability* which is required to solve the gathering problem.

In this setting, we devise algorithms that, starting from any exclusive rigid (i.e. aperiodic and asymmetric) configuration, solve the three above mentioned problems in anonymous ring-topologies. Our main algorithms consist of two phases. The first phase is common to all problems and allows $k > 2$ robots to achieve a particular configuration in an n-node ring, $k < n - 2$. The second phase consists of two distinct sub-procedures: one works for $5 \leq k < n - 3$ robots, $n \geq 10$ (but for $k = 5$ and $n = 10$), and solves both the exploration and graph searching problems and the second one solves the gathering problem for any n and $2 < k < n - 2$. For exploration and graph searching we also give a specific algorithm for the case of $k = n - 3$. Besides being a unified approach for three different tasks, the given algorithms solve some open problems. Moreover, we provide some impossibility results for the perpetual graph searching problem, showing that it is impossible to solve it with $k < n$ robots if $n \leq 9$ or $k \in \{1, 2, 3, n - 2, n - 1\}$. All together, we obtain an almost full characterization of exclusive perpetual graph searching in rings, leaving only open the cases $(k = 4, n > 9)$ and $(k = 5, n = 10)$.

Key-words: distributed computing, CORDA model, graph searching, gathering, graph exploration

Approche unifiée pour réaliser diverses tâches dans un anneau grâce à des robots

Résumé : Des robots autonomes doivent collaborer pour accomplir une tâche dans un réseau en anneau dont ni les nœuds ni les liens sont étiquetés. Nous présentons une approche unifiée pour résoudre trois importants problèmes du domaine: l'exploration perpétuelle, le netoyage perpétuel, et le rassemblement. Dans le premier problème, chaque sommet doit être visiter infiniment souvent par chaque robot; dans le deuxième, une équipe de robots doit nettoyer les arêtes d'un graphe infiniment souvent; et dans le dernier problème, tous les robots doivent occuper le même nœud simultanément.

Nous étudions ces trois problèmes dans le modèle de calcul distribué CORDA. Dans ce modèle, les robots ne peuvent communiquer qu'en observant les positions des autres robots. Plus précisemment, chaque robot est équipé de capteurs visuels et peut se déplacer d'un nœud à l'autre. Un robot agit par cycle *Voir-Calculer-Se déplacer*. Lors d'un cycle, le robot reçoit un instantané de la configuration courante (Voir), puis il prend la décision de se déplacer sur un nœud voisin ou pas, selon la configuration (Calculer) ; enfin il effectue l'action qu'il a calculée (Déplacer). Par ailleurs, les robots ont des capacités très restreintes: ils sont anonymes, sans mémoire, uniforme (ils exécutent le même algorithme) et n'ont pas de sens commun de l'orientation. Concernant les deux problèmes, la contrainte d'exclusion doit également être satisfaite : un nœud ne peut être occupé que par un robot.

Dans ce contexte, nous proposons des algorithmes qui, partant de n'importe quelle configuration asymmétrique et apériodique, résolvent les problèmes mentionnés ci-dessus dans un anneau anonyme. Notre principal algorithme est composé de deux phases. La première est commune à tous les problèmes et permet à $k \geq 2$ robots d'atteindre une configuration particulière dans un anneau de n sommets, $k < n - 2$. La seconde phase consiste en deux sous-procédures distinctes: l'une pour résoudre le problème du rassemblement, et la seconde qui permet à $5 \leq k < n - 3$ robots, $n \geq 10$ (sauf pour$k = 5$ et $n = 10$), de résoudre à la fois le problème de l'exploration et du nettoyage perpétuel. En plus d'être une approche unifiée, nos algorithmes résolvent quelques problèmes ouverts. De plus, nous prouvons des résultats d'impossibilité dans le cas du nettoyage perpétuel d'un anneau. En résumé, nous caractérisons presque complètement le problème du nettoyage perpétuel d'un anneau.

Mots-clés : calcul distribué, modèle CORDA

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. Various problems have been studied in this setting such as *pattern formation* (in Euclidean metric spaces), *graph exploration with stop*, *exclusive perpetual exploration*, *exclusive perpetual graph searching* and *gathering*. Recently, several algorithms have been proposed to solve these problems in particular topologies such as lines, rings, trees and grids. Here, we propose a unified approach to solve the last three problems in rings.

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. Moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges. This model is referred in the literature also as the *CORDA* model [13, 21]. We consider the minimalist variant of the CORDA 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*, *uniform* (i.e. they all execute the same algorithm), *oblivious* (memoryless) and have no common sense of orientation. 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 [1].

The 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. In the following, we review the literature concerning on graph topologies focussing on rings.

Related work. In the problem of graph exploration with stop [10, 11, 12], 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 [1, 2, 5, 6] requires that each robot visits each node of the graph infinitely many times. Moreover, it adds the exclusivity constraint. In [5], 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 searching has been widely studied in centralized [14] and distributed setting (e.g., [15]). 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 clear edge can be recontaminated if there is a path without robots from a contaminated edge to it. The study of graph searching in the CORDA model when the exclusivity property must be always satisfied is introduced in [4] where a characterization of the perpetual exclusive graph searching on tree topologies is given. As far as we know, no results have been proposed in ring topologies for the perpetual exclusive graph searching problem in the CORDA model.

The gathering problem consists in moving all the robots in the same node and remain there. In [7], a full characterization of the gathering on grid topologies 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 [20], 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 [20], some impossibility results have been proven. Then, several algorithms have been proposed for different kinds of exclusive initial configurations in [8, 19, 20]. These papers left open some cases which have been closed in [9] 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 [16]. In [17], the case where k is odd and strictly smaller than $n-3$ has been solved. In [18], the authors provide an algorithm for the case where n is odd, k is even, and $10 \leq k \leq n-5$. The remaining cases with local multiplicity detection are left open and a the design of a unified algorithm is still not known.

Contribution. In this work, we provide a unified approach for solving different tasks in the minimalist-CORDA model on ring topologies. Namely, we present an algorithm that, starting from any rigid configuration, solves: the exclusive perpetual exploration, the exclusive perpetual graph searching, and the gathering with local multiplicity detection capability. Our main 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. On the one hand, we design an algorithm that, starting from configuration $\mathcal{C}^*$, solves the gathering 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). On the other hand, we present an algorithm that, starting from configuration $\mathcal{C}^*$, solves both the perpetual exclusive graph exploration and the perpetual exclusive graph searching problems, for any team of k robots in n-node rings, $n \geq 10$, $5 \leq k < n-3$ (but for $k = 5$ and $n = 10$). Moreover, we design a specific algorithm that, starting from any rigid configuration, solves the perpetual exclusive graph searching problem using $n-3$ robots in any n-node ring, $n \geq 10$. Finally, we provide some impossibility results for the perpetual exclusive graph searching problem, showing that for $2 < n \leq 9$ 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 graph searching in rings, leaving only open the cases $(k = 4, n > 9)$ and $(k = 5, n = 10)$.

Outline. In the next section we define the notation used in the paper and describe the CORDA model. In Section 3, we propose an algorithm to achieve the special configuration $\mathcal{C}^*$. Perpetual exclusive graph searching is formally defined and studied in Section 4. We note that the algorithms given in this section also solve the perpetual exclusive exploration problem. The gathering problem is considered in Section 5.

2 Model and Notations

We consider a team of $k \geq 1$ robots spread in an n-node ring, $n \geq 3$. The ring is *anonymous*, that is its nodes and edges are undistinguishable. Moreover, 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 the remaining ≥ 2 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* at 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. Note that, if $\mathcal{C}$ is exclusive, then $j = k - 1$ and $\sum_{0 \leq i < j} q_j = n - k$. Note also that, 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.

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 $\mathcal{C}$. For instance, $|I_{\mathcal{C}^*}| = 1$.

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*.

We now give some useful properties that are proved in [9]. In particular, Lemma 1 is used to detect possible symmetry or periodicity of a configuration.

Property 1 ([9]) *Given a view W of a configuration $\mathcal{C}$, (i) there exists $0 < i \leq j$ such that $W = W_i$ iff $\mathcal{C}$ is periodic; (ii) there exists $0 \leq i \leq j$ such that $W = \overline{W_i}$ iff $\mathcal{C}$ is symmetric; (iii) $\mathcal{C}$ 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 ([9]) *Given a configuration C, (i) $|I_{\mathcal{C}}| = 1$ if and only if $\mathcal{C}$ is either rigid or it admits only one axis of symmetry passing through the supermin; (ii) $|I_{\mathcal{C}}| = 2$ if and only if $\mathcal{C}$ is either aperiodic and symmetric with the axis not passing through any supermin or it is periodic with period $\frac{n}{2}$; (iii) $|I_{\mathcal{C}}| > 2$ if and only if $\mathcal{C}$ is periodic, with period at most $\frac{n}{3}$.*

2.1 min-CORDA model

We consider the CORDA model [13, 21] where the robots have no explicit way of communicate to each other (e.g., they cannot exchange messages). However, they are are endowed with visibility sensors allowing each robot to perceive their own position in the graph and the positions of all the other robots.

In the CORDA model, 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.

Following [4], we consider the *min-CORDA model*, where the robots have very weak abilities. Robots are anonymous, i.e., they do not have identifiers, and are *uniform*, i.e., they all run the same algorithm. Robots are *oblivious* (memoryless). The robots 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 *weak 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 [20].

Our goal is to investigate the feasibility of several collaborative tasks with these weak hypothesis. Throughout the paper, we assume that the starting configuration is rigid and it respects the exclusivity property. Note however that our impossibility results hold for any starting configuration.

3 Reaching configuration $\mathcal{C}^*$ in the min-CORDA model

In this section, we propose an algorithm, called ALIGN, in the min-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 graph searching, exploration, or gathering problems. We first describe the algorithm allowing to reach configuration $\mathcal{C}^*$. In the second subsection, we prove its correctness.

3.1 Algorithm ALIGN

The assumption of initial rigidity and exclusivity ensures that one single robot moves at time. The moves performed aim to reduce the unique supermin of a rigid configuration in a way that the obtained configuration is again rigid and exclusive. The algorithm is performed 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. Therefore, the snapshots provided to the robot allow 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. Four rules, called REDUCTION, are defined below where, for each rule, a single robot is asked to move to an empty node. REDUCTION$_0$ is executed only if the supermin has length at least one. If the supermin has null length, REDUCTION$_1$ is executed if the corresponding move does not create any symmetry. Otherwise, REDUCTION$_2$ is executed if it does not create any symmetry, and REDUCTION$_{-1}$ 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 perform two times REDUCTION$_1$ and show that this always 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$_i$, $i \in \{0, 1, 2\}$ strictly decreases the supermin. Finally, from some configuration $\mathcal{C}$, applying REDUCTION$_{-1}$ 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 to $\mathcal{C}^*$.

We now formally define the four rules mentioned above. Let $\mathcal{C}$ be any exclusive rigid configuration and let $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$ be its unique supermin configuration view. Let ℓ_1 be the smallest integer such that $q_{\ell_1} > 0$ and let ℓ_2 be the second smallest integer such that $q_{\ell_2} > 0$. That is, if $0 < \ell_1$ and $\ell_1 + 1 < \ell_2$, $W^{\mathcal{C}}_{min} = (0, \ldots, 0, q_{\ell_1}, 0, \ldots, 0, q_{\ell_2}, q_{\ell_2+1}, \ldots, q_{k-1})$. Let a, b, c and d be the nodes between the intervals q_0 and q_{k-1}, q_{ℓ_1} and q_{ℓ_1+1}, q_{ℓ_2} and q_{ℓ_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 given in Fig. 1 and it is performed by a generic robot r. It makes use of procedure REDUCTION$_i$ whose pseudo-code is given in Fig. 2 and described below.

Algorithm: ALIGN
Input: Rigid and exclusive configuration $\mathcal{C}$ with view $W = (q_0, q_1, \ldots, q_{k-1})$ 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);
```

Figure 1: Algorithm ALIGN.

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 Fig. 2.

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

```
1  if i = −1 then
2      if W_1 = W^C_min then
3          move towards q_0;
4      else
5          if the reverse of W_{j−1} = (W^C_min) then
6              move towards q_j;
7  if i = 0 then
8      if W = W^C_min then
9          move towards q_0;
10 if i ∈ {1, 2} then
11     if for some m, the reverse of C_m = W^C_min and m = ℓ_i then
12         move towards q_0;
13     else
14         if for some m, W_m = W^C_min and j − m = ℓ_i then
15             move towards q_j;
```

Figure 2: Procedure REDUCTION.

Let $W = (q_0, q_1, \ldots, q_k)$ be the view of $\mathcal{C}$ read by the robot r which performs the procedure and let $W^{\mathcal{C}}_{min} = (q'_0, q'_1, \ldots, q'_k)$. At lines 1–6, the algorithm moves the last robot in a supermin configuration view, that is it performs REDUCTION$_{-1}$. If $(W^{\mathcal{C}}_{min})_{j-1} \geq (W^{\mathcal{C}}_{min})_j$, then the robot has to move if and only if $W_1 = W^{\mathcal{C}}_{min}$ (line 2), that is, $q_0 = q'_j$, $q_1 = q'_0$, $\ldots$, $q_j = q'_{j-1}$, and it has to move towards q_0 (line 3) in order to reduce q'_j by enlarging q'_{j-1}. If $(W^{\mathcal{C}}_{min})_{j-1} \leq (W^{\mathcal{C}}_{min})_j$, then the robot has to move if and only if the reverse of $W_{j-1} = W^{\mathcal{C}}_{min}$ (line 5) and it has to move towards q_j (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 which view is the supermin one. At lines 10–15 the algorithm performs REDUCTION$_i$ for $i \in \{1, 2\}$. 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 $q'_0 = q_m$, $q'_1 = q_{m-1}$, $\ldots$, $q'_{\ell_i} = q_0$, that is if and only if the reverse of $C_m = W^{\mathcal{C}}_{min}$ 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 $q'_0 = q_m$, $q'_1 = q_{m+1}$, $\ldots$, $q'_{\ell_i} = q_j$, that is if and only if $W_m = W^{\mathcal{C}}_{min}$ and $j - m = \ell_i$ (line 14). In this case, such robot has to move towards q_j (line 15).

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 direction (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.

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 have to 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:

Lemma 2 ([9]) *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. Therefore REDUCTION$_1$, REDUCTION$_2$ or REDUCTION$_{-1}$ are applied 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 3 *Let $\mathcal{C}$ be a rigid exclusive configuration with supermin configuration view $W^{\mathcal{C}}_{min} = (q_0, q_1, \ldots, q_{k-1})$, $q_0 = 0$ and $\ell_1 > 0$ be the smallest integer such that $q_{\ell_1} > 0$. Then, the configuration $\mathcal{C}'$ resulting from the application of* REDUCTION$_1$ *is aperiodic. Moreover, $\mathcal{C}'$ is symmetric if and only if conditions 1–4 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, there must be $j > 0$ 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, because $q'_{\ell_1+1} > 0$, $j > \ell_1 + 1$. Hence, in that case, the view $(q_j, \ldots, q_{k-1}, q_0, \ldots, q_{j-1})$ would be 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 $W = (0, \ldots, 0, q_{\ell_1+1} + 1, q_{\ell_1+2}, q_{\ell_1+3}, \ldots q_{k-2}, q_{\ell_1+1} + 1)$ 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 ℓ_1. Let us assume that $\mathcal{C}'$ is symmetric.

For the sake of contradiction, let us assume that $q_{\ell_1} > 1$ Then, because $q_{\ell_1} \leq q_{k-1}$ 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 ℓ_1 elements of W are 0 and, by the same arguments 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 $q_{\ell_1+1} + 1 = q_j$ and that the sequence $q_{\ell_1+2}, q_{\ell_1+3}, \ldots, q_{j-1}$ 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, if applying REDUCTION$_2$ does not create any symmetry, it is applied. Lemma 4 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 4 *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 $3 \leq k < n - 2$, $q_0 = 0$, ℓ_1 be the smallest integer such that $q_{\ell_1} > 0$ 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 in the middle of the sequence $1, \{0^{\{\ell_1-1\}}, 1\}^*$ in the first case, and 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_{ℓ_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 $\ell_1 + 2 = k - 2$, and 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 $q_{\ell_1} \neq q_{k-1}$. It follows that, if there exists an axis of symmetry in $\mathcal{C}'$, it does not pass through the initial sequence of 0s.

Therefore, W contains a subsequence $(q'_j, \ldots, q'_{j+\ell_1}) = (1, 0^{\{\ell_1\}})$ where the sequence of ℓ_1 zeros is disjoint from the initial sequence of zeros, 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 $(q_0, \ldots, q_{\ell_1-1})$, which would constitute another supermin, smaller or equal than 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 ℓ_1 must equal 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 used before, $\mathcal{C}'$ 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 us denote W as $W = (q'_0, q'_1, \ldots, q'_{k-1}) \in (0^{\{\ell_1\}}, 1, 0, S', 1)$, 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 ℓ_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:

  - the move performed by 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.

– the move performed by 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_{ℓ_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 $W^{\mathcal{C}}_{min}$, $W^{\mathcal{C}}_{min} \in (0^{\{\ell_1\}}, 1, \{0^{\{\ell_1-1\}}, 1\}^+, 0^{\{\ell_1-2\}}, 1)$.

– the move performed by 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_{ℓ_2} nor q_{ℓ_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$_{-1}$, the resulting configuration being rigid.

Lemma 5 *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 Algorithm ALIGN then follows from next theorem.

Theorem 1 *Let $k \geq 3$ robots standing in an n-node ring with $k < n - 2$. Starting from a rigid exclusive configuration, Algorithm* ALIGN *eventually terminates achieving configuration $\mathcal{C}^*$ and all intermediate configurations obtained are 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.

Let us first assume that the initial configuration is different from $\mathcal{C}^s$.

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 satisfies the desired properties by Lemma 2.

If $q_0 = 0$, a unique robot executes REDUCTION$_1$ if the resulting configuration is rigid and exclusive. Otherwise, by Lemma 3, $W^{\mathcal{C}}_{min}$ satisfies Conditions 1–4. In that case, a unique robot executes REDUCTION$_2$ if the resulting configuration is rigid and exclusive. Otherwise, by Lemma 4, $W^{\mathcal{C}}_{min} \in (0, 1, 1^+, 2)$ 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 5, as the initial configuration is different from $\mathcal{C}^s$, this results in a configuration satisfying the desired properties.

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 the 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 at line 17 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 a ring in the min-CORDA model

In this section, we study the exclusive perpetual graph searching problem of an n-node ring ($n \geq 3$) by a team of $1 \leq k \leq n$ robots in the min-CORDA model, starting from any rigid exclusive configuration. In the case, $5 \leq k < n-3$ and $n \geq 10$ (or $n > 10$ if $k = 5$), we propose an algorithm that makes use of Algorithm Align presented in previous section. We then propose a specific algorithm for the case $k = n-3$ and $n \geq 10$. On the other hand, we show that for $k \in \{1, 2, 3, n-2, n-1\}$ and $n > 3$, or for $2 < n \leq 9$ and $k < n$, there is no algorithm that solves the problem, even if the initial configuration is given. The cases $k = 4$ and $(k = 5, n = 10)$ are left as open problems.

4.1 Perpetual exclusive graph searching

Given a n-node graph G where all edges are *contaminated*, the graph searching 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* by a robot when it traverses it or if both its ends are simultaneously occupied by some robots. However, a clear edge is instantaneously *recontaminated* if there is a path from one of its end to the end of a contaminated edge and no node of this path is occupied by some robot. This variant of graph searching is classically referred as *mixed graph searching* [3]. Motivated by physical constraints and following [4], we moreover impose the *exclusivity constraint*, i.e., a node can be occupied by at most one robot.

A *search strategy* using $1 \leq k \leq n$ robots consists of 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, it is obvious that there is no search 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 searching problem in n-node rings in the min-CORDA model. More precisely, we aim at designing algorithms that allow robots to clear a n-node ring starting from any rigid exclusive configuration. Because our algorithms ensure that all met configurations are rigid and exclusive, and because the robots are oblivious and cannot know whether an edge is cleared or not, the resulting strategies clear the ring *perpetually*, i.e., each edge is cleared infinitely often. Moreover, we study the exclusive perpetual exploration. Perpetual graph searching and perpetual exploration look similar but are not equivalent. For instance, one robot always moving clockwise will perpetually explore a ring without clearing it. On the other hand, the above search 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 perpetually clear the rings.

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 min-CORDA model allows to clear an n-node graph using k robots. We start with a simple result.

Lemma 6 *For any $n > 2$ and for any exclusive configuration $\mathcal{C}$, there is no algorithm that solves the exclusive perpetual graph searching 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 unoccupied node. Since these two robots have the same view of the network, whatever be the algorithm in the min-CORDA model, they take the same decision. Either they never move and the ring cannot be cleared, or both decide to move to their unoccupied neighbor. In the latter case, their moves can be scheduled (due to the asynchronicity) such that they collide. ■

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 are occupying two diametral nodes.

We show that any algorithm for perpetual searching with two robots needs to reach a configuration where the two robots occupy two diametral nodes. Then, we show that when the two robots reach 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 searching problem in a n-node ring using $k \leq 2$ robots starting from $\mathcal{C}$.*

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

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 clear, 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 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.

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* actions, 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 pendent move. 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 of them must move when they look such a configuration.

- Assume first that n is even. Then, since there are no pendent move, the adversarial scheduler can synchronize the two robots such that after their respective moves the configuration has not changed and there still are no pendant 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 searching the ring can make use of them. However, the adversarial scheduler will use them. Hence, let us call the three robots by A, B and C. At any step s, we denote by $dist_s(X, Y)$ the distance (i.e., the minimum number of consecutive edges) between the nodes occupied by robots X and Y at this step (if no ambiguity, the subscript will be omitted).

Let $\mathcal{C}_c$ be the configuration where the 3 robots occupy three consecutive nodes. Given any algorithm $\mathcal{A}$ for perpetually clearing a ring with 3 robots, we say that a configuration $\mathcal{C}$ is *bad* if, in this configuration, $dist(A, B) \leq dist(B, C)$ and there exists a robot such that, if this robot executes $\mathcal{A}$ in configuration $\mathcal{C}$, then the configuration reached after its move is such that $dist(A, B) > dist(B, C)$.

In what follows, we show that any algorithm for perpetually clearing a ring with 3 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; and (2) this reaches a configuration such that $dist(A, B) \leq dist(B, C)$ and B has not met C in the meantime; and (3) if the new configuration is not $\mathcal{C}_c$, then from this new configuration, $\mathcal{A}$ will reach another bad configuration before B meets C.

Since any algorithm for perpetually clear the ring must ensure that B meets infinitely often C, 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 searching 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 $\mathcal{A}$ that perpetually clears the ring with 3 robots. Let us consider the periodic infinite sequence $\mathcal{S}$ of moves of the robots following $\mathcal{A}$, subject to a scheduler that alternate the robots, i.e., first A makes its *Look-Compute-Move* actions, then B, then C and so on. The goal of considering such a scheduler is to be able to analyze the behavior of $\mathcal{A}$ in some particular configurations (without pending moves). Then, taking use of a more clever scheduler, $\mathcal{A}$ can be faked. 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 unoccupied neighbor.

Then, let us consider the following adversarial scheduler against $\mathcal{A}$. First, it schedules the robots alternatively until they reach configuration $\mathcal{C}_c$. W.l.o.g., say B is in the middle. From this step, the adversarial schedules twice A, then B and then twice C. That is, A moves to its unoccupied neighbor, then comes back; then B cannot move, and finally C goes and back. Clearly, $\mathcal{A}$ cannot 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 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 A and B are occupying adjacent nodes and then a further step s' when B and C are occupying adjacent nodes, such that C and A 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 $x = dist_{s_0}(A,B) \leq dist_{s_0}(B,C) = y$ and after the next step, $dist_{s_0+1}(A,B) > dist_{s_0+1}(B,C)$. 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 $\mathcal{A}$ behaves in configuration $\mathcal{C}_0$, i.e., what are the moves that can be done by robots A, B and C in configuration $\mathcal{C}_0$ when executing $\mathcal{A}$.

- 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 $\mathcal{A}$ in configuration $\mathcal{C}_0$.

  Moreover, if B decides to move in configuration $\mathcal{C}_0$, we make it moving towards A. This is possible since $\mathcal{C}_0$ is symmetric from the point of view of B. Similarly, A and C must move symmetrically: either both of them go towards B, or they move to their neighbor on the path between A and C not containing B.

  There cannot be a collision since otherwise $\mathcal{A}$ would not be a valid algorithm. Therefore, after each robot has moved, the reached configuration $\mathcal{C}_1$ is such that $dist(A,B) \leq dist(B,C)$ and B has not met C.

  There are two cases, depending on $\mathcal{C}_1$.

  - Either $\mathcal{C}_1$ is $\mathcal{C}_c$ and then Case 1. ensures that $\mathcal{A}$ cannot 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 B mets C. Note that, possibly Configuration $\mathcal{C}_1$ is a bad configuration. Again, we process as in Case 2. Therefore, we can avoid forever that B mets C, which contradicts the correctness of $\mathcal{A}$.

- Second, assume that $x < y$. Note that, since only one robot moves during step $s_0 + 1$ and $dist_{s_0+1}(A,B) > dist_{s_0+1}(B,C)$, this implies that the robot that moves during this step is B. Therefore, $y = x + 1$ and the configuration $\mathcal{C}'_0$ reached after step $s_0 + 1$ is symmetric with $\mathcal{C}_0$. In particular, B cannot distinguish $\mathcal{C}_0$ and $\mathcal{C}'_0$. Note that, Robot A must move in configuration $\mathcal{C}'_0$, and Robot C must move in configuration $\mathcal{C}_0$. Indeed, otherwise, scheduling alternatively Robot B, then Robot A, then Robot B, then Robot C, and so on, Algorithm $\mathcal{A}$ would oscillate between configurations $\mathcal{C}_0$ and $\mathcal{C}'_0$ which cannot clear the ring since $n > 4$. Moreover, in the configuration $\mathcal{C}'_0$, robot A acts in the same way as robot C in the configuration $\mathcal{C}_0$ (by symmetry).

  There are three cases to be considered.

- First assume that $x = 0$ and, when executing Algorithm $\mathcal{A}$ in configuration $\mathcal{C}_0$, robot C goes to the neighbor of B. In that case, configuration $\mathcal{C}_c$ is reached then Case 1 ensures that $\mathcal{A}$ cannot perpetually clear the ring, a contradiction.
- Second assume that $x > 0$. In that case, B and C first look and compute in configuration $\mathcal{C}_0$, then B moves. Note that, because $x > 0$ and $y = x + 1$, B does not meet C. Then, A and B look and compute in configuration $\mathcal{C}'_0$. Then, B moves and reaches the configuration $\mathcal{C}_0$. Finally, B and C move. As mentioned above, they must move symmetrically, i.e., in opposite orientation. Therefore, after each robot has moved (B has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(A, B) \leq dist(B, C)$ and B has not met C. We get a contradiction as above.
- The last case to be considered is when $x = 0$ and, when executing Algorithm $\mathcal{A}$ in configuration $\mathcal{C}_0$, robot C goes to its neighbor that is not adjacent with B. In that case, there are two possibilities for Robot A 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 C and A move first (but A remains on its position) and finally B moves. Therefore, after each robot has moved (B has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(A, B) \leq dist(B, C)$ and B has not met C. We get a contradiction as above.
  - Otherwise, Robot A looks, computes and moves. Here the configuration is such that B is at distance 2 from both other robots. Then, A and C looks, computes and moves. Since their view are identical, they must do the same move.
    If they go to the neighbors of B, we have reached the configuration $\mathcal{C}_c$ and then Case 1 ensures that $\mathcal{A}$ cannot perpetually clear the ring, a contradiction. Otherwise, B looks, computes and moves. Since its view is symmetrical, the adversarial can let it move toward A. Therefore, after each robot has moved (B has moved twice), the reached configuration $\mathcal{C}_1$ is such that $dist(A, B) \leq dist(B, C)$ and B has not met C. 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 searching 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 Fig. 3a) or they are far (see Fig. 3b). In both cases, all the possible configurations are symmetric and the axis of symmetry either passes on the middle of the interval of empty nodes or in 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). The obtained configuration falls into the second case (see Fig. 3c) and the ring is not searched. Hence, we can assume without loss of generality that the initial configuration falls in the second case.

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

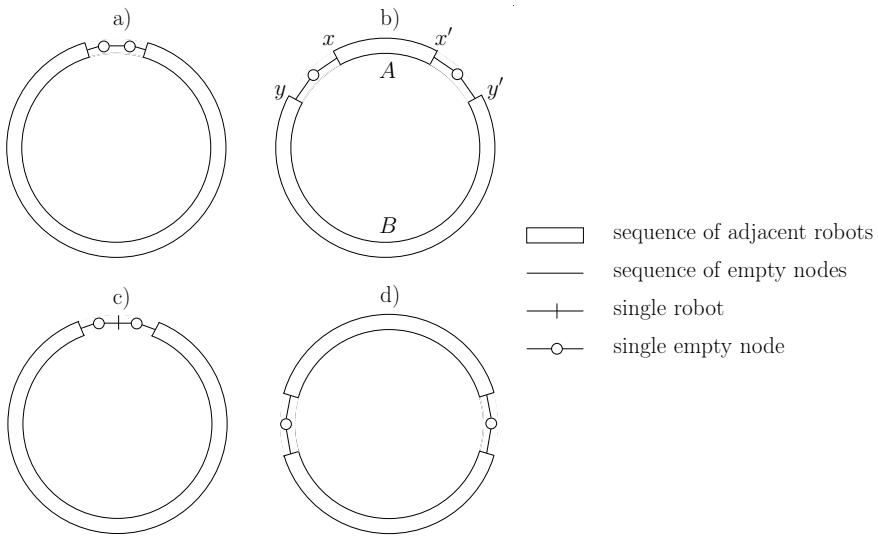


Figure 3: Configurations with $k = n - 2$.

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 Fig. 3d). 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 together 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 Fig. 3b 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 (Fig. 3a) or at distance one (Fig. 3c). 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 searching 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 searching the ring. ■

The next theorem is proven by an exhaustive study of the possible configurations. To prove Theorem 5, we first need the following lemmata.

Lemma 7 *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 perpetual searching of the ring.*

Proof. Indeed, there is a node v that is unoccupied on the axis of symmetry of the initial configuration. Because of the symmetry, the adversarial scheduler can e 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. ■

Lemma 8 *No algorithm starting from (or reaching at some step) a configuration where all the $k < n$ robots occupy consecutive nodes allows the perpetual searching of the ring.*

Proof. The proof is similar to the one in the case $k = 3$, in the proof of Theorem 3. ■

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 graph searching problem in a n-node 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 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.

In what follows, we consider any algorithm $\mathcal{A}$ for perpetual graph searching. 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 4-7. In these figures, grey nodes are the occupied nodes. An arc from Configuration C_1 to Configuration C_2, with label a means that Robot a in Configuration C_1 moves such that Configuration C_2 is reached. Note that the labels of robots in C_1 and C_2 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 4. Moreover, Configurations A_2, A_3 and A_4 are symmetric and satisfy the requirements of Lemma 7. Therefore, by Lemma 7, no algorithm can perpetually clear a ring if it reaches one of these three configuration. It only remains to prove that Algorithm $\mathcal{A}$ cannot perpetually avoid these configurations. Indeed, in Configuration A_1, moving robot b or c, or moving robot a toward c leads to A_4, A_3 or A_2 respectively. The only remaining move consists in moving a toward b. However, perpetually executing this move cannot clear the ring.

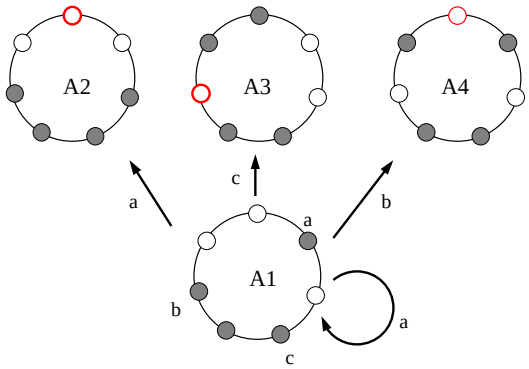


Figure 4: Theorem 5. Case $(k, n) = (4, 7)$. Grey nodes are the occupied ones.

- **Case** $(k, n) = (4, 8)$**.** In that case, there are 8 distinct configurations that are depicted in Figure 5. By Lemma 8, if Algorithm $\mathcal{A}$ reaches Configuration B_1, then the clearing fails. Similarly, if Configuration B_8 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 $\mathcal{A}$ eventually reaches Configurations B_1 or B_8 and thus cannot clear the ring.

  In Configuration B_3, 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 unoccupied neighbor. Moreover, since Robots a and b have the same view, the adversary can schedule both of them simultaneously such that Configuration B_8 is reached. In Configuration B_7, all robots have the same view. Then, the adversary can schedule Robots a and b such that Configuration B_8 is reached. Therefore, if $\mathcal{A}$ reaches Configurations B_3 or B_7, it will eventually reach Configuration B_8, which, by previous paragraph, cannot clear the ring. Thus, $\mathcal{A}$ must never reach Configurations B_3 or B_7.

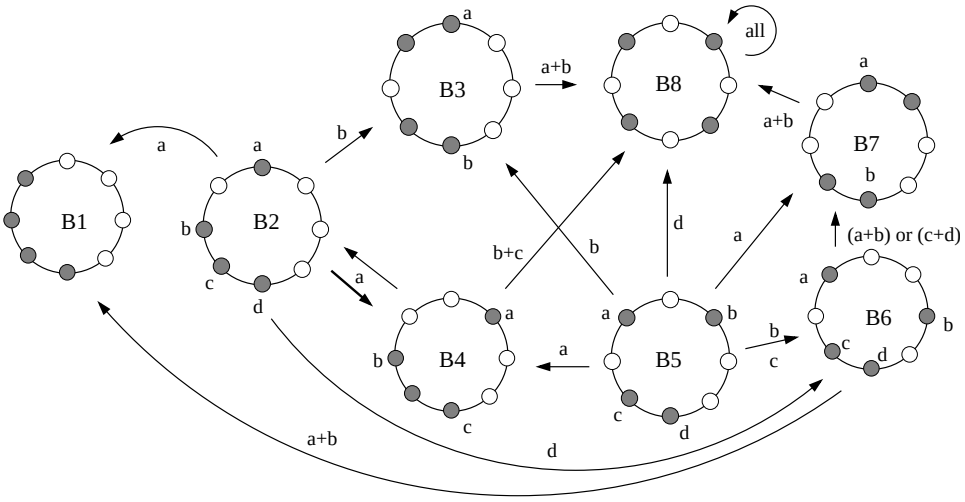


Figure 5: Theorem 5. Case $(k, n) = (4, 8)$. Grey nodes are the occupied ones.

Now, let us consider Configuration B_6. In such a configuration, Robots a and b have the same view and Robots c and d have the same view. If $\mathcal{A}$ allows Robots c and d to move, the adversary can move them simultaneously to reach Configuration B_7. If Robots a and b can move, the adversary can move them simultaneously and symmetrically to reach Configuration B_7 or B_1. Thus, $\mathcal{A}$ must never reach Configuration B_6.

In Configuration B_2, if Algorithm $\mathcal{A}$ makes Robot a move towards b, then the adversary can reach Configuration B_1. If Robot b, resp. Robot d, can move to its unoccupied neighbor, then the adversary can reach Configuration B_3, resp., Configuration B_6. Therefore, any Algorithm $\mathcal{A}$ that perpetually clears the ring must never reach Configuration B_2 or, in such a configuration, only allows a to move to its unoccupied neighbor that is symmetric to c.

In Configuration B_4, Robots b and c have the same view. If Algorithm $\mathcal{A}$ allows them to move, the adversary can move them simultaneously to reach Configuration B_8. Therefore, only Robot a can move. However, in that case, Configuration B_2 is reached and, by previous paragraph, this would lead to a strategy where only Robot a moves, oscillating between Configurations B_4 and B_2, which does not clear the ring.

To conclude, in Configuration B_5, 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 6.

  By Lemma 8, if Algorithm $\mathcal{A}$ reaches Configuration C_2, then the clearing fails. Therefore, in Configuration C_4, Algorithm $\mathcal{A}$ must not allow Robots a and b (that have the same view) to move since, otherwise, the adversary may reach Configuration C_2 or to make them collide in their common neighbor. Therefore, in Configuration C_4, only Robots c and d (that have the same view) can move to their unique unoccupied neighbor. Hence, any Algorithm $\mathcal{A}$ that clears the ring and that reaches Configuration C_4 must reach Configurations C_5.

  We now show that any Algorithm $\mathcal{A}$ that clears the ring must reach Configurations C_1, C_4 or C_5. Indeed, otherwise (since C_2 cannot be reached), it would mean that Algorithm

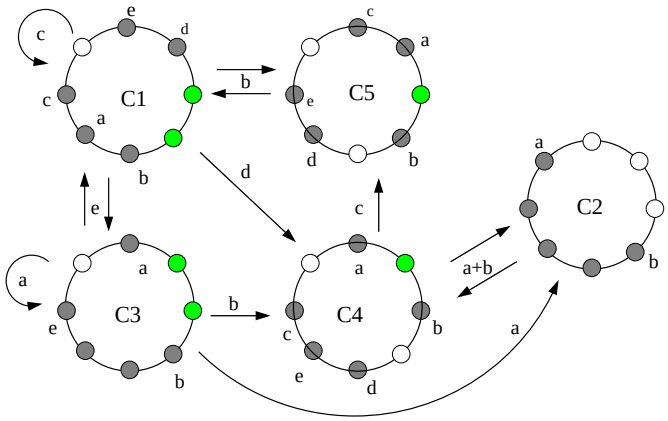


Figure 6: Theorem 5. Case $(k, n) = (5, 8)$. Grey nodes are the occupied ones.

$\mathcal{A}$ perpetually stays in Configuration C_3 (the single possibility is then that Robot a oscillates and that the neighbor of b is never occupies nor cleared) which clearly does not clear the ring. By previous paragraph, any Algorithm $\mathcal{A}$ that clears the ring must reach Configurations C_1 or C_5.

In Configuration C_5, Robots c and e (that have the same view) cannot move, since otherwise, the adversary could make them collide in their common unoccupied 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 C_4, then b (the robot labelled b in C_5) 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 C_5. 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 C_5, any Algorithm $\mathcal{A}$ that perpetually clears the ring can only allow Robot b to move. In particular, any Algorithm $\mathcal{A}$ that clears the ring must reach Configuration C_1.

Now, assume that, in Configuration C_1, Algorithm $\mathcal{A}$ allows Robots b to move. Therefore, in Configuration C_1, the adversary can schedule Robot b to reach Configuration C_5, 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 C_1. 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 $\mathcal{A}$ that perpetually clears the ring must not allow Robot b to move in Configuration C_1.

Similarly, let us assume that, in Configuration C_1, Algorithm $\mathcal{A}$ allows Robots d to move. Therefore, in Configuration C_1, the adversary can schedule Robot d to reach Configuration C_4 where the adversary schedules Robots a, b and e that cannot move by above paragraphs. Then, the adversary schedules Robot c, which reaches Configuration C_5 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 C_1, still without clearing the ring. Therefore, any Algorithm $\mathcal{A}$ that perpetually clears the ring must not allow Robot d to move in Configuration C_1.

Since we proved that any Algorithm $\mathcal{A}$ that perpetually clears the ring must reach Configuration C_1 and that, in such a configuration, neither d nor b can move, then Robot e must be able to move. Indeed, otherwise, $\mathcal{A}$ would reach Configuration C_1 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 C_3, Algorithm $\mathcal{A}$ allows Robot b to move. In that case, the adversary can schedule b (in C_3), then a, b, d, e in C_4 without any move, then c in C_4, then b in C_5, and finally e in C_1 reaching back C_3 without clearing the ring the green vertices are never cleared). Therefore, in C_3, only e may move towards a or a may move towards e.

To conclude, it is sufficient to note that, when C_1 is eventually reached (which must be by above discussion), then the adversary can ensure to oscillate between Configurations C_1 and C_3, 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 7.

  By Lemma 7, if Algorithm $\mathcal{A}$ reaches Configuration D_3 or D_4, then the clearing fails. If Configuration D_2 is reached, all robots have the same view, if one of them can move, the one with the same common unoccupied neighbor can also move which would create a collision. Therefore, if Algorithm $\mathcal{A}$ reaches Configuration D_2, then the clearing fails. If Configuration D_1 is reached, only Robots a and b may move (if the other two robots with an unoccupied neighbor move, they would collide in it). In that case, Robots a and b are scheduled simultaneously which reaches Configuration D_2 that is forbidden. Therefore, any Algorithm $\mathcal{A}$ that clears the ring must never reach one of these four configurations.

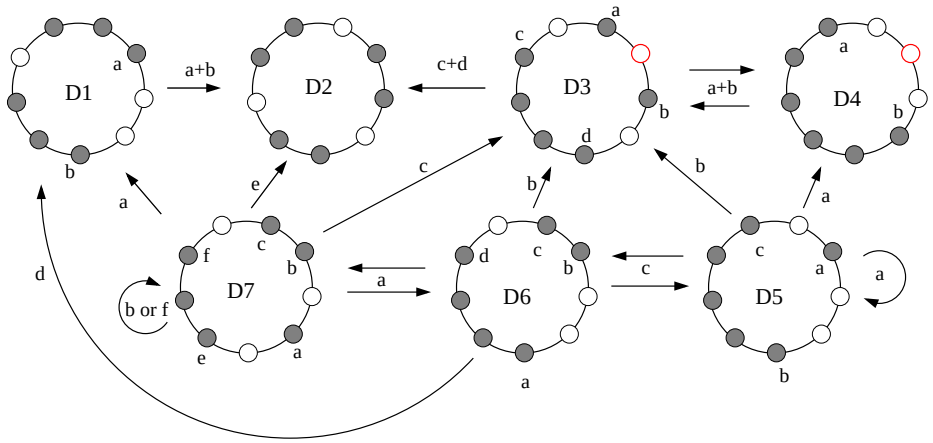


Figure 7: Theorem 5. Case $(k, n) = (6, 9)$. Grey nodes are the occupied ones.

Therefore, the only possible moves are the following. In Configuration D_5, Robot a can move towards b, and c can move. In Configuration D_6, only Robots c and a can move. In Configuration D_7, b and f can move, and a can move towards e. Hence, the adversary executes the following schedule. Arriving (or starting) in D_7, 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 D_7 by moving a (resp., arriving from D_5 by moving c, resp., starting) at D_6, 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 D_7, resp. to D_5) and goes on. Arriving from D_6 by moving c (or starting) at D_5, the adversary sequentially schedules b, d, e, f (that does not move), then a twice, and then c (possibly going to D_6) 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 8.

  By Lemma 7, if Algorithm $\mathcal{A}$ reaches Configuration E_1 to E_6, then the clearing fails. Therefore, in particular, any Algorithm $\mathcal{A}$ that clears the ring never reaches Configuration E_7 or only b is allowed to move towards a in that configuration. Similarly, in Configuration E_9, the possible moves are: Robot b moving towards c and Robot a moving towards d. In Configuration E_{10}, 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 E_8. It is easy to check that, allowing only these moves cannot clear the green vertices. ■

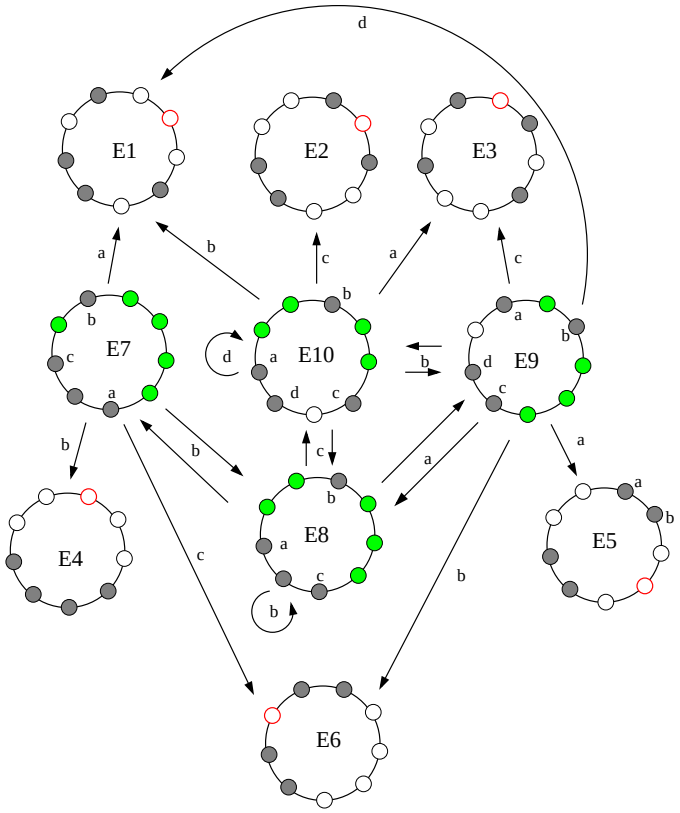


Figure 8: Theorem 5. Case $(k, n) = (4, 9)$. Grey nodes are the occupied ones.

- **Case** $(k, n) = (5, 9)$**.** In that case, there are 10 distinct configurations that are depicted

in Figure 9. For purpose of contradiction, let us assume that Algorithm $\mathcal{A}$ clears the ring. We consider several case depending on the Algorithm $\mathcal{A}$.

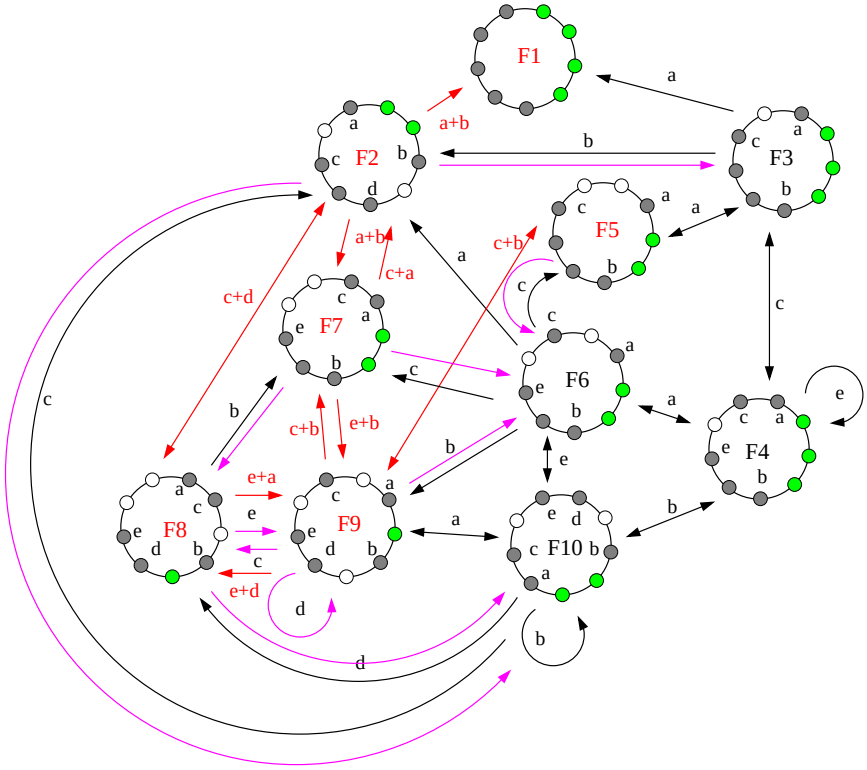


Figure 9: Theorem 5. Case $(k, n) = (5, 9)$. Grey nodes are the occupied ones. In case of symmetrical configurations, there are several cases depending whether the adversary schedules one robot (pink transitions) or two symmetrical robots (red transitions).

If Algorithm $\mathcal{A}$ eventually reaches Configuration F_1, then the clearing fails by Lemma 8. We then assume it is not the case.

First, let us assume that, in configuration F_6, $\mathcal{A}$ 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 F_6 by moving another robot first. This case is depicted in Figure 10 where the dotted move ins 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 F_8 to F_7, 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 $\mathcal{A}$ cannot clear the ring.

By previous paragraph, in Configuration F_6, Algorithm $\mathcal{A}$ only allows Robots a to move towards b. Moreover, Configuration F_6 followed by Configuration F_2 must be met infinitely often. Therefore, we can assume that F_2 is the initial configuration and that we have to go back to it via F_6.

Now, assume that, in Configuration F_9, $\mathcal{A}$ allows other move than moving Robot a towards

c or moving d. Such an Algorithm $\mathcal{A}$ cannot clear the ring since the adversary can ensure that only Configurations F_2, F_7, F_8 and F_9 are perpetually reached, which does not allow to clear the green nodes. Therefore, in Configuration F_9, Algorithm $\mathcal{A}$ 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).

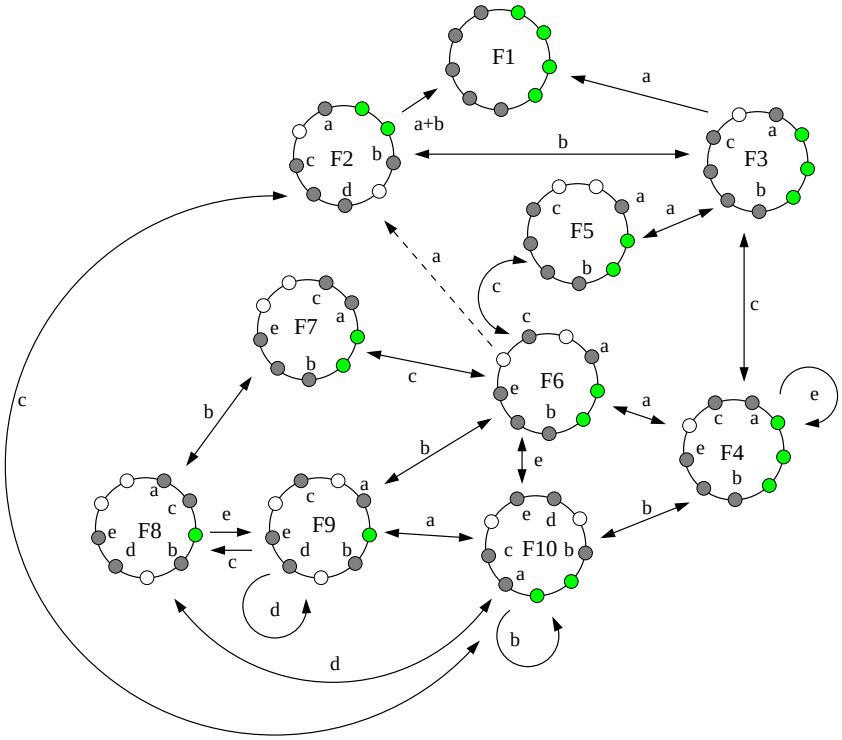


Figure 10: Theorem 5. Case $(k, n) = (5, 9)$. Grey nodes are the occupied ones.

We now show that, in Configuration F_2, Algorithm $\mathcal{A}$ has to allow Robots c and d to move and that these are the single possible moves. Indeed, let us assume that, in Configuration F_2, Algorithm $\mathcal{A}$ allows Robot a (resp., b) decides to move toward b (resp., towards a). In that case, the adversary executes the following schedule: in F_2, it schedules Robot a going to Configuration F_6, then in F_6, it schedules all robots but a, that do not move, and then a going back to Configuration F_2 without having cleared the ring. Therefore, in Configuration F_2, Algorithm $\mathcal{A}$ does not allow Robot a to move toward b. Since, because Configuration F_1 cannot be reached, 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 F_{10}, Robot e must be allowed to move.

First, assume Robot b is allowed to move in F_{10}. Then, the adversary can schedule Robots c and d in F_2 but only Robot c moves, i.e., Robot d has looked and computed but not moved yet. Therefore, arriving in Configuration F_{10}, 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 F_{10}.

Second, assume for purpose of contradiction, that Robot a (resp., Robot d) is allowed to

move in F_{10}. Then, the adversary can execute the following schedule: Robot c moves in F_2 to reach F_{10}, then Robot a moves reaching F_9 (resp., Robot d moves to F_8 and then Robot e moves to F_9), then all Robots but a are scheduled without moving in F_9 (see above), then Robot a move to reach back F_{10} and then, the adversary can alternate Configurations F_9 and F_{10} (resp., F_8, F_9 and F_{10}) without clearing the ring.

Similarly, assume for purpose of contradiction, that Robot c is allowed to move in F_{10}. Then, the adversary can execute the following schedule: in F_2, Robots a, b, e are scheduled without moving (see above), then Robot c moves in F_2 to reach F_{10}, then Robot c moves reaching back F_2, and then this cycle is done again with Robot d moving in F_2 instead of c. Thus, the adversary can alternate Configurations F_2 and F_{10} without clearing the ring.

Therefore, in Configuration F_{10}, 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 F_2 to reach F_{10}, and then, it schedules all robots, where only Robot b moves perpetually remaining in Configuration F_{10} which does not clear the ring.

To conclude the proof, the adversary does the following. In Configuration F_2 (we proved above that we may assume we can start with this configuration), Robot c (or d) moves and reaches Configuration F_{10}, where Robot e moves which reaches Configuration F_6, where all robots but a are scheduled without moving. Finally, Robot a is moving reaching back Configuration F_2. It is easy to check that this does not allow to clear the ring.

■

4.3 Algorithm Ring Clearing

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

Algorithm Ring Clearing works in two phases and it is formally described in Fig. 11. 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 Fig. 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. Note that the configuration $\mathcal{C}^*$ belongs to the set of configurations $\mathcal{A}$-f.

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

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

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

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

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

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

The pseudo-code of Algorithm Ring Clearing is reported in Fig. 11. First, at line 2, the algorithm performs Align until a configuration in $\mathcal{A}$ is achieved. Let us denote as r and r' as in the proof of Theorem 6. At line 4, the algorithm identifies robot r in a configuration in $\mathcal{A}$-a. The configuration read by r is $(q_0, q_1, \ldots, q_j) = (0, 1, 0, 0, \ldots, 0, q_j)$, where $q_j > 2$. In this case r has to move towards the direction opposite to r' that is towards q_j (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, 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 Fig. 12, then the configuration read is $(q_0, q_1, \ldots, q_j) = (q_0, 0, 0, \ldots, 0, 1, q_j)$, where $q_0 > 2$ and $q_j > 0$ (line 11) and r has to move towards q_0 (line 15). Otherwise, the configuration read is $(q_0, q_1, \ldots, q_j) = (q_0, 1, 0, 0, \ldots, 0, q_j)$, where $q_0 > 0$ and $q_j > 2$ (line 5) and r has to move towards q_j (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 conditions is satisfied for exactly one robot.

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

```
1  if C ∉ A then
2      Apply Algorithm Align
3  else
4      if (q_0 = 0, q_1 = 1, q_i = 0 ∀i ∈ {2,3,...,k−2}, q_{k−1} > 2)        // A-a
5      OR (q_0 > 0, q_{k−1} > 2, q_1 = 1, q_i = 0 ∀i ∈ {2,3,...,k−2})        // A-b
6      OR (q_i = 0 ∀i ∈ {0,1,...,k−4}, q_{k−3} = 2, q_{k−2} > 0, q_{k−1} = 1) // A-c
7      OR (q_0 > 0, q_1 = 0, q_2 = 1, q_i = 0 ∀i ∈ {3,4,...,k−2}, q_{k−1} > 2) // A-d
8      OR (q_i = 0 ∀i ∈ {0,1,...,k−3}, q_{k−2} > q_{k−1} > 0, q_{k−2} + q_{k−1} > 3) // A-f
9      then
10         move towards q_{k−1};
11     if (q_0 > 2, q_{k−1} > 0, q_i = 0 ∀i ∈ {1,2,...,j−2}, q_{k−2} = 1)   // A-b
12     OR (q_0 = 2, q_i = 0 ∀i ∈ {1,2,...,k−4}, q_{k−3} = 1, q_{k−2} = 0, q_{k−1} > 0) // A-d
13     OR (q_0 = 1, q_i = 0 ∀i ∈ {1,2,...,k−4}, q_{k−3} = 1, q_{k−2} = 0, q_{k−1} > 1) // A-e
14     then
15         move towards q_0;
```

Figure 11: Algorithm Ring Clearing.

The algorithm perpetually cycles among configurations $\mathcal{A}$-a — $\mathcal{A}$-e as depicted in Fig. 12. The next theorem shows that it perpetually clears the ring.

Theorem 6 *Starting from any exclusive and rigid configuration, Algorithm* Ring Clearing *solves both the perpetual exclusive graph searching and exploration problems using k robots in any n-node ring, $n \geq 10$ and $5 \leq k < n-3$ (but for $n = 10$ and $k = 5$).*

Proof. By Theorem 1, Algorithm Align (line 2) 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

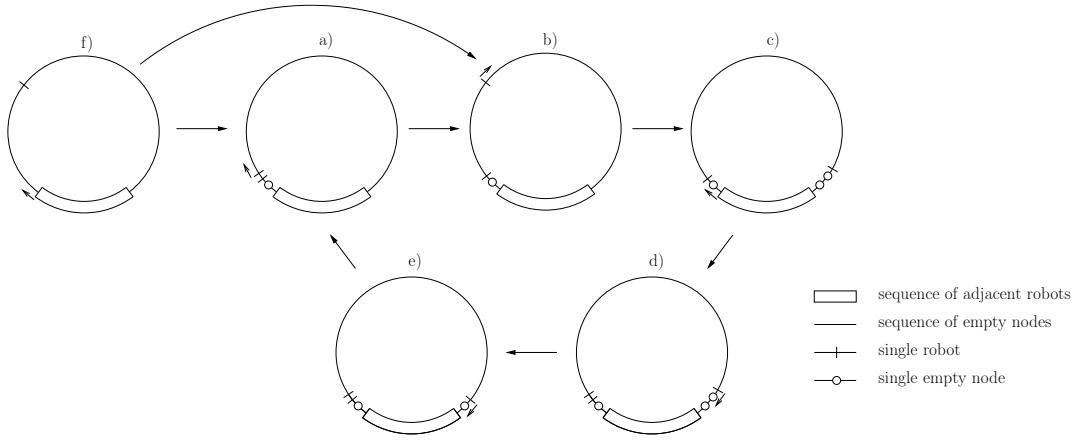


Figure 12: Second phase of Algorithm Ring Clearing. The arrows close to the robots indicate the robot that is moving and its direction.

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 moves 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. At line 4, 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 searched are the internal edges of S and the edge between r and r'. The algorithm first searches the edges in the sequence of empty nodes. To this aim, it moves robot r towards the direction opposite to r' (line 10). 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. Configurations in $\mathcal{A}$-b are identified at lines 5 and 11. In particular, robot r can read the configuration in two directions, depending on the size of the its adjacent intervals.

In this way, r searched 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 searched. 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 Fig. 12, then the configuration read is $(q_0, q_1, \ldots, q_j) = (q_0, 0, 0, \ldots, 0, 1, q_{k-1})$, 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 $(q_0, q_1, \ldots, q_j) = (q_0, 1, 0, 0, \ldots, 0, q_{k-1})$, where $q_0 > 0$ and $q_{k-1} > 2$ (line 5) and r has to move towards q_{k-1} (line 10).

When the configuration is of type $\mathcal{A}$-c, the only edges which are not searched 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', searching 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 searched 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 configuration, 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 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: $(0, 2, k-2)$, $(0, 3, k-3)$, $(1, 2, k-3)$. 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 and it is formally given in Fig. 13.

Procedure: NminusThree
Input: Rigid and exclusive configuration $\mathcal{C}$ 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
```

Figure 13: Algorithm NminusThree.

It works in two phases: In the first phase, it creates a final configuration and in the second one it performs the perpetual searching.

The first phase is performed if the configuration is not one of the final ones 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;

R1.2: If $B = 1$ move towards B the robot on the border of C which is closer to B;

R1.3: If $B > 3$ move towards C the robot on the border of B which is closer to C.

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 9 *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 searching. 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;

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;

R2.3: If $(A, B, C) = (1, 2, k - 3)$ move the robot of A towards C.

The following theorem states the correctness of the algorithm.

Theorem 7 *Starting from any exclusive and rigid configuration, Algorithm* NminusThree *solves both the perpetual exclusive graph searching 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 9 the first phase of the algorithm always achieves a final configuration. It remain to show that the second phase perpetually performs the searching. 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 the min-CORDA model

In this section, we devise a strategy to accomplish the *gathering* task on a ring under the min-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 in [20] it has been proved that the gathering on rings is unfeasible without any multiplicity detection, and the local is the lightest one that can be assumed.

In accordance to the tasks previously shown, we make use of procedure ALIGN in order to achieve configuration $\mathcal{C}^*$ starting from any (exclusive) rigid configuration on rings of $n > k+2$ nodes and $k > 2$ robots. In fact, any configuration with $n = 2$, $n = k+1$, or $n = k+2$ nodes 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 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 Fig. 14.

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;
```

Figure 14: 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 Further work

In this work, we provided a unified strategy for solving three tasks in the minimalist-CORDA model on ring topologies when the initial configuration is rigid. Namely we solved the exclusive perpetual search, 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 graph searching problem fully characterize any initial 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.

References

[1] 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.

[2] 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 *LNCS*, pages 428–445. Springer-Verlag, 2008.

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

[4] L. Blin, J. Burman, and N. Nisse. Perpetual Graph Searching. Technical Report RR-7897, INRIA, 2012. Submitted.

[5] 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 *LNCS*, pages 312–327. Springer, 2010.

[6] 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 *LNCS*, pages 251–265. Springer, 2011.

[7] 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 *LNCS*, pages 327–338, 2012.

[8] 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 *LNCS*, pages 174–185, 2011.

[9] G. D'Angelo, G. Di Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. Rapport de recherche RR-7963, INRIA, 2012. Submitted.

[10] 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 *LNCS*, pages 195–208, 2009.

[11] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*. To appear.

[12] 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.

[13] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *10th Int. Symp. on Algorithms and Computation (ISAAC)*, volume 1741 of *LNCS*, pages 93–102. Springer, 1999.

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

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

[16] 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 *LNCS*, pages 101–113, 2010.

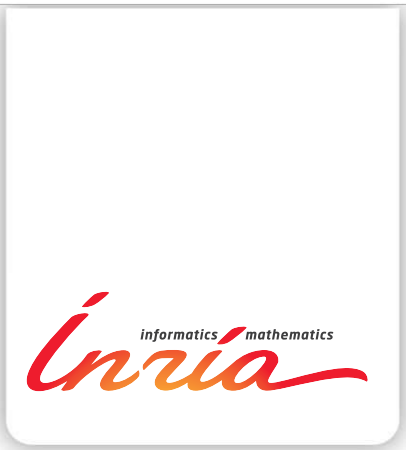
[17] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations. In *18th Int. Coll. on Structural Information and Communication Complexity (SIROCCO)*, volume 6796 of *LNCS*, pages 150–161, 2011.

[18] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *37th Int. Symp. on Mathematical Foundations of Computer Science (MFCS)*. Springer-Verlag, 2012. To appear.

[19] 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.

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

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



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399