Noname manuscript No.
(will be inserted by the editor)

A unified approach for Gathering and Exclusive Searching on rings under weak assumptions.

Gianlorenzo D'Angelo · Alfredo Navarra · Nicolas Nisse

Received: date / Accepted: date

Abstract Consider a set of mobile robots placed on distinct nodes of a discrete, anonymous, and bidirectional ring. Asynchronously, each robot takes a snapshot of the ring, determining the size of the ring and which nodes are either occupied by robots or empty. Based on the observed configuration, it decides whether to move to one of its adjacent nodes or not. In the first case, it performs the computed move, eventually. This model of computation is known as *Look-Compute-Move*. The computation depends on the required task. In this paper, we solve both the well-known *Gathering* and *Exclusive Searching* tasks. In the former problem, all robots must simultaneously occupy the same node, eventually. In the latter problem, the aim is to clear all edges of the graph. An edge is cleared if it is traversed by a robot or if both its endpoints are occupied. We consider the *exclusive* searching where it must be ensured that two robots never occupy the same node. Moreover, since the robots are oblivious, the clearing is *perpetual*, i.e., the ring is cleared infinitely often.

In the literature, most contributions are restricted to a subset of initial configurations. Here, we design two different algorithms and provide a characterization of the initial configurations that permit the resolution of the problems under very weak assumptions. More precisely, we provide a full characterization (except for few pathological cases) of the initial configurations for which gathering can be solved. The algorithm relies on the necessary assumption of the local-weak multiplicity detection. This means that during the Look phase a robot detects also whether the node it occupies is occupied by other robots, without acquiring the exact number.

For the exclusive searching, we characterize all (except for few pathological cases) aperiodic configurations from which the problem is feasible. We also provide some impossibility results for the case of periodic configurations.

Preliminary results concerning this work were presented in the 15th International Conference on Distributed Computing and Networking (ICDCN'14) [15].

G. D'Angelo
Gran Sasso Science Institute (GSSI).
Via F. Crispi, 7, I-67100, L'Aquila, Italy.
Tel: +39 0862 4280 406
E-mail: gianlorenzo.dangelo@gssi.infn.it

A. Navarra
Dipartimento di Matematica e Informatica,
Università degli Studi di Perugia.
Via Vanvitelli, 1, I-06123, Perugia, Italy.
Tel: +39 075 585 5046
E-mail: alfredo.navarra@unipg.it

N. Nisse
INRIA/I3S(CNRS/UNSA).
2004, route des Lucioles, B.P. 93, F-06902 Sophia Antipolis Cedex, France.
Tel: +33 (0)4 97 15 53 28
E-mail: nicolas.nisse@inria.fr

1 Introduction

In the field of robot-based computing systems, the study of the minimal settings required to accomplish specific tasks represents a challenging issue. We consider k robots initially placed on distinct nodes of a discrete, anonymous, and bidirectional ring of n nodes, and we investigate two fundamental problems requiring complex coordination: *Gathering* (see, e.g., [1,6,9,11,18,32]) and *Exclusive Searching* (see, e.g., [3,24,25]).

We assume minimal abilities for the robots. They are oblivious (without memory of the past), uniform (running the same deterministic algorithm), autonomous (without centralized control), anonymous (without identities), unoriented (without a common coordinate system, or common compass), asynchronous (without a global clock that synchronize their actions), and silent (without the capability to communicate). Neither nodes nor edges are labeled and no local memory is available on nodes. Robots are equipped with visibility sensors and motion actuators, and operate in *Look-Compute-Move* cycles in order to achieve a common task (see [22]). 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, makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case it moves to this node (Move). In other words, each robot executes an algorithm that takes as input a snapshot or *configuration*, i.e., the graph topology and the set of nodes occupied by the robots, and computes the *move* of the robot. Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move operations is finite but unbounded, and it is decided by an adversary for each robot. Hence, robots that cannot communicate may move based on outdated perceptions. The adversary (scheduler) that determines the timing of the Look-Compute-Move cycles is assumed to be fair: each robot performs its cycle within finite time and infinitely often.

The asynchronous Look-Compute-Move model, also called *CORDA*, was first defined in continuous environments [8,23,27,33]. The inaccuracy of the sensors used by robots to scan the surrounding environment motivates its discretization. Robots can also model software agents moving on a computer network. Many robots coordination problems were considered in discrete environments. Exploration with stop was studied in paths [20], trees [19], rings [21], and general graphs [7]. More recently, the gathering problem (a.k.a. Rendez-vous) was considered in rings [12–14,31], grids [2,10,16] and trees [10]. Exclusive perpetual exploration was studied in rings [4] and grids [5]. The *exclusivity property* states that any node must be occupied by at most one robot. Very recently, exclusive searching was defined and studied in trees [3] and rings [14]. In all previous works as well as in this paper, initial *configurations* are assumed to be *exclusive*, that is, any node is occupied by at most one robot.

In this paper, we focus on the ring topology. The relevance of the ring topology is motivated by its completely symmetric structure. It means that algorithms for rings are more difficult to devise as they cannot exploit any topological structure, assuming that all nodes look the same. In fact, our algorithms are only based on robots' positioning and not on the graph topology. On rings, different types of exclusive configurations may require different approaches. In particular, periodicity and symmetry arguments must be carefully handled. An exclusive configuration is called *periodic* if it is invariable under non-complete rotations. It is called *symmetric* if the ring has an *axis of symmetry* that reflects single robots into single robots, and empty nodes into empty nodes. It is called *rigid* if it is aperiodic and asymmetric. We consider the following two problems.

Gathering. The gathering problem consists in moving all the robots toward the same node and remain there. On rings, under the Look-Compute-Move model, gathering is infeasible if the robots are not empowered by the so-called *multiplicity detection* capability [31]. This capability permits to robots to perceive during the Look phase whether a node is occupied by robots or not. In the *global-strong* version, a robot is able to perceive the exact number of robots that occupy a same node. In the *global-weak* version, a robot perceives only whether a node is occupied by one robot or if a multiplicity occurs, i.e., the node is occupied by an undefined number of robots greater than one. In the *local-strong* version, a robot can perceive only whether a node is occupied or not, but it is able to perceive the exact number of robots occupying the node where it resides. Finally, in the *local-weak* version, a robot can perceive the multiplicity only on the node where it resides but not the exact number of robots composing it. In this paper, we assume that robots are empowered with the local-weak multiplicity detection capability.

For the ring topology under the global-weak multiplicity detection capability, some impossibility results were proven [31]. Such results clearly hold also when assuming the local-weak multiplicity detection. Several algorithms were proposed for different kinds of exclusive initial configurations [13,30,31]. These papers left open some cases which were closed in [12] where a unified strategy for all the gatherable configurations was provided.

Configuration type	number of nodes n	number of robots k	Gathering feasibility
periodic	-	-	NO [31]
symmetric with edge-edge axis	-	-	NO [31]
-	-	$k = 2$	NO [31]
symmetric	odd	$k = 4$	NO [12,17,30]
rigid	-	$k < \lfloor \frac{n}{2} \rfloor$	YES [26]
aperiodic	-	odd, $k < n - 3$	YES [28]
aperiodic	odd	even, $10 \leq k \leq n - 5$	YES [29]
rigid	-	-	YES [14]
aperiodic, without edge-edge axis	-	$3 \leq k \leq n - 4$, $k \neq 4$	**YES (this paper)**

Table 1: Resume of the known results about gathering in a ring under the Look-Compute-Move model with the local-weak multiplicity detection. All the mentioned configurations are exclusive.

With local-weak multiplicity detection capability, see Table 1, an algorithm starting from rigid configurations where the number of robots k is strictly smaller than $\lfloor \frac{n}{2} \rfloor$ was designed in [26]. In [28], the case where k is odd and strictly smaller than $n-3$ was solved. In [29], the authors provide an algorithm for the case where n is odd, k is even, and $10 \leq k \leq n-5$. Recently, the case of rigid configurations was solved in [14]. The remaining cases are left open and a unified algorithm for all the cases is still unknown.

Exclusive Searching. Graph searching was widely studied in centralized and distributed settings (see e.g., [24,25]). 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 endpoints are occupied. However, a cleared edge is re-contaminated if there is a path without robots from a contaminated edge to it. A graph is *searched* if there exists a time when all its edges are simultaneously clear. For instance, in a centralized setting, two robots are sufficient to clear a ring, starting from a node and moving in opposite directions. In a distributed setting, the task is much harder due to symmetries and asynchrony. Following [3,14], we also consider an additional constraint: the so called *exclusivity property*, that is, no two robots can be concurrently on the same node or cross the same edge. We consider the exclusivity constraint since in the Look-Compute-Move model two robots occupying the same node may act like a single one as they execute the same deterministic algorithm. It follows that there are no strategies that can take advantage from allowing more robots on a single node. If a strategy is based on the occupancy of a same node by means of more than one robot, the adversary can easily break this synchrony. One of the advantages of studying exclusive searching in this model is that it provided the first "fault-tolerant" algorithms for this task. Indeed, whatever be the starting configuration (among the ones we prove to be feasible), the robots are able to perpetually (infinitely often) clear the ring. See Table 2 for results on the ring topology. Moreover, as the robots are oblivious, they cannot recognize which edges are already cleared, therefore they must repeatedly perform the task. The searching is called *perpetual* if it is accomplished infinitely many times.

The study of exclusive searching in the discrete model was introduced in [3] for tree topologies. Concerning rings, in [14] the case of initial rigid configurations was tackled.

Contribution. We consider the gathering with local-weak multiplicity detection and the perpetual exclusive searching problems for k robots in an n-node ring.

For any $k < n - 4$, $k \neq 4$, we characterize the exclusive configurations from which the *gathering* problem is feasible, see Table 1. In particular, we design an algorithm that solves the problem starting from any exclusive configuration with $k < n - 4$, $k \neq 4$, robots empowered by the local-weak multiplicity detection, except for the infeasible configurations that will be specified later. Similarly to the case of $k = 4$ in [12] and $(n, k) = (7, 6)$ in [13], the cases left out from our characterization ($k = 4$ and $k \geq n-4$), if gatherable, would require specific algorithms difficult to generalize.

We then provide a characterization of any aperiodic exclusive configuration with $k \neq 4$, and $(n, k) \notin \{(10, 5), (10, 6)\}$ from which *exclusive searching* is feasible, see Table 2. That is, we design an algorithm that solves the problem starting from any such aperiodic exclusive configurations except for the infeasible ones. For periodic configurations, we provide some impossibility results.

Configuration type	number of nodes n	number of robots k	Searching feasibility
-	$3 \leq n \leq 9$	$k < n$	NO [14]
-	$4 < n$	$k \leq 3$ or $n-2 \leq k \leq n-1$	NO [14]
rigid	$10 \leq n$, $5 \leq k \leq n-3$ and $(n,k) \neq (5,10)$		YES [14]
axis through empty node	-	even	**NO (this paper)**
periodic with 1 or 2 empty nodes per period	-	-	**NO (this paper)**
aperiodic, and (only if k even) without axis through empty node	$10 \leq n$	$5 \leq k \leq n-3$ and $(n,k) \notin \{(5,10);(6,10)\}$	**YES (this paper)**

Table 2: Resume of the known results about exclusive searching in a ring under the Look-Compute-Move model. All the mentioned configurations are exclusive.

Designing a unified algorithm for all (periodic or not) configurations seems challenging.

The algorithms for gathering and exclusive searching (given in Sections 4 and 5, respectively) exploit a common technique (provided in Section 6) that allows to achieve some special configurations suitable for the subsequent phases. This result mainly relies on a non-trivial characterization of aperiodic configurations in a ring that could be used for further problems.

Outline. In the next section we provide useful definitions and the notation used in the paper. In Section 3, we present an high-level description of the algorithm that achieves some special configurations subsequently exploited by the specific algorithms for solving both the gathering and the exclusive searching tasks. The gathering problem is considered in Section 4. Exclusive searching is studied in Section 5. Section 6 is devoted to the formal details and proofs of the algorithm described in Section 3. This represents the core of our technical results. We then conclude by Section 7 with some possible future research directions.

2 Notation and preliminary

In this paper, we consider a bidirectional ring with $n \geq 3$ nodes $\{v_0, \cdots, v_{n-1}\}$, where v_i is connected to $v_{i+1 \bmod n}$ for any $0 \leq i < n$. Moreover, let $k \geq 1$ robots occupy k distinct nodes of the ring.

A *configuration* $\mathcal{C}$ is defined by the k nodes occupied by robots. In what follows, any configuration is seen as a binary sequences where "0" represents an occupied node while "1" stands for an empty node. More formally, given a configuration $\mathcal{C}$, and for any $i \leq n$, let $\mathcal{S}_i = (r_0^i, \cdots, r_{n-1}^i) \in \{0,1\}^n$ be the sequence such that $r_j^i = 0$ if $v_{i+j \bmod n}$ is occupied in $\mathcal{C}$ and $r_j^i = 1$ otherwise, $0 \leq j < n$. Intuitively, $\mathcal{S}_i$ represents the positions of robots, starting at node v_i.

For any $X = (x_0, \cdots, x_{r-1})$, let us denote $\overline{X} = (x_{r-1}, \cdots, x_0)$ and $X_i = (x_{i \bmod r}, \ldots, x_{r-1+i \bmod r})$. A *representation* of $\mathcal{C}$ is any sequence in $\mathcal{S}_\mathcal{C} = \{\mathcal{S}_i, \overline{(\mathcal{S}_i)}\}_{i<n}$. Abusing the notation, we say $\mathcal{C} = S$ for any $S \in \mathcal{S}_\mathcal{C}$. A configuration is said *exclusive* if no node is occupied by more than one robot. Note that, for any exclusive configuration $S = (s_0, \cdots, s_{n-1}) \in \mathcal{S}_\mathcal{C}$, $\sum_{i<n} s_i = n - k$. The view from a node/robot v_i is the minimum between $\mathcal{S}_i$ and $\overline{\mathcal{S}_i}$, this also represents what we call the *snapshot* of a configuration acquired by a robot during the Look phase.

A *supermin* of $\mathcal{C}$ is any representation of $\mathcal{C}$ that is minimum in the lexicographical order. We denote the supermin of $\mathcal{C}$ as $\mathcal{C}^{\min}$. In any supermin $(s_0, \cdots, s_{n-1})$, if $k < n$ then $s_{n-1} = 1$. From their view, all robots can compute the supermin of a configuration.

For $x \in \{0,1\}$, we denote by x^h a sequence of $h \geq 0$ consecutive x. Similarly, given a sequence X, denote by X^h a sequence of $h \geq 0$ consecutive replications of X. We say that a sequence X is a *palindrome* if $X = \overline{X}$, it is *symmetric* if X_i is a palindrome or $X_i = \overline{(X_{i+1})}$ for some i, and it is *periodic* if $X = X_i$, for some $0 < i < |X| - 1$. A configuration is symmetric (periodic, respectively) if at least one of its representations is symmetric (periodic, respectively). A configuration is *rigid* if it is neither symmetric nor periodic. It is known that an aperiodic configuration admits at most one axis of symmetry [12]. Moreover, an aperiodic configuration has either a unique supermin representation or two symmetrical supermin representations [12]. A configuration is said *feasible* with respect to a specific task if there exists an algorithm solving the task starting from such a configuration.

Execution model. Given a task that robots have to solve, one has to design a distributed algorithm that each robot applies. Robots operate in *Look-Compute-Move* cycles. In one cycle a robot takes a

snapshot of the current configuration (Look). We recall that a snapshot is the view taken from a robot where it cannot distinguish nor the nodes of the ring neither the identities of the other robots but it can distinguish whether each node is occupied by none, one or more robots and its relative position in the ring. In particular, from a snapshot, a robot can infer the size of the ring and the number of occupied nodes. Based on the perceived configuration, the robot applies the designed algorithm hence deciding whether to stay idle or to move to a neighboring node (Compute). If a move is computed, the robot performs it, eventually (Move). Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move operations is finite but unbounded, and it is decided by an adversary for each robot. Whereas, moves are considered instantaneous, that is, during the Look phase robots are always perceived on the nodes of the ring and not on its edges.

Allowed configurations. We now summarize the known feasible and infeasible exclusive configurations for both gathering and exclusive searching. In [31], it is shown that gathering is infeasible for $k = 2$, for any periodic initial configuration, and for any initial configuration with an axis of symmetry passing through two edges. In [14], it is shown that, for any exclusive configuration, it is not possible to search a ring using k robots if $n \leq 9$ or $k \leq 3$, or $k \geq n-2$. Here, we prove that exclusive searching is not feasible for any even k starting from any configuration with an axis of symmetry passing through an empty node.

The main goal of this paper is to extend the known feasibility results to a larger class of configurations. This class is called the class of *allowed* configurations, and our contribution is to show that they are feasible.

In what follows, an exclusive configuration is *allowed* for problem P if it is not periodic, if it does not admit an axis of symmetry (as described above) for which P is infeasible, and if the number of robots does not fall in the above defined impossibility ranges. In particular, all rigid configurations where the number of robots falls outside of the impossibility ranges are allowed. For gathering, the symmetric allowed configurations are all the aperiodic ones with the axis of symmetry not passing through two edges and with $3 \leq k < n-4$, $k \neq 4$. For exclusive searching, the symmetric allowed configurations are all the aperiodic ones with odd k and those with even k where the axis does not pass through an empty node, provided that $4 < k < n-2$ and $n > 9$.

Global and local view. For the ease of presentation, we prefer to describe the algorithms from a global perspective rather than a local one. This also helps the explanation of the correctness proofs. It is easy to see that each robot has all the information to compute whether it has to move or not according to the acquired configuration during its Look phase (i.e. its snapshot). For instance, suppose that from a given configuration $\mathcal{C}$, with supermin $\mathcal{C}^{\min} = (r_0, r_1, \ldots, r_{n-1})$, an algorithm (from a global perspective) makes the robot at r_i move toward r_{i+1}. Let $\mathcal{C}' = (r'_0, r'_1, \ldots, r'_{n-1})$ be the *local* view of a generic robot r. Then, r must check whether $\mathcal{C}' = \mathcal{C}_i^{\min}$ or $\mathcal{C}' = \overline{(\mathcal{C}_i^{\min})}$. If one of such cases occurs, then it deduces it is candidate to move toward r'_1 or r'_{n-1}, respectively.

Dealing with symmetry. The core of the technique in [14] for solving the problems from asymmetric exclusive configurations is Algorithm Asym. This allows to achieve a particular configuration called $\mathcal{C}^a = (0^{k-1}, 1, 0, 1^{n-k-1})$ made of $k-1$ consecutive robots, one empty node and one robot (see Figure 1a).

Lemma 1 ([14]) *Let $3 \leq k < n-2$ robots standing in an n-node ring and forming a rigid exclusive configuration, Algorithm* Asym *eventually terminates achieving configuration $\mathcal{C}^a$ and all intermediate configurations obtained are exclusive and rigid.*

Basically, Algorithm Asym ensures that, from any rigid exclusive configuration, one robot can be uniquely detected and is moved to an unoccupied neighbor by achieving another rigid configuration while strictly decreasing the supermin. Here, our main contribution is Algorithm Align that generalizes Asym by handling all allowed configurations (not only rigid). Difficulties are multiple.

First, in allowed symmetric configurations, we cannot ensure that a unique robot will move. In such a case, the algorithm may allow a robot r to move, while r is reflected by the axis of symmetry to another robot r'. Since r and r' are indistinguishable and execute the same algorithm, then r' should perform the same (symmetric) move. However, due to asynchrony, r may move while the corresponding move of r' is postponed (i.e. r' has performed the Look phase but not yet the Move phase). The configuration reached after the move

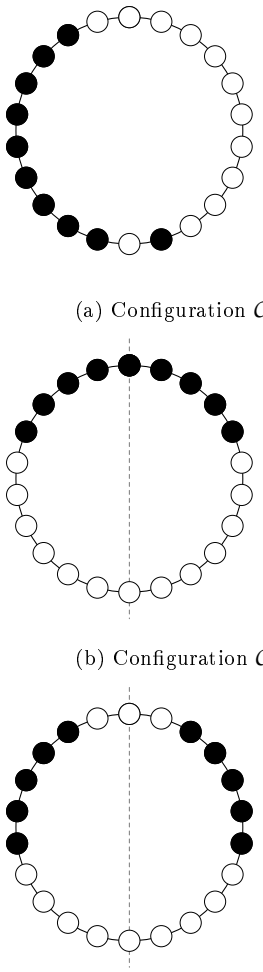
of r has a potential so-called *pending* move (the one of r' that will be executed eventually). To deal with this problem, our algorithm ensures that all the reached configurations that might have a pending move can be always detected as asymmetric configurations with a unique pending move. Therefore, in such a case, our algorithm forces to perform the pending move. That is, contrary to [14] where Algorithm Asym ensures to only go through rigid configurations, the subtlety here consists in possibly going from an asymmetric configuration to a symmetric one. To detect such configurations, we define the notion of adjacent configurations. Let us consider an algorithm $\mathcal{A}$ and a procedure M allowed by $\mathcal{A}$, that is algorithm $\mathcal{A}$ performs M for some configuration. Possibly, procedure M moves two (symmetric) robots. An asymmetric configuration $\mathcal{C}$ is *adjacent* to a symmetric configuration $\mathcal{C}'$ with respect to procedure M if $\mathcal{C}$ can be obtained from $\mathcal{C}'$ by applying M to only one of the robots permitted to move by M and the algorithm performs M on $\mathcal{C}$. In other words, if $\mathcal{C}$ is adjacent to $\mathcal{C}'$ with respect to M, there might exist a pending move permitted by M in $\mathcal{C}$. Another difficulty is to ensure that all met configurations are allowed for the considered problem P.

3 High-level description of Algorithm Align

Our contribution mainly relies on Algorithm Align. Such an algorithm starts from any configuration that is allowed either for the gathering or the exclusive searching problems and aims at reaching one of the configurations $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$ having supermin $(0^{k-1}, 1, 0, 1^{n-k-1})$, $(0^k, 1^{n-k})$, or, $(0^{\frac{k}{2}}, 1^j, 0^{\frac{k}{2}}, 1^{n-k-j})$ for k even and $j < \frac{n-k}{2}$, respectively (see Figure 1). From such configurations, we will show how to solve the gathering (Section 4) and the exclusive searching (Section 5) problems.

In this section, we describe the main principles of Algorithm Align. Let $3 \leq k < n-2$, $k \neq 4$, and let us consider any allowed configuration $\mathcal{C}$ for Problem P.

If $\mathcal{C}$ is symmetric, then Algorithm Align is executed by two symmetric robots. In this case, if only one of them actually moves, then the obtained configuration $\mathcal{C}'$ might be symmetric or adjacent to a symmetric configuration different from $\mathcal{C}$. One of our main results is the characterization of the symmetric configurations that may lead to these cases. Therefore, the procedures performed by Algorithm Align in case of symmetric configurations are designed in such a way that $\mathcal{C}'$ is asymmetric and adjacent only to $\mathcal{C}$. In other words, Align ensures that it is possible to univocally determine from $\mathcal{C}'$ the possible pending move. Then, the possible pending move is forced to be performed on $\mathcal{C}'$.



(a) Configuration $\mathcal{C}^a$.

(b) Configuration $\mathcal{C}^b$.

(c) Configuration $\mathcal{C}^c$.

Fig. 1: Configurations achieved at the end of Algorithm Align.

If $\mathcal{C}$ is asymmetric and not adjacent to a symmetric configuration, then Algorithm Asym can be executed without ambiguity.

If the initial allowed configuration is symmetric and k is even, Align eventually achieves either configuration $\mathcal{C}^b$ or $\mathcal{C}^c$, and the original type of symmetry is preserved, hence the obtained configuration is still allowed. If the configuration is asymmetric and k is even, then any of $\mathcal{C}^a$, $\mathcal{C}^b$, and $\mathcal{C}^c$ can be reached, if they are allowed. If k is odd, then the configuration reached is either $\mathcal{C}^a$ or $\mathcal{C}^b$,

if this latter is allowed. The general strategy of the algorithm is the following.

- If the configuration is symmetric, then ALIGN preserves the symmetry by performing a procedure that moves two symmetric robots in a way that, if only one of such robots actually moves, then the obtained configuration is guaranteed to be asymmetric and not adjacent to another symmetric configuration with respect to any other procedure that can be possibly performed by ALIGN. When k is odd, the symmetry is preserved until it can be safely broken by moving the unique robot lying on the axis of symmetry in an arbitrary direction.
- If the configuration is asymmetric, then always only one robot is permitted to move by ALIGN. First, the algorithm checks whether the asymmetric configuration is adjacent to some allowed symmetric configuration with respect to some procedure possibly performed by ALIGN. In this case, ALIGN forces the only possible pending move. We recall that the procedures performed on a symmetric configuration are designed in a way that the configuration obtained is not adjacent to any other symmetric configuration different from the correct one. Therefore, from an asymmetric configuration adjacent to an allowed symmetric one with respect to the procedures of ALIGN, the robot that has to move can be univocally determined and the original symmetry preserved. Note that, such behavior is performed even if the initial configuration is asymmetric. In this case, the configuration obtained after the move is symmetric and allowed, and the algorithm proceeds like in the case that the initial configuration was symmetric. In fact, as the robots are oblivious, they cannot distinguish the two cases.
- If an asymmetric configuration is not adjacent to any symmetric configuration with respect to any procedure of ALIGN, then algorithm ASYM in [14] is performed. Such algorithm, ensures that only one move is performed and the obtained configuration is always rigid, thus it is allowed.

We prove that the procedures performed by ALIGN always reduce the supermin (in lexicographical ordering) and that only allowed configurations are reached.

Our main result is stated in the next theorem whose proof is postponed in Section 6 for the sake of readability.

Theorem 1 *Let $3 \leq k < n-2$, $k \neq 4$, robots standing in an n-node ring forming an exclusive allowed configuration, Algorithm* ALIGN *eventually terminates achieving one exclusive allowed configuration among $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$.*

Before providing all the details concerning Algorithm ALIGN, the next two sections are devoted to the resolution of the gathering and the exclusive searching problems, respectively. The provided algorithms exploit the above theorem.

4 Gathering in a ring

In this section, we provide the full strategy for achieving the gathering under to Look-Compute-Move model and assuming the local-weak multiplicity detection. The idea is to allow to move always one single robot from each configuration so as to be sure which configuration will be reached next. The only exception to this ideal behavior comes from symmetric configurations where it is not possible to determine a priori whether one or two symmetric robots move concurrently. As described before, our algorithm ensures that all the reached configurations that might have a pending move can be always detected as asymmetric configurations with a unique pending move.

As defined in Section 2, the allowed configurations tackled by our gathering algorithm are all rigid configurations with a number of robots $k > 2$, and all symmetric configurations with $3 \leq k < n-4$, $k \neq 4$, not periodic and not admitting an axis of symmetry passing through two edges. Note that, as recalled in Table 1, our algorithm leaves open very few types of configurations. They are the configurations with only 4 robots different from those proven to be ungatherable in [12, 17], and configurations with $n-4 \leq k \leq n-1$ for which a gathering algorithm (if exists) would be very difficult to generalize to other configurations. Finally, configurations with $k = n$ are periodic and hence ungatherable [31] as well as configurations with $k = 2$ [31], while configurations with $k = 1$ do not require any algorithm.

We make use of procedure ALIGN to reach one of its output configurations: $\mathcal{C}^a = (0^{k-1}, 1, 0, 1^{n-k-1})$ with k even, $\mathcal{C}^b = (0^k, 1^{n-k})$, with k or n odd, $\mathcal{C}^c = (0^{\frac{k}{2}}, 1^j, 0^{\frac{k}{2}}, 1^{n-k-j})$, with k even and j or n odd. Actually, procedure ALIGN is invoked until either the obtained configuration is one of the three above, or if it is one of the configurations generated by Algorithm

GATHERING that we are going to describe. To this respect, we define two further types of configurations: $\mathcal{C}^d = (0^{k-1}, 1, 1, 0, 1^{n-k-2})$, and $\mathcal{C}^e = (0^{\frac{p}{2}-1}, 1, 0, 1, 0^{\frac{p}{2}-1}, 1^{n-p-1})$ with p even (and hence $k = p - 1$ odd).

Since to solve gathering we need to create a multiplicity, we need to handle configurations containing multiplicities. As we assume local-weak multiplicity detection, we remind that robots perceive a multiplicity only if they are part of it. So, they cannot deduce the actual total number of robots. Hence in this section k represents the number of occupied nodes and not the number of robots.

Moreover, we need to define three moves that Algorithm GATHERING applies:

- **COMPACT$_0(\mathcal{C})$:** Applied when $\mathcal{C}$ is of the form $(0^{\frac{k}{2}-i}, 1, 0^i, 1^j, 0^i, 1, 0^{\frac{k}{2}-i}, 1^{n-k-j-2})$, $0 \leq i < \frac{k}{2}$.
  - If n is even and $j > \frac{n-k-4}{2}$ or n is odd and j is even then
    - if $i = 0$ then a robot at node v_0 moves to node v_{n-1};
    - otherwise a robot at node $v_{\frac{k}{2}-i+1}$ moves to node $v_{\frac{k}{2}-i}$;
  - otherwise a robot at node $v_{\frac{k}{2}-i-1}$ moves to node $v_{\frac{k}{2}-i}$.
- **COMPACT$_1(\mathcal{C})$:** Applied when $\mathcal{C}$ is of the form $(0^{\frac{k-i}{2}}, 1, 0^i, 1, 0^{\frac{k-i}{2}}, 1^{n-k-2})$, $1 \leq i < k$, both k and i odd. A robot at node $v_{\frac{k-i}{2}-1}$ moves to node $v_{\frac{k-i}{2}}$;
- **COMPACT$_2(\mathcal{C})$:** Applied when in $\mathcal{C}$ there are only two nodes occupied (and exactly one is occupied by a multiplicity). The robot not belonging to the multiplicity moves to the other node occupied.

Theorem 2 *Let $3 \leq k < n - 4$, $k \neq 4$ robots, forming an allowed configuration in an n-node ring, Algorithm* GATHERING *achieves the gathering.*

Proof Algorithm GATHERING is structured in a way that procedure ALIGN is invoked only at the end (line 23), that is once it is sure that the current configuration does not belong to those directly managed for gathering.

If the initial configuration has both k and n even, then ALIGN either reaches configuration $\mathcal{C}^a$, or $\mathcal{C}^c$ with j odd. In the former case, GATHERING leads to $\mathcal{C}^d = (0^{k-1}, 1, 1, 0, 1^{n-k-2})$ (lines 3-4). As $k < n - 4$, then $\mathcal{C}^d$ is asymmetric and it is not adjacent to any possible symmetric configuration with respect to any procedure of GATHERING. From $\mathcal{C}^d$, GATHERING makes v_0 move toward v_1 (lines 5-6), hence creating a multiplicity, and still obtaining a configuration of type $\mathcal{C}^d$. This process is repeated until only two nodes remain occupied. At this point, only one of the two occupied nodes contains a multiplicity, while the other contains one single robot. The single robot will be the only one permitted to move toward the other occupied node by means of Procedure COMPACT$_2$ (lines 1-2) until the gathering is accomplished.

In the latter case, that is, from $\mathcal{C}^c$ with j odd, GATHERING leads to configuration $\mathcal{C}^c$ with $j = 1$. This is achieved by alternately iterating procedure COMPACT$_0$ (lines 7-8) with the call to procedure ALIGN. As $\mathcal{C}$ is symmetric, COMPACT$_0$ permits two robots to move. If both robots move synchronously, the resulting configuration is of the form $\mathcal{C}' = (0^{\frac{k}{2}-i-1}, 1, 0^{i+1}, 1^j, 0^{i+1}, 1, 0^{\frac{k}{2}-i-1}, 1^{n-k-j-2})$. If only one robot moves, the obtained configuration $(0^{\frac{k}{2}-i-1}, 1, 0^{i+1}, 1^j, 0^i, 1, 0^{\frac{k}{2}-i}, 1^{n-k-j-2})$ is asymmetric and not adjacent to any other symmetric configuration with a smaller supermin, and hence $\mathcal{C}'$ can be easily obtained (lines 9-10).

Once $\mathcal{C}^c$ with $j = 1$ is reached, again COMPACT$_0$ is applied (lines 7-8). In fact, from the definition of COMPACT$_0$ by considering $i = 0$ and $j = n - k - 3$, the two robots neighboring the empty node located in between the two sequences of aligned robots are allowed to move toward such an empty node to form a multiplicity. If both the permitted robots move, a symmetric configuration of type $\mathcal{C}^e$, with $v_{\frac{k}{2}}$ being a multiplicity, is reached. This will be discussed later in the case of symmetric configurations with odd k. If only one robot moves, configuration $(0^{\frac{k}{2}-1}, 1, 0^{\frac{k}{2}+1}, 1^{n-k-1})$ is reached. As k is even and $4 < k < n - 4$, it follows that such a configuration is asymmetric and can only be transformed into one of type $\mathcal{C}^e$ (by decreasing k) again by lines 9-10.

If k is even and n is odd, we can consider that ALIGN has reached either a configuration of type $\mathcal{C}^b$ or $\mathcal{C}^c$ with either j or $n - k - j$ odd. In fact, if a configuration of type $\mathcal{C}^a$ with k even and n odd is given as input to algorithm GATHERING, it is treated (at lines 13-14) as though it has been obtained from one of type $\mathcal{C}^b$ (lines 11-12), where only one of the two robots allowed to move by means of COMPACT$_0$ has performed its movement. In fact, from the definition of COMPACT$_0$ by considering $i = 0$ and $j = n - k - 2$ (that is j is odd), the two robots neighboring empty nodes in a configuration of type $\mathcal{C}^b$ are allowed to move toward the empty nodes. For managing configurations of type

Algorithm: GATHERING
Input: Allowed configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$

1 **if** $k = 2$ **then**
2 ⌊ COMPACT$_2(\mathcal{C})$ and exit;
3 **if** *k is even and n is even and $\mathcal{C}$ is of type $\mathcal{C}^a$* **then**
4 ⌊ The robot at v_k moves to v_{k+1} and exit;
5 **if** *$\mathcal{C}$ is of type $\mathcal{C}^d$* **then**
6 ⌊ The robot at v_0 moves to v_1, and exit;
7 **if** *k is even and n is even and $\mathcal{C}$ is of type $\mathcal{C}^c$ with j odd* **then**
8 ⌊ COMPACT$_0(\mathcal{C})$ and exit;
9 **if** *k is even and n is even and $\mathcal{C}$ could have been obtained from a configuration of type $\mathcal{C}^c$ with j odd by applying* COMPACT$_0$ **then**
10 ⌊ Move the unique robot that can re-establish the assumed symmetry while decreasing either the current supermin or k and exit;
11 **if** *k is even and n is odd and $\mathcal{C}$ is of type $\mathcal{C}^b$ or $\mathcal{C}^c$ with either j or $n-j-k$ odd* **then**
12 ⌊ COMPACT$_0(\mathcal{C})$ and exit;
13 **if** *k is even and n is odd and $\mathcal{C}$ could have been obtained from a configuration of type $\mathcal{C}^b$ or $\mathcal{C}^c$, by applying* COMPACT$_0$ **then**
14 ⌊ Move the unique robot that can re-establish the assumed symmetry while decreasing either the current supermin or k and exit;
15 **if** *k is odd and $\mathcal{C}$ is of type $\mathcal{C}^b$* **then**
16 ⌊ The robots at $v_{\frac{k-1}{2}-1}$ and $v_{\frac{k-1}{2}+1}$ move toward $v_{\frac{k-1}{2}}$, and exit;
17 **if** *k is odd and $\mathcal{C}$ could have been obtained from a configuration of type $\mathcal{C}^b$* **then**
18 ⌊ Move the symmetrical robot to reach a configuration of type $\mathcal{C}^e$, and exit;
19 **if** *$\mathcal{C}$ is of type $\mathcal{C}^e$* **then**
20 ⌊ COMPACT$_1(\mathcal{C})$ and exit;
21 **if** *k is even and $\mathcal{C}$ could have been obtained from a configuration of type $\mathcal{C}^e$ by applying* COMPACT$_1$ **then**
22 ⌊ Move the symmetrical robot to obtain a configuration of type $\mathcal{C}^e$ while decreasing k and exit;
23 Apply ALIGN;

Fig. 2: Algorithm GATHERING.

$\mathcal{C}^b$ or $\mathcal{C}^c$, GATHERING behaves as above but creating the multiplicity at the central node of the only odd sequence of consecutive empty nodes among j and $n-k-j$ (lines 11-14). Eventually, GATHERING achieves a configuration of type $\mathcal{C}^e$. Again, this will be discussed later in the case of symmetric configurations with odd k. Note that, this case is similar to the technique presented in [29] where the solved configurations are only those with k even and n odd.

If k is odd, either the configuration is of type $\mathcal{C}^b$ or it will be of type $\mathcal{C}^b$ within two applications of ALIGN. To see why, note that in the case of odd k, ALIGN will lead to $\mathcal{C}^b$ or $\mathcal{C}^a$ with odd k. If $\mathcal{C}^a$ with odd k is the input to algorithm GATHERING, then it is managed by ALIGN and leads to a configuration of type $\mathcal{C}^b$ since ALIGN always reduces the supermin. For managing configurations of type $\mathcal{C}^b$, the used technique is similar to that presented in [28] where the solved configurations are only those with k odd. From $\mathcal{C}^b$, GATHERING permits robots at $v_{\frac{k-1}{2}-1}$ and $v_{\frac{k-1}{2}+1}$ to move toward $v_{\frac{k-1}{2}}$ (lines 15-16). If only one robot actually moves, configuration $(0^{\frac{k'}{2}-1}, 1, 0^{\frac{k'}{2}+1}, 1^{n-k'-1})$ is achieved with $k' = k-1$. By the parity of k', a configuration of type $\mathcal{C}^e$ is achieved subsequently (lines 17-18). If both robots move synchronously, again a configuration of type $\mathcal{C}^e$ is reached. From here, GATHERING performs Procedure COMPACT$_1$ (lines 19-20). As configurations of type $\mathcal{C}^e$ are symmetric, COMPACT$_1$ permits two robots to move. If both move synchronously, the resulting configuration is similar to one of type $\mathcal{C}^e$ but with a larger middle interval of 0's. If only one robot moves, as before, the obtained configuration is asymmetric and not adjacent to any other symmetric configuration, and the possible pending move can be easily forced (lines 21-22). From there, procedure ALIGN is invoked until a new configuration of type $\mathcal{C}^b$ will be reached, but with a smaller k with respect to the previous one.

Eventually, this process terminates with only one occupied node without pending moves, that is, gathering is achieved. □

5 Exclusive searching in a ring

In this section, we present an algorithm that allows a team of robots to exclusively search a ring starting from any allowed configuration, see Table 2. We also give some partial impossibility results in the case of periodic starting configuration.

As defined in Section 2, the allowed configurations tackled by our searching algorithm are all rigid exclusive configurations where $4 < k \leq n-3$ where $n > 9$ and $(n,k) \neq (10,5)$. Moreover, the symmetric allowed configurations are the exclusive configuration with a single axis of symmetry, $4 < k < n-2$ and $n > 9$ and $(n;k) \notin \{(10,5),(10,6)\}$, and, if the axis does pass through an empty node, then k must be odd.

First, we recall some known results concerning the exclusive searching of a ring in the Look-Compute-Move model. Let us start with some impossibility results. If k is even and there exists an axis of symmetry passing through an empty node, exclusive searching is clearly infeasible because a synchronous execution of any algorithm either cause a multiplicity in the node lying on the axis or does not allow to search the edges incident to such a node. In [14], it is shown that, for any starting configuration, it is not possible to search an n-node ring using k robots if $n \leq 9$, $k \leq 3$, or $k \geq n-2$.

On the other hand, there exists an algorithm that allows $5 \leq k \leq n-3$ robots to search exclusively a ring with $n \geq 10$ nodes (except for $(k,n) = (5,10)$), for rigid initial configurations [14]. Here, we improve over this algorithm by addressing also allowed aperiodic symmetric configurations.

The main result of this section is the design of Algorithm Search-Ring.

Theorem 3 *Algorithm* Search-Ring *exclusively (and perpetually) searches the ring starting from any allowed configuration.*

We first describe the general behavior of Algorithm Search-Ring whose pseudo-code is described in Figure 6. Before actually searching the ring, Algorithm Search-Ring needs the robots to achieve particular configuration. For this purpose, Algorithm Search-Ring first applies one of the sub-procedures Compact-Align and Break-Symmetry, that are described in Figures 3 and 4 respectively. Each of these sub-procedures uses Algorithm Align presented in Section 6. Procedure Compact-Align is used only if k and n are even, and Procedure Break-Symmetry may be used if k is odd.

5.1 Algorithm Compact-Align

We now define an algorithm that complements Algorithm Align given in Section 6. In detail, Algorithm Compact-Align is applied when k and n are even, until one of the configurations $\mathcal{C}^a$ (all robots but one occupy consecutive nodes) or $\mathcal{C}^b$ (all robots occupy consecutive nodes) is achieved.

Set $\mathcal{R}$ of configurations. Let us first define some sets of particular configurations. Roughly, in these configurations, robots are divided into two pairs of segments of consecutive nodes and, for each pair, the two segments of the pair are separated by one unique empty node. More formally, let n and k be even. For any $0 \leq a < b$ with a and $b = n-k-a-2$ even (in particular $b > 1$), let us define $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ as the set of all configurations $\mathcal{C} = (u_0, \cdots, u_{n-1})$ with the following form:

- $\mathcal{R}_1(a,b,c) = (0^{k/2-c}, 1, 0^c, 1^a, 0^c, 1, 0^{k/2-c}, 1^b)$, where $0 \leq c < k/2$. Note that $\mathcal{R}_1(a,b,0) \in \mathcal{C}^c$. Moreover, any configuration $\mathcal{R}_1(a,b,c)$ is symmetric with one unique axis (because $a < b$) and this axis does not pass through an empty node (because a and b are even). Moreover, $u_{k/2-c-1}$ and u_{n-1-b} can be identified because $a < b$ and $b > 1$. Let $\mathcal{R}_1$ be the set of all such configurations $\mathcal{R}_1(a,b,c)$ with $0 \leq a < b$, a and $b = n-k-a-2$ even, and $0 \leq c < k/2$.
- $\mathcal{R}_2(a,b,c) = (0^{k/2-c-1}, 1, 0^{c+1}, 1^a, 0^c, 1, 0^{k/2-c}, 1^b)$, where $0 \leq c < k/2$. Such a configuration is asymmetric and $u_{n-b-(k/2-c)}$ can be identified. Note also that $\mathcal{R}_2(0,b,k/2-1) \in \mathcal{C}^a$. Let $\mathcal{R}_2$ be the set of all such configurations $\mathcal{R}_2(a,b,c)$ with $0 \leq a < b$, a and $b = n-k-a-2$ even, and $0 \leq c < k/2$.

Recall that any allowed configuration in $\mathcal{C}^c$ has the following form: $(0^{\frac{k}{2}}, 1^j, 0^{\frac{k}{2}}, 1^{n-k-j}) = \mathcal{R}_1(j-2, n-k-j, 0)$, with $0 < j < \frac{n-k}{2}$ and j even.

Algorithm Compact-Align. Algorithm Compact-Align first applies Align. Then, either a configuration $\mathcal{C}^a$ or $\mathcal{C}^b$ is achieved, in which case we are done, or a configuration in the set $\mathcal{R}$ is achieved. Since every allowed configuration in $\mathcal{C}^c$ (the robots are divided into two segments of consecutive nodes) belongs to $\mathcal{R}$, the specifications of Align (it achieves either $\mathcal{C}^a$ or $\mathcal{C}^b$ or $\mathcal{C}^c$ and it may reach $\mathcal{C}^c$ only if k and n are even) ensure that such a configuration is eventually reached. Finally, from any configuration in $\mathcal{R}$, Algorithm Compact-Align allows the robots to achieve either $\mathcal{C}^a$ or $\mathcal{C}^b$.

If $\mathcal{C} \in \mathcal{R}_1(a,b,c)$ for some a, b, and c, then COMPACT-ALIGN moves the robot at $u_{k/2-c-1}$ to $u_{k/2-c}$. Otherwise, if $\mathcal{C} \in \mathcal{R}_2(a,b,c)$ for some a, b, and c, then it moves the robot at $u_{n-b-(k/2-c)}$ to $u_{n-b-(k/2-c-1)}$, otherwise it applies ALIGN.

Since a and b are even, then $b \geq a+2$, and therefore we can check that any configuration $\mathcal{C}$ in $\mathcal{R}$ is not adjacent to any other allowed symmetric configuration. Indeed, if $\mathcal{C}$ is adjacent to an allowed and symmetric configuration, then the axis must pass through the unique long segment of at least $b-1$ consecutive 1's, and it is easy to check that such a configuration would not be allowed because b is even. Therefore, there is no conflict between the procedures described above if $\mathcal{C} \in \mathcal{R}$ and the ones permitted by ALIGN when $\mathcal{C} \notin \mathcal{R}$.

5.2 Algorithm BREAK-SYMMETRY

In the following, we give an algorithm that allows an odd number k of robots to eventually reach an asymmetric configuration. More precisely, the algorithm first applies Algorithm ALIGN. Then, when all k robots are occupying consecutive nodes, they move to reach a symmetric configuration where one robot is on the axis and has its two neighbors that are empty. The robot on the axis moves to one of its neighbors, breaking the symmetry.

Let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ be the set of all configurations $\mathcal{C} = (u_0, \cdots, u_{n-1})$ with the following form. Roughly, robots are divided into three segments of consecutive nodes such that the central segment is separated from each other segment by one empty node. Moreover, the lengths of the two non-central segments differ by at most one. More formally:

- $\mathcal{B}_1(\ell) = (0^\ell, 1, 0^{k-2\ell}, 1, 0^\ell, 1^{n-k-2})$, where $0 \leq \ell \leq \lfloor k/2 \rfloor$. Moreover, any configuration $\mathcal{B}_1(\ell)$ is symmetric with one unique axis and nodes $u_{\ell+1}$ and $u_{k-\ell+1}$ can be univocally identified. Let $\mathcal{B}_1 = \bigcup_{0 \leq \ell \leq \lfloor k/2 \rfloor} \{\mathcal{B}_1(\ell)\}$.
- $\mathcal{B}_2(\ell) = (0^{\ell-1}, 1, 0^{k-2\ell+1}, 1, 0^\ell, 1^{n-k-2})$, where $0 < \ell \leq \lfloor k/2 \rfloor$. Such a configuration is asymmetric and u_ℓ can be univocally identified. Let $\mathcal{B}_2 = \bigcup_{0 < \ell \leq \lfloor k/2 \rfloor} \{\mathcal{B}_2(\ell)\}$.

When a configuration $\mathcal{C}$ is in $\mathcal{B}_1(\ell)$ for some ℓ, then BREAK-SYMMETRY moves the robot at $u_{\ell+1}$ to u_ℓ. When $\mathcal{C}$ is in $\mathcal{B}_1(\ell)$ for some ℓ, then BREAK-SYMMETRY moves robot at u_ℓ to $u_{\ell-1}$. Eventually, configuration $\mathcal{C} = (0^{\lfloor k/2 \rfloor}, 1, 0, 1, 0^{\lfloor k/2 \rfloor}, 1^{n-k-2})$ is reached. At this point the robot at $u_{\lfloor k/2 \rfloor+1}$ moves to $u_{\lfloor k/2 \rfloor}$ (or arbitrarily to $u_{\lfloor k/2 \rfloor+2}$). At this point the obtained configuration is asymmetric and applying Algorithm ALIGN leads to configuration $\mathcal{C}^a = (0^{k-1}, 1, 0, 1^{n-k-1})$. Finally, the robot at u_k moves to u_{k+1}. The obtained configuration is suitable to be used in the algorithm of [14] for graph searching. As any asymmetric configuration in $\mathcal{B}_2$ is not adjacent to any symmetric configuration not in $\mathcal{B}_1$, there are no conflicts between the procedures of ALIGN, those of the algorithm in [14] and those of BREAK-SYMMETRY.

5.3 General description of Algorithm SEARCH-RING

Finally, we are ready to describe the general behavior of Algorithm SEARCH-RING in more details.

The algorithm first checks whether $k = n-3$ or if n is odd and k is even. In the affirmative case, any allowed configuration must be asymmetric, and therefore the algorithm of [14] can be applied and the ring is searched.

In the other cases, the algorithm proceeds in two phases. Phase 1 consists in achieving a configuration from which the search will be performed. Phase 2 consists in actually searching the ring.

If k is odd, Phase 1 consists in using Algorithm BREAK-SYMMETRY to break the potential symmetry (Line 8 of Algorithm SEARCH-RING) and then, Phase 2 executes the algorithm of [14] (Line 6 of Algorithm SEARCH-RING). Each of these configurations used during the searching phase of the algorithm of [14] are asymmetric and are not adjacent to any symmetric configuration reached by Algorithm BREAK-SYMMETRY. Therefore, there is no ambiguity (no pending move) when a robot recognizes such a configuration.

If n and k are even, we may be in allowed symmetric configurations and therefore the SEARCH-RING proceeds in two phases. Phase 1 (Line 23 of Algorithm SEARCH-RING) consists in applying Algorithm COMPACT-ALIGN until one of the configurations in $\mathcal{A}$ (see the definition below) is achieved. This is guaranteed by the fact that both $\mathcal{C}^a$ and $\mathcal{C}^b$ belong to $\mathcal{A}$. Then, the algorithm proceeds to Phase 2 (Lines 10-20 of Algorithm SEARCH-RING) which actually performs the searching.

Set $\mathcal{A}$ of configurations. We now define the set $\mathcal{A}$ of configurations required to define the algorithm for Phase 2. These configurations are depicted in Figure 5. We consider the following hypothesis: $n-k$ is even, $n-k \geq 4$, $k \geq 6$, $n \geq 10$ and, if $k = 6$ then $n \geq 11$. The set $\mathcal{A}$ is defined as the set

Procedure: COMPACT-ALIGN
Input: Allowed configuration $\mathcal{C} = (u_0, \cdots, u_{n-1})$, with an even number of robots.

1 **if** $\mathcal{C} \in \mathcal{R}$ **then**
2 **if** $\mathcal{C} = (0^{k/2-c}, 1, 0^c, 1^a, 0^c, 1, 0^{k/2-c}, 1^b)$ **then** // $\mathcal{C} \in \mathcal{R}_1$
3 Robot at $u_{k/2-c-1}$ moves to $u_{k/2-c}$; // Two symmetrical moves
4 **if** $\mathcal{C} = (0^{k/2-c-1}, 1, 0^{c+1}, 1^a, 0^c, 1, 0^{k/2-c}, 1^b)$ **then** // $\mathcal{C} \in \mathcal{R}_2$
5 Robot at $u_{n-b-(k/2-c)}$ moves to $u_{n-b-(k/2-c-1)}$; // unique move performed
6 **else**
7 Apply ALIGN;

Fig. 3: Algorithm COMPACT-ALIGN.

Procedure: BREAK-SYMMETRY
Input: Exclusive configuration $\mathcal{C} = (u_0, \cdots, u_{n-1})$, with $\sum_i u_i = k$ robots, such that k is odd and $\mathcal{C}$ has at most one axis of symmetry and, if any, this axis does not pass through an empty node.

1 **if** $\mathcal{C} \in \mathcal{B}$ **then**
2 **if** $\mathcal{C} = (0^{\lfloor k/2 \rfloor}, 1, 0, 1, 0^{\lfloor k/2 \rfloor}, 1^{n-k-2})$ **then** // $\mathcal{C} = \mathcal{B}_1(\lfloor k/2 \rfloor)$
3 Robot at $u_{\lfloor k/2 \rfloor+1}$ moves to $u_{\lfloor k/2 \rfloor}$ (or symmetrically to $u_{\lfloor k/2 \rfloor+2}$); // symmetry is broken
4 **if** $\mathcal{C} = (0^\ell, 1, 0^{k-2\ell}, 1, 0^\ell, 1^{n-k-2})$ *where* $0 \le \ell < \lfloor k/2 \rfloor$ **then** // $\mathcal{C} \in \mathcal{B}_1$
5 Robot at $u_{\ell+1}$ moves to u_ℓ; // Two symmetrical moves
6 **if** $\mathcal{C} = (0^{\ell-1}, 1, 0^{k-2\ell+1}, 1, 0^\ell, 1^{n-k-2})$ *where* $0 < \ell \le \lfloor k/2 \rfloor$ **then** // $\mathcal{C} \in \mathcal{B}_2$
7 Robot at u_ℓ moves to $u_{\ell-1}$; // unique move performed
8 **else**
9 Apply ALIGN;

Fig. 4: Algorithm BREAK-SYMMETRY.

of all configurations $\mathcal{C} = (u_0, \cdots, u_{n-1})$ with the following forms:

$\mathcal{A}$-a(ℓ) $= (0^{k-2}, 1^\ell, 0, 1^{n-2\ell-k}, 0, 1^\ell)$, $0 \le \ell \le (n-k)/2$. Note that $\mathcal{A}$-a(0) $= \mathcal{C}^b$.
In this case, $\mathcal{C}$ is symmetric with a unique axis because $k - 2 > 1$. This axis does not pass through an empty node because $n - k$ is even. Clearly, nodes $u_{k-2+\ell}$ and $u_{n-1-\ell}$ can be identified as occupied and adjacent to one (case $\ell = 0$) or two (case $0 < \ell < (n-k)/2$) empty nodes, or (case $\ell = (n-k)/2$) they form the unique (because $k \ge 5$) segment of exactly two consecutive occupied nodes.

$\mathcal{A}$-b(ℓ) $= (0^{k-2}, 1^\ell, 0, 1^{n-2\ell-k-1}, 0, 1^{\ell+1})$, $0 \le \ell < (n-k)/2$.
In this case, $\mathcal{C}$ is asymmetric for any $0 \le \ell \le (n-k)/2$. In particular, if $\ell = 0$, it is asymmetric because $n - k \ge 4$. Then, $u_{k-2+\ell}$ can be identified.

$\mathcal{A}$-c $= (0^{k-3}, 1, 0, 1^{(n-k)/2-1}, 0, 0, 1^{(n-k)/2})$.
In this case, $\mathcal{C}$ is asymmetric, because $k \ge 6$ and $n - k \ge 4$. Then, u_0 can be identified.

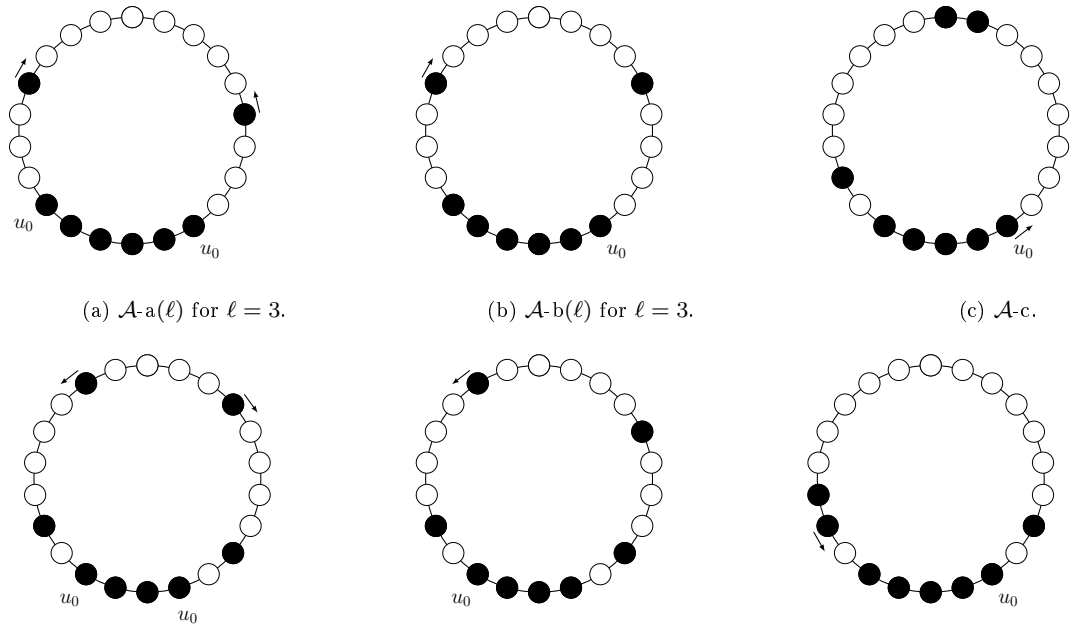
$\mathcal{A}$-d(ℓ) $= (0^{k-4}, 1, 0, 1^{(n-k)/2-1-\ell}, 0, 1^{2\ell}, 0, 1^{(n-k)/2-\ell-1}, 0, 1)$, $0 \le \ell \le (n-k)/2 - 1$.
In this case, $\mathcal{C}$ is symmetric with one unique axis not passing through an empty node. Indeed, it is easy to check if $k \ne 6$. If $k = 6$ and $\ell = 0$, it is true because $n > 10$.
If $\ell > 0$, $u_{(n+k)/2-3-\ell}$ and $u_{(n+k)/2-2+\ell}$ can be identified since $(u_0, \cdots, u_{k-5})$ is the single segment with at least two occupied nodes.
If $\ell = 0$, $u_{(n+k)/2-3}$ and $u_{(n+k)/2-2}$ can be identified as the single segment of two occupied nodes (if $k > 6$) and as the single segment of two occupied nodes adjacent to segments of more than one empty node (if $k = 6$ and $n > 10$).

$\mathcal{A}$-e(ℓ) $= (0^{k-4}, 1, 0, 1^{(n-k)/2-1-\ell}, 0, 1^{2\ell+1}, 0, 1^{(n-k)/2-\ell-2}, 0, 1)$, $0 \le \ell \le (n-k)/2 - 2$.
In this case, $\mathcal{C}$ is asymmetric (this is true in particular, when $\ell = 0$, because if $k = 6$ then $n > 10$) Then, $u_{(n+k)/2-3-\ell}$ can be identified.

$\mathcal{A}$-f $= (0^{k-3}, 1, 0, 0, 1^{n-k-2}, 0, 1)$
In this case, $\mathcal{C}$ is asymmetric because $k \ge 5$ and $n - k \ge 4$. Then, u_{k-2} can be identified.

Phase 2 *of Algorithm* SEARCH-RING*: searching the ring.* If $k = n-3$ or n or k is odd, the searching is done using the Algorithm of [14] and the correctness follows.

Hence, we only need to consider the case when k and n are even. In such case, Phase 2 of Algorithm SEARCH-RING proceeds as follows. All



(a) $\mathcal{A}$-a(ℓ) for $\ell = 3$. (b) $\mathcal{A}$-b(ℓ) for $\ell = 3$. (c) $\mathcal{A}$-c.

(d) $\mathcal{A}$-d(ℓ) for $\ell = 2$. (e) $\mathcal{A}$-e(ℓ) for $\ell = 2$. (f) $\mathcal{A}$-f

Fig. 5: Configurations reached by Phase 2 of Algorithm Search-Ring when k and n are even. In this example, $n = 22$ and $k = 8$. Robots are depicted in black and the arrows represent the pending moves. In the symmetric cases, two nodes can be identified as u_0.

robots are aligned on consecutive nodes (configuration $\mathcal{A}$-a(0) $= \mathcal{C}^b$). Then, each of the two robots X and Y at the ends of this segment moves (one clockwise and the other anti-clockwise) to meet on the two adjacent nodes opposite to the occupied segment (alternating between configurations $\mathcal{A}$-$a(\ell)$ and $\mathcal{A}$-$b(\ell)$ for $\ell = 0 \cdots (n-k)/2$, and stopping at $\mathcal{A}$-$a((n-k)/2)$). Then, the two robots X' and Y' occupying the ends of the "long" occupied segment move to their empty neighbor ($\mathcal{A}$-c and $\mathcal{A}$-d(0)). These moves are two indicate to X and Y that it is time to go back toward the "long" segment, and that is what happens (alternating between configurations $\mathcal{A}$-$d(\ell)$ and $\mathcal{A}$-$e(\ell)$ for $\ell = 0 \cdots (n-k)/2 - 2$, and stopping at $\mathcal{A}$-$d((n-k)/2-1)$). Finally, when X is adjacent to X' and Y is adjacent to Y', X' and Y' move to their empty neighbor (passing through the configuration $\mathcal{A}$-f) such that they re-integrate the segment. Then, Configuration $\mathcal{A}$-a(1) is achieved and the process is repeated perpetually.

It is easy to check that such a sequence of performed moves actually searches the ring. Indeed, after having reached the configuration $\mathcal{A}$-$d((n-k)/2-1)$, configurations $\mathcal{A}$-f and then $\mathcal{A}$-$a(1)$ are reached, ensuring that all edges between the two isolated robots (on the side of the "long" segment of robots) are clear. Then, the two isolated robots goes along the ring until they occupy adjacent nodes. At this step, all edges are clear. Moreover, by definition of the configurations met during the process (configurations in $\mathcal{A}$), there is no ambiguity. In Figure 6, $\mathcal{O}$ denotes the set of configurations used during the searching phase of the Algorithm of [14].

The distinct configurations that can be achieved in Phase 2 are the ones in $\mathcal{A}$ and can be characterized succinctly such that they are pairwise distinguishable without ambiguity. Moreover, each of those configurations is either asymmetric and only one (identifiable) robot can move, or it is symmetric with one unique axis of symmetry and two (identifiable) symmetric robots move. In the latter case, when only one of these symmetric robots moves, then we reach an asymmetric configuration where the only robot permitted to move is the other one (i.e., the possible pending move and the permitted move coincide). Therefore there is never ambiguity in the choice of the robot(s) that must move.

The validity of algorithms for Phase 2 and the fact that they actually search the ring are easy to obtain. Therefore, to prove the correctness of Algorithm Search-Ring, it will be sufficient to prove

Procedure: SEARCH-RING
Input: Exclusive configuration $\mathcal{C} = (u_0, \cdots, u_{n-1})$, with $\sum_i u_i = k$ robots, $k \geq 6$ and $n \geq 10$ and $(k, n) \neq (6, 10)$ and $n - k \geq 3$, and such that $\mathcal{C}$ has at most one axis of symmetry and, if any and if k is even, this axis does not pass through an empty node.

if $k = n - 3$ *or (*n *is odd and* k *even)* **then**
  Apply Algorithm of [14]
else
  if k *is odd* **then**
    if $\mathcal{C} \in \mathcal{O}$ **then** // $\mathcal{C}$ is asymmetric
      Apply Algorithm of [14]
    else
      Apply BREAK-SYMMETRY($\mathcal{C}$)
  else
    if $\mathcal{C} \in \mathcal{A}$ **then**
      if $\mathcal{C} = (0^{k-2}, 1^{\ell}, 0, 1^{n-2\ell-k}, 0, 1^{\ell})$ *with* $0 \leq \ell \leq (n-k)/2$
      OR $\mathcal{C} = (0^{k-2}, 1^{\ell}, 0, 1^{n-2\ell-k-1}, 0, 1^{\ell+1})$ *with* $0 \leq \ell < (n-k)/2$ **then** // $\mathcal{C} \in \mathcal{A}$-a OR $\mathcal{C} \in \mathcal{A}$-b
        The robot at $u_{k-2+\ell}$ moves to $u_{k-2+\ell+1}$; // two symmetrical moves if $\mathcal{C} \in \mathcal{A}$-a
      if $\mathcal{C} = (0^{k-2}, 1^{(n-k)/2}, 0, 0, 1^{(n-k)/2})$
      OR $\mathcal{C} = (0^{k-3}, 1, 0, 1^{(n-k)/2-1}, 0, 0, 1^{(n-k)/2})$ **then** // $\mathcal{C} = \mathcal{A}$-a$((n-k)/2)$ OR $\mathcal{C} = \mathcal{A}$-c
        The robot at u_0 move to u_{n-1}; // two symmetrical moves if $\mathcal{C} \in \mathcal{A}$-a
      if $\mathcal{C} = (0^{k-4}, 1, 0, 1^{(n-k)/2-1-\ell}, 0, 1^{2\ell}, 0, 1^{(n-k)/2-\ell-1}, 0, 1)$ *with* $0 \leq \ell \leq (n-k)/2 - 1$
      OR $\mathcal{C} = (0^{k-4}, 1, 0, 1^{(n-k)/2-1-\ell}, 0, 1^{2\ell+1}, 0, 1^{(n-k)/2-\ell-2}, 0, 1)$ *with* $0 \leq \ell \leq (n-k)/2 - 2$
      then // $\mathcal{C} \in \mathcal{A}$-d OR $\mathcal{C} \in \mathcal{A}$-e
        The robot at $u_{(n+k)/2-3-\ell}$ moves to $u_{(n+k)/2-4-\ell}$; // two symmetrical moves if $\mathcal{C} \in \mathcal{A}$-d
      if $\mathcal{C} = (0^{k-4}, 1, 0, 0, 1^{n-k-2}, 0, 0, 1)$
      OR $\mathcal{C} = (0^{k-3}, 1, 0, 1^{n-k-2}, 0, 0, 1)$ **then** // $\mathcal{C} = \mathcal{A}$-d$((n-k)/2 - 1)$ OR $\mathcal{C} = \mathcal{A}$-f
        The robot at u_{n-2} move to u_{n-1}; // two symmetrical moves if $\mathcal{C} \in \mathcal{A}$-d
      else
        Apply COMPACT-ALIGN;

Fig. 6: Algorithm SEARCH-RING.

that Phase 1 and Phase 2 are not in conflict (i.e., that robots can decide which phase to proceed). It is enough to note that any configuration in $\mathcal{A}$ is not adjacent to any symmetric configuration not in $\mathcal{A}$.

5.4 Impossibility for periodic configurations

To conclude this section, we give partial results on periodic configurations. More precisely, we describe some periodic configurations for which we prove the graph searching problem to be infeasible. Since these configurations have not the forbidden symmetry (one empty node on an axis of symmetry and k even), it shows that periodicity actually introduces new impossibility results.

A *period* S of a periodic configuration $\mathcal{C}$ is a sequence, with minimum length, such that $\mathcal{C} = S^q$ for some $q > 1$. We say that such a configuration $\mathcal{C}$ is *q-periodic*. Note that, any period of a periodic configuration has the same number of empty nodes. For any $i > 0$, let $\mathcal{P}_i$ be the set of all periodic configurations with exactly i empty nodes in any period.

Theorem 4 *For any configuration $\mathcal{C} \in \mathcal{P}_1 \cup \mathcal{P}_2$, there is no algorithm that solves the graph searching problem starting from $\mathcal{C}$.*

Proof Let $q > 1$ and $\mathcal{C}$ be q-periodic. Let $p \geq 1$ be the number of robots in each period of $\mathcal{C}$. The ring has size $n = q(p + i)$, where $i \geq 1$ is the number of empty nodes in a period, and there are $k = pq$ robots.

First, let us assume that $\mathcal{C} = (0^p, 1)^q \in \mathcal{P}_1$. If q or p is even, then k is even and there is an axis of symmetry passing through an empty node. Hence, by previous results, no algorithm can solve the graph searching problem starting from $\mathcal{C}$. Let us assume that p and q are odd. Because k is odd and $q > 1$, we are not in the previous impossible cases. However, any two robots adjacent to the same empty node have exactly the same view. Therefore, any move in this configuration will lead to a multiplicity. Thus, it is impossible to solve the exclusive searching problem starting from $\mathcal{C}$.

Now, let us assume that $\mathcal{C} \in \mathcal{P}_2$., i.e., $\mathcal{C} \in \{\mathcal{Q}_j = (1, 0^{p-j}, 1, 0^j)^q \mid 0 \leq j \leq \lfloor p/2 \rfloor\}$. Note that each period consists of two segments of consecutive robots (but for $j = 0$ where there is only one

segment). The intuition is that any algorithm has only one possible action for any fixed $j \leq \lfloor p/2 \rfloor$. Either the two robots occupying the ends of the left segment (the one between the second node and the $p-j+1^{th}$ node of the period) have to move, or the two robots at the end of the right segment have to move. Indeed, if none of these moves is ever performed, the ring cannot be searched. On the other hand, if the algorithm allows the four robots to move, it results in multiplicities. Hence, for any $j \leq \lfloor p/2 \rfloor$, the algorithm must be of the left type or of the right type. Roughly, in what follows, we show that if the allowed moves are all of the same type, then the algorithm achieves a configuration in $\mathcal{P}_1$ and fails by previous paragraph. Otherwise, there is some j such that the algorithm performs the left move for j and the right move for $j+1$. In this case, the adversary can schedule the moves in order to force a multiplicity.

Let us prove the result more formally.

Let $0 \leq j \leq \lfloor p/2 \rfloor$. We note $(u_0, \cdots, u_{n-1})$ the nodes of the ring such that the representation from u_0 is $\mathcal{C} = (1, 0^{p-j}, 1, 0^j)^q$. Let us consider any algorithm $\mathcal{A}$ for solving the exclusive searching problem. We say that $\mathcal{A}$ is *j-left* if, in configuration $(1, 0^{p-j}, 1, 0^j)^q$, the robots that must move are those at u_ℓ, where $\ell = 1 \bmod q$ and $\ell = p - j \bmod q$. $\mathcal{A}$ is *j-right* if, in configuration $(1, 0^{p-j}, 1, 0^j)^q$, the robots that must move are those at u_ℓ, where $\ell = p+1 \bmod q$ and $\ell = p+2-j \bmod q$. Because of the periodicity, $\mathcal{A}$ must be either j-left or j-right. Moreover, it cannot be both since otherwise a multiplicity would occur. Clearly, $\mathcal{A}$ is 0-left.

Let us assume the starting configuration is $\mathcal{Q}_j = (1, 0^{p-j}, 1, 0^j)^q$ for some $0 \leq j \leq \lfloor p/2 \rfloor$. Assume first that $\mathcal{A}$ is j-right. In configuration $\mathcal{Q}_j = (1, 0^{p-j}, 1, 0^j)^q$, the adversary wakes up and makes the $p-j+1^{th}$ robot of each period $(1, 0^{p-j}, 1, 0^j)$ move. Therefore, following $\mathcal{A}$, the configuration $\mathcal{Q}_{j-1} = (1, 0^{p-j+1}, 1, 0^{j-1})^q$ is eventually reached. Since, $\mathcal{A}$ is 0-left, the algorithm eventually reaches a configuration $\mathcal{Q}_{j^*}$ such that $\mathcal{A}$ is j^*-left. Hence, let us now assume that $\mathcal{A}$ is j-left.

Let h be the smallest integer such that $j < h \leq \lfloor p/2 \rfloor$ and $\mathcal{A}$ is h-right. If no such h exists, we set $h = \lfloor p/2 \rfloor + 1$. For any $j \leq s < h$, $\mathcal{A}$ is s-left and, in configuration $\mathcal{Q}_s = (1, 0^{p-s}, 1, 0^s)^q$, the adversary wakes up and makes the first robot of each period $(1, 0^{p-s}, 1, 0^s)$ move. Therefore, following $\mathcal{A}$, the configuration $\mathcal{Q}_{h-1} = (1, 0^{p-h+1}, 1, 0^{h-1})^q$ is eventually reached. If p is even and $h = \lfloor p/2 \rfloor$, the adversary proceeds similarly to reach $\mathcal{Q}_h \in \mathcal{P}_1$ and the algorithm fails by previous result. Otherwise, the adversary wakes up and makes all robots to compute. However, only the first robot of each period moves. Therefore, the configuration $\mathcal{Q}_h = (1, 0^{p-h}, 1, 0^h)^q$ is reached where the $p-h^{th}$ robot of each period is going to move (there are q pending moves). Since $\mathcal{A}$ is h-right (the case $h = \lfloor p/2 \rfloor + 1$ works similarly by symmetry), the adversary wakes up and makes the $p-h+1^{th}$ robot of each period move. Finally, all pending moves are done, resulting in multiplicities.

Note that the impossibility does not rely on the task to be executed but on the exclusivity property that must be satisfied. □

6 Details on Algorithm Align

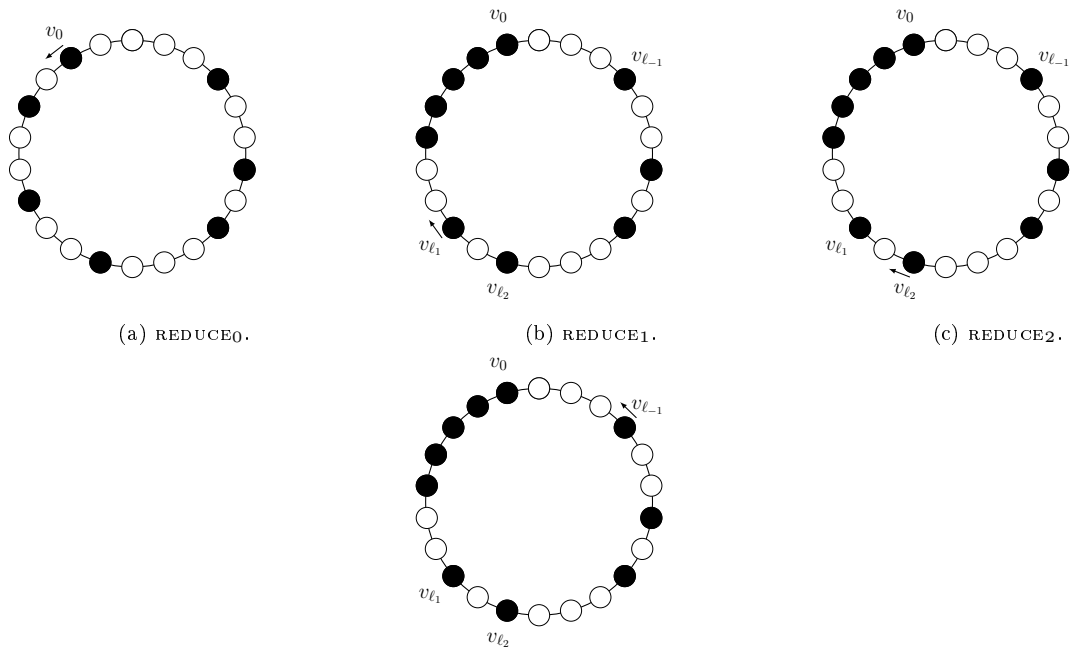
In this section, we provide the details to formally describe algorithm Align that, starting from any allowed configuration, reaches one of the exclusive configurations $\mathcal{C}^a$, $\mathcal{C}^b$, and $\mathcal{C}^c$ previously defined.

In Section 6.1 we present the algorithm and describe its general behavior, and in Section 6.3 we analyze some particular special cases which are omitted in the general discussion for the sake of simplicity. In Section 6.2, we give two examples of the execution. In Section 6.4, we provide the pseudo-code of the algorithm. For the ease of reading, Section 6.5 is devoted to the proofs of some lemmata stated in Section 6.1 along with the correctness proof of the algorithm.

6.1 Algorithm Align

Algorithm Align is based on four procedures described below. Let $\mathcal{C}$ be any allowed configuration and let $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$ be its supermin. Abusing notation, we denote by v_i both the $(i+1)$-th node and the $(i+1)$-th value of sequence $\mathcal{C}^{\min}$. Let ℓ_1 be the smallest integer such that $\ell_1 > 0$, $v_{\ell_1} = 0$ and $v_{\ell_1 - 1} = 1$ (i.e. v_{ℓ_1} is the first node of the second sequence of consecutive occupied nodes); let ℓ_2 be the smallest integer such that $\ell_2 > \ell_1$, $v_{\ell_2} = 0$ and $v_{\ell_2 - 1} = 1$ (i.e. v_{ℓ_2} is the first node of the third sequence of consecutive occupied nodes); let ℓ_{-1} be the largest integer such that $\ell_{-1} < n$ and $v_{\ell_{-1}} = 0$ (i.e. $v_{\ell_{-1}}$ is the last occupied node). The four procedures permitted by Align are the following (see Figure 7):

- **reduce$_0(\mathcal{C})$:** The robot at node v_0 moves to node v_1;



(a) REDUCE$_0$. (b) REDUCE$_1$. (c) REDUCE$_2$.

(d) REDUCE$_{-1}$.

Fig. 7: Procedures permitted by ALIGN.

- **REDUCE$_1(\mathcal{C})$:** The robot at node v_{ℓ_1} moves to node v_{ℓ_1-1};
- **REDUCE$_2(\mathcal{C})$:** The robot at node v_{ℓ_2} moves to node v_{ℓ_2-1};
- **REDUCE$_{-1}(\mathcal{C})$:** The robot at node $v_{\ell_{-1}}$ moves to node $v_{\ell_{-1}+1}$.

Note that in some configurations ℓ_1 and ℓ_2 might be not defined. However, we will show that in these cases our algorithm does not perform procedures REDUCE$_1$ and REDUCE$_2$, respectively.

The pseudo-code of Algorithm ALIGN is given in Figure 10 and works in two phases. The first phase (Algorithm ALIGN-ONE in Figure 11) copes with configurations without any consecutive occupied nodes (i.e. $v_1 = 1$) and aims at reaching configurations with at least two consecutive occupied nodes (i.e. $v_1 = 0$), once one such configuration is reached, the second phase starts and its general aim is to increase the number of consecutive occupied nodes until $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$ are reached. The second phase is given in Algorithm ALIGN-TWO-SYM in Figure 12, if the configuration is symmetric, and ALIGN-TWO-ASYM in Figure 13, otherwise.

Algorithm ALIGN-ONE. If $v_1 = 1$ (i.e. there are no two adjacent robots) and the configuration $\mathcal{C}$ is symmetric, the general strategy is to reduce the supermin by performing REDUCE$_0$ (see line 5 of Procedure ALIGN-ONE). If the two symmetric robots that should move perform their Look-Compute-Move cycles synchronously, then the obtained configuration $\mathcal{C}'$ is symmetric, the axis of symmetry of $\mathcal{C}$ is preserved, and the supermin is reduced. Hence, $\mathcal{C}'$ is allowed.

If only one of the two symmetric robots that should move actually performs the move (due to the asynchronous execution of their respective Look-Compute-Move cycles), then the following lemma ensures that the configuration $\mathcal{C}'$ obtained is asymmetric and not adjacent to any symmetric configuration other than $\mathcal{C}$ with respect to any possible procedure that allows at most two robots to move.

Lemma 2 ([12]) *Let $\mathcal{C}$ be an allowed configuration and let $\mathcal{C}'$ be the one obtained from $\mathcal{C}$ after a* REDUCE$_0$ *performed by a single robot. Then, $\mathcal{C}'$ is asymmetric and at least two robots have to move to obtain $\mathcal{C}'$ from an aperiodic symmetric configuration different from $\mathcal{C}$.*

It follows that robots can recognize whether $\mathcal{C}'$ has been obtained by performing REDUCE$_0$ from $\mathcal{C}$ by performing such a procedure on $\mathcal{C}'$ backwards. In fact, if the configuration is asymmetric, then ALIGN-ONE first checks whether it has been obtained from a symmetric configuration (By Lemma 2, such a configuration is unique), and in the affirmative case, it performs the possible pending move. In detail, let $\mathcal{C} = (0, 1, X, 1)$

be the supermin view of an *asymmetric* configuration (line 7). Performing REDUCE$_0$ backwards from $\mathcal{C}$ means computing the configuration $\mathcal{C}' = (1, 1, X, 0)$ (line 8) since the move from $\mathcal{C}'$ to $\mathcal{C}$ corresponds to REDUCE$_0$. If $\mathcal{C}'$ is symmetric and allowed (line 9), then a REDUCE$_0$ move is possibly pending and hence it is forced to be performed (line 11).

However, it is not always possible to perform REDUCE$_0$ on a symmetric configuration $\mathcal{C}$. Indeed, in case that $\mathcal{C}^{\min} = (0, 1, 0, R)$, for some $R = \overline{R}$, then performing REDUCE$_0$ would imply that two robots occupy the same node (a multiplicity occurs but we want to avoid it in this phase). In fact, note that in this case the node symmetric to v_0 is v_2 and performing REDUCE$_0$ consists in moving both robots from v_0 and v_2 to v_1. In this case, we perform REDUCE$_{-1}$ (line 3). In the next lemma (for $j = 1$) we show that such a procedure performed by only one robot from a configuration $\mathcal{C}$ such that $\mathcal{C}^{\min} = (0, 1, 0, R)$, with $R = \overline{R}$, does not create a configuration with two consecutive occupied nodes, does not create a symmetric configuration and the configuration obtained is not adjacent to a configuration different from $\mathcal{C}$ with respect to any possible procedures performed by ALIGN.[1]

Lemma 3 *Let $\mathcal{C}$ be a symmetric and allowed configuration with supermin $\mathcal{C}^{\min} = (0, 1^j, 0, R)$, for $R = \overline{R}$ and $j \geq 1$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_{-1}$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ has no consecutive occupied nodes and either $\mathcal{C}^{\min} = (0, 1^j, 0, 1^{j+1}, 0, 1^{j+1})$ or $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_0$ *and* REDUCE$_{-1}$ *to any symmetric configuration different from $\mathcal{C}$.*

It follows that we can again preserve the symmetry by forcing to perform the symmetric move. Note that also in this case, performing REDUCE$_{-1}$ results in reducing the supermin.

In order to recognize whether a REDUCE$_{-1}$ move is possibly pending, we use a technique similar to that used to recognize a possible pending REDUCE$_0$ move. In detail, Let $\mathcal{C} = (0, 1, 0, R)$ be the supermin view of a symmetric configuration with $R = \overline{R}$. We can assume that $R = (1^j, 0, R', 0, 1^j)$ with $R' = \overline{R'}$ and $j > 1$. After performing a REDUCE$_{-1}$ on only one robot, the obtained configuration is $\mathcal{C}' = (0, 1, 0, 1^j, 0, R', 1, 0, 1^{j-1})$. Two cases may occur: ${\mathcal{C}'}^{\min} = (0, 1, 0, 1^{j-1}, 0, 1, R', 0, 1^j)$; or, if $j = 2$, ${\mathcal{C}'}^{\min} = (0, 1, 0, 1, 0, 1, 1, 0, R', 1)$. In the first case, we can compute $\mathcal{C}$ from $\mathcal{C}'$ by moving the robot in position $j+2$ to position $j+3$ (lines 17–21), in the second case by moving the robot in position 0 to position $n - 1$ (lines 13–15).

If the configuration is asymmetric and it cannot be obtained by performing REDUCE$_0$ or REDUCE$_{-1}$ from any possible allowed symmetric configuration, then we execute Algorithm ASYM in [14] (line 22). Lemma 1 ensures that such algorithm always leads to rigid configurations.

Algorithm ASYM ensures that each procedure permits only one robot to change its position, and then no pending moves are possible. If by applying ASYM, we produce an asymmetric configuration which is adjacent to a symmetric configuration with respect to some of the procedures permitted by ALIGN, then we force to perform the possible pending move. Moreover, it has been shown that algorithm ASYM reduces the supermin after each move [14].

Note that, in some symmetric configurations there exists a robot r that occupies a node lying on the axis of symmetry. In these cases, REDUCE$_0$ or REDUCE$_{-1}$ may consist in moving r (in any arbitrary direction). We cannot move the robot symmetric to r as it does not exist, but we can safely perform ASYM as there are no pending moves. To avoid to force the pending move in this case, we test whether the robot that moved from $\mathcal{C}$ to $\mathcal{C}'$ is not the one on the axis of symmetry of $\mathcal{C}$ (see test at lines 10, 14, and 20).

Eventually, ALIGN-ONE leads to a configuration with two consecutive occupied nodes. In detail, we can obtain one of the following four configurations: (i) an asymmetric configuration with two consecutive occupied nodes which is not adjacent to any symmetric configuration with respect to a procedure permitted by ALIGN-ONE; (ii) an asymmetric configuration with two consecutive occupied nodes which is adjacent to a symmetric configuration with respect to some procedure permitted by ALIGN-ONE; (iii) a symmetric configuration with two or three consecutive occupied nodes with the axis of symmetry passing in their middle; (iv) a symmetric configuration with two symmetric pairs of consecutive occupied nodes.

Algorithm ALIGN-TWO-SYM. Once a configuration with two consecutive occupied nodes is achieved, the second phase of Algorithm ALIGN starts. Now it is not possible to perform REDUCE$_0$ as it would cause a multiplicity. Hence, one proce-

[1] Configuration $\mathcal{C} = (0, 1, 0, 1, 1, 0, 1, 1)$ is the only exception, see Section 6.3.

dure among REDUCE$_1$, REDUCE$_2$ or REDUCE$_{-1}$ is performed.

In symmetric configurations there are cases when we cannot perform REDUCE$_1$ or REDUCE$_2$. For instance, REDUCE$_1$ cannot be applied if $\mathcal{C}^{\min} = (0^i, 1^j, 0^i, R)$, with $R = \overline{R}$. In fact, in this case, $\mathcal{C}^{\min} = \overline{(\mathcal{C}^{\min}_{2i+j})}$ and performing REDUCE$_1$ corresponds to moving the robot at v_{i+j} which is symmetric to that at v_{i-1}. Therefore, such a move would increase the supermin. Similar instances where it is not possible to perform REDUCE$_2$ can occur. For example if $\mathcal{C}^{\min} = (0^i, 1^j, 0^x, 1^j, 0^i, R)$, then performing REDUCE$_2$ corresponds to moving the robot at v_{i+2j+x} which is symmetric to that at v_{i-1}. Moreover, since we are coping with symmetric configurations, it can happen that the asynchronous execution of the two symmetric robots that should perform one of the three procedures generates a symmetric configuration with a different axis of symmetry or a configuration that is adjacent to a different symmetric configuration with respect to some other procedure permitted by ALIGN. Algorithm ALIGN-TWO-SYM appropriately performs REDUCE$_1$, REDUCE$_2$ or REDUCE$_{-1}$ in a way that the conditions described above cannot occur.

To give more detail on the behavior of the algorithm in the case of symmetric configurations, we define the following three sets. Let S_1 be the set of symmetric configurations with supermin $(0^i, 1, R)$, where $i \geq 2$ and R contains a sequence 0^i. Let S_2 be the set of configurations $\mathcal{C} \in S_1$ such that $\mathcal{C}^{\min} = (0^i, 1^j, 0^i, Z)$ for some $Z = \overline{Z}$ and $j > 0$. Let S_3 be the set of configurations $\mathcal{C} \in S_1$ such that $\mathcal{C}^{\min} = (0^i, 1^{j'}, 0^x, 1^j, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j, j' > 0$ and $1 \leq x \leq i$, or configurations $\mathcal{C} \in S_1$ such that $\mathcal{C}^{\min} = (0^i, 1^{j'}, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j' > 0$ and $1 \leq x \leq i$, or configurations $\mathcal{C} \in S_1$ such that $\mathcal{C}^{\min} = (0^i, 1^j, 0^{i-1}, 1, 0, R, 1)$, $R = \overline{R}$, $j > 0$. Finally, set S_4 is a set of symmetric configurations such that $S_4 \not\subseteq S_1 \setminus S_3$ that is used to handle some special cases and will be defined in Section 6.3.

The sets S_2 and S_3 contain the configurations where it is not possible to perform REDUCE$_1$ or REDUCE$_2$, respectively, as it will increase the supermin.

In Lemmata 4–7, given in Section 6.5, we identify the procedures that can be safely performed on the configurations in such sets. We report the statements of such lemmata in what follows.

Lemma 4 *Let $\mathcal{C}$ be a symmetric and allowed configuration with $i > 1$ consecutive occupied nodes and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_1$ *on only one robot on $\mathcal{C}$. If $\mathcal{C}'$ is symmetric, then $\mathcal{C} \in S_1 \setminus S_3$ or $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, for some $\ell \geq 1$, or $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $\ell \geq 1$ and $1 \leq x \leq i$.*

Lemma 5 *Let $\mathcal{C}$ be a symmetric and allowed configuration with $i > 1$ consecutive occupied nodes and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_1$ *on only one robot on $\mathcal{C}$. If $\mathcal{C}'$ is adjacent with respect to* REDUCE$_1$ *to a symmetric configuration $\mathcal{C}''$ different from $\mathcal{C}$, then $\mathcal{C} \in S_1 \setminus S_3$, or $\mathcal{C}'' \in S_1 \setminus S_3$ or $\mathcal{C}, \mathcal{C}'' \in S_4$.*

Lemma 6 *Let $\mathcal{C}$ be a configuration in $S_1 \setminus S_3$ and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_2$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_1$ *or* REDUCE$_2$ *to any symmetric configuration different from $\mathcal{C}$.*

Lemma 7 *Let $\mathcal{C}$ be a configuration in S_2, or such that $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, for some $\ell \geq 1$, or such that $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $\ell \geq 2$ and $1 \leq x \leq i$, or such that $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, 0^i, 1, 0^y, 1)$, for some $1 < y < x \leq i$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_{-1}$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_1$, REDUCE$_2$, *or* REDUCE$_{-1}$ *to any symmetric configuration different from $\mathcal{C}$.*

Based on these results, Algorithm ALIGN-TWO-SYM works as follows. If $\mathcal{C}$ is in S_2, then REDUCE$_1$ cannot be performed. However, by Lemma 7, we can safely perform REDUCE$_{-1}$ (lines 1 and 6 of Procedure ALIGN-TWO-SYM). If $\mathcal{C} \notin S_2$, then ALIGN-TWO-SYM first computes the configuration $\mathcal{C}'$ that would be obtained from $\mathcal{C}$ by applying REDUCE$_1$ on only one robot (line 8). If $\mathcal{C}'$ is symmetric, then we know by Lemma 4 that $\mathcal{C} \in S_1 \setminus S_3$ or $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, for some $\ell \geq 1$, or $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $\ell \geq 1$ and $1 \leq x \leq i$. In the first case, we can safely perform REDUCE$_2$ (line 10) as the obtained configuration is neither symmetric nor adjacent to any other symmetric configuration (see Lemma 6, and the first example in Section 6.2). In the last two cases, we cannot perform REDUCE$_2$ but, by Lemma 7, we can safely perform REDUCE$_{-1}$ (lines 2, 3, and 6).[2]

[2] Except for the case of $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$ for $\ell = 1$ and $x = 1$

If $\mathcal{C}'$ is asymmetric, then ALIGN-TWO-SYM checks whether it can be obtained by applying REDUCE$_1$ from a symmetric configuration $\mathcal{C}''$ different from $\mathcal{C}$. To this aim, it computes all the configurations that can possibly generate $\mathcal{C}'$. As REDUCE$_1$ reduces the supermin, then by performing it, the starting node of the supermin in the obtained configuration is either the same of the previous one or it is one of the endpoints of a sequence of consecutive occupied nodes which is generated by the procedure itself. It follows that $\mathcal{C}''$ can be computed by increasing the supermin of $\mathcal{C}'$ by moving one of the robots in the endpoints of the sequence of consecutive occupied nodes at the beginning of the supermin sequence or the possible robot in position v_{ℓ_1}. In other words, if $\mathcal{C}' = (0^i, 1^j, 0, R, 1)$ for $i \geq 2$ and $j \geq 1$ (line 12), then $\mathcal{C}''$ can be only one of the following configurations: $\mathcal{C}^\alpha := (0^{i-1}, 1, 0, 1^{j-1}, R, 1)$, $\mathcal{C}^\beta := (0^{i-1}, 1^j, R, 0, 1)$, and, if $R = (1, R')$, $\mathcal{C}^\gamma := (0^i, 1^{j+1}, 0, R', 1)$ (lines 13–15). If at least two among $\mathcal{C}^\alpha$, $\mathcal{C}^\beta$, and $\mathcal{C}^\gamma$ are symmetric and the procedure from both of them to $\mathcal{C}'$ corresponds to REDUCE$_1$ (i.e. two symmetric configurations are adjacent to $\mathcal{C}'$ with respect to REDUCE$_1$), then by Lemma 5 follows that at least one of them belongs to $S_1 \setminus S_3$ or both belong to S_4. In the former case we can safely perform REDUCE$_2$ on the configuration belonging to $S_1 \setminus S_3$ (line 17) and REDUCE$_1$ (line 22) on the other one (see Lemma 6 and second example in Section 6.2). The latter case will be explained in detail in Section 6.3. In any other symmetric configuration, ALIGN-TWO-SYM applies REDUCE$_1$ (line 22).

Algorithm ALIGN-TWO-ASYM. This algorithm works similarly to ALIGN-ONE when the configuration is asymmetric. First, it checks whether the given configuration $\mathcal{C}$ has been obtained from a symmetric and allowed configuration $\mathcal{C}'$ by performing only one of the two symmetric moves. In the affirmative case, it performs the possible pending move, otherwise it performs Algorithm ASYM. Given the procedures performed by ALIGN-ONE and ALIGN-TWO-SYM, a configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (0^i, 1^j, 0^x, 1^{j'}, R, 1)$, $j \geq 1$, $x \geq 1$, and $j' \geq 0$ can be adjacent to a symmetric configuration $\mathcal{C}'$ with respect to one such procedure only if $\mathcal{C}'$ is one of the following configurations: $\mathcal{C}^\alpha := (0^{i-1}, 1, 0, 1^{j-1}, 0^x, 1^{j'}, 0, R, 1)$, $\mathcal{C}^\beta := (0^{i-1}, 1^j, 0^x, 1^{j'}, 0, R, 0, 1)$, if $j' > 0$, $\mathcal{C}^\gamma := (0^i, 1^j, 0^{x-1}, 1, 0, 1^{j'-1}, R, 1)$, or, if $R = (0, 1, R')$, $\mathcal{C}^\delta := (0^i, 1^j, 0^x, 1^{j'+1}, 0, R', 1)$ (see lines 2–5 of ALIGN-TWO-ASYM). Note that, at most one of the above configurations can be symmetric. Let $\mathcal{C}^y$, $y \in \{\alpha, \beta, \gamma, \delta\}$, be such a configuration, if by applying ALIGN-TWO-SYM (or ALIGN-ONE if $\mathcal{C}^y$ has no consecutive occupied nodes) on a single robot of $\mathcal{C}^y$ we obtain $\mathcal{C}$, then $\mathcal{C}$ has been possibly obtained from $\mathcal{C}^y$ and then ALIGN-TWO-ASYM performs the possible pending move (see lines 7–14 and the first example in Section 6.2). If none of $\mathcal{C}^y$, $y \in \{\alpha, \beta, \gamma, \delta\}$, is symmetric, then $\mathcal{C}$ has not been obtained from any symmetric configurations and then ALIGN-TWO-ASYM performs ASYM (line 15). As in the case of ALIGN-ONE, if the robot leading from $\mathcal{C}^y$ to $\mathcal{C}$ is that on the axis of symmetry of $\mathcal{C}^y$, then Algorithm ASYM is performed.

We are now ready to provide the proof of Theorem 1 which represents the correctness proof for algorithm ALIGN. Such a proof relies on Lemmata 3–7 and Lemmata 15–18, proven in Section 6.5. For the sake of readability, we also recall the statement of the theorem.

Theorem 5 (Theorem 1) *Let $3 \leq k < n - 2$, $k \neq 4$, robots standing in an n-node ring forming an exclusive allowed configuration, Algorithm* ALIGN *eventually terminates achieving one exclusive allowed configuration among $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$.*

Proof We model all the possible executions of ALIGN as a directed graph where each configuration is represented as a node and there exists an arc (u, v) if there exist a procedure and a time schedule of the algorithm that starting from the configuration represented by u lead to that represented by v, even with possible pending moves. An execution of ALIGN is represented by a path in this graph. In what follows, we show that such paths are acyclic, are made of nodes representing allowed configurations, and they always lead to a node representing one of the configurations $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$.

We can partition the nodes into three sets representing: the symmetric configurations; the asymmetric configurations which are adjacent to some symmetric configurations with respect to one of the procedures permitted by ALIGN; and the remaining asymmetric configurations. We denote such sets as S, $AS1$ and $AS2$, respectively. Lemmata 1, 2, 3–7, and 15–17 imply the following properties.

- A node in S representing a configuration $\mathcal{C}$ has either one or two outgoing arcs (but for nodes

(i.e. $\mathcal{C}^{\min} = (0^i, 1, 0, 1, 0^i, 1, 1)$) where REDUCE$_1$ is performed. This case will be discussed in Section 6.3, along with the case $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, 0^i, 1, 0^y, 1)$, $y < x$.

corresponding to $\mathcal{C}^b$ and $\mathcal{C}^c$ that have no outgoing arcs, see last item). If it has exactly two outgoing arcs, then one of them is directed to the node v' representing the configuration $\mathcal{C}'$ obtained if both the symmetric robots permitted to move by ALIGN perform their moves synchronously. The other arc is directed to the node v'' representing the configuration $\mathcal{C}''$ obtained if only one of the two symmetric robots permitted to move by ALIGN actually moves. In other words, the former arc models the case where both the two symmetric robots permitted to move perform the entire cycle Look-Compute-Move, while the latter arc models the case where only one of them performs entirely such cycle. Note that, v' belongs to S, while v'' belongs to $AS1$. Moreover, if $\mathcal{C}$ is allowed, then also $\mathcal{C}'$ is such. If the node has exactly one outgoing arc then the robot r moved by ALIGN lies on the axis of symmetry. In this case, any procedure performed by ALIGN moves r in an arbitrary direction. Then, the arc is directed to a node in $AS1$.

- A node in $AS1$ representing a configuration $\mathcal{C}''$ has exactly one incoming arc from a node in S, it can have some incoming arcs from nodes in $AS2$, and it has exactly one outgoing arc, directed to a node in S or in $AS2$. If the outgoing arc is directed to a node in S, then one of the incoming arcs comes from a node u in S and models the case when only one of the two symmetric robots permitted to move by ALIGN from the configuration $\mathcal{C}$ represented by u actually moves. From Lemmata 3–15, there exists only one such node. The outgoing arc leads to the node in S representing configuration $\mathcal{C}'$ which can be obtained by moving synchronously both the symmetric robots permitted to move by ALIGN from $\mathcal{C}$. Note that both $\mathcal{C}$ and $\mathcal{C}''$ are allowed configurations (see line 7 of Procedure ALIGN-TWO-ASYM). If the outgoing arc is directed to a node in $AS2$, then $\mathcal{C}''$ has been obtained from a configuration, corresponding to a node in S, such that the robot moved by ALIGN lies on the axis of symmetry. In this case, ALIGN performs ASYM from $\mathcal{C}''$ obtaining a configuration in $AS2$.
- A node in $AS2$ has exactly one outgoing arc, directed either to another node in $AS2$ or to a node in $AS1$ but it cannot be directed to a node in S (by Lemma 1). It can have some arcs coming from nodes in $AS1$ or $AS2$. If the node corresponding to $\mathcal{C}^a$ belongs to $AS2$ it has no outgoing arcs, see next item.
- Node corresponding to configurations $\mathcal{C}^b$ and $\mathcal{C}^c$ belong to S but they have no outgoing arcs as the algorithm stops when one such configuration is achieved. Configuration $\mathcal{C}^a$ can belong to $AS1$ or to $AS2$, depending on whether $\mathcal{C}^b$ is allowed or not. In the former case, there is an arc between the node corresponding to $\mathcal{C}^a$ to that corresponding to $\mathcal{C}^b$, in the latter case the node corresponding to $\mathcal{C}^a$ has no outgoing edge and the algorithm stops such configuration is achieved.

It follows that any execution path performed by the algorithm is made of nodes representing allowed configurations. In fact, the arcs outgoing nodes in S represent moves that preserve the axis of symmetry and the number of robots. Therefore, if an allowed symmetric configuration $\mathcal{C}$ corresponds to a node v and there is an arc (v, v'), then the symmetric configuration $\mathcal{C}'$ corresponding to node v' is allowed. Furthermore, configurations in $AS1$ and $AS2$ are always allowed and there is an arch from a node in $AS1$ to a node in S only if this latter corresponds to an allowed configuration (see line 7 of Procedure ALIGN-TWO-ASYM).

Moreover each allowed configuration (but $\mathcal{C}^b$, $\mathcal{C}^c$, and, in some cases, $\mathcal{C}^a$) has an outgoing arc that is traversed by the execution path of the algorithm.

Since nodes in $AS1$ have only one outgoing arc, without loss of generality we can consider a condensed graph build in the following way: replace each arc (u, v), such that $v \in AS1$ and the unique out-neighbor z of v belongs to S with arc (u, z). Any execution path in the original graph has a unique correspondent in the condensed graph, while an execution path in the condensed graph only omits the arcs corresponding to forced pending moves. By Lemma 18 and since ASYM always reduces the supermin, it follows that each arc in the condensed graph corresponds to a reduction of the supermin. This implies that the condensed graph is acyclic, as we can define a topological ordering of the nodes as a linear extension of the partial ordering given by the supermin of the corresponding configurations. The statement is then proven by observing that configurations in $\mathcal{C}^a$, $\mathcal{C}^b$, or $\mathcal{C}^c$ are those with the minimum possible supermin and hence are the only possible sinks of the graph. □

6.2 Examples of execution

First example. The example in Figure 8 shows a case where applying REDUCE$_1$ on a symmetric configuration results in a symmetric configuration with a different axis of symmetry. Let us consider the configuration $\mathcal{C}$ in Figure 8a such that $\mathcal{C}^{\min} = (0,0,1,0,0,1,0,1,0,0,1,0,0,1,0,1,0,1)$. As $v_1 = 0$ and $\mathcal{C}$ is symmetric, then Algorithm ALIGN-TWO-SYM is performed. It first computes configuration $\mathcal{C}'$ in Figure 8b which is the one that would be obtained from $\mathcal{C}$ by applying REDUCE$_1$ on only one robot. As such a configuration is symmetric, REDUCE$_2$ is applied. If only one robot moves, then the configuration $\mathcal{C}''$ in Figure 8c is obtained. Such configuration is asymmetric and it could have a possible pending move.

From configuration $\mathcal{C}''$, Algorithm ALIGN-TWO-ASYM is applied. Such a procedure computes the unique symmetric configuration which $\mathcal{C}''$ is adjacent to. To this aim, it computes the four possible configurations that can generate $\mathcal{C}''$ by applying ALIGN. Such configurations are:

- $\mathcal{C}^\alpha$ given in Figure 8d;
- $\mathcal{C}^\beta$ which is equivalent to $\mathcal{C}$;
- $\mathcal{C}^\gamma$ given in Figure 8e;
- $\mathcal{C}^\delta$ given in Figure 8f.

Among such configurations, only one is symmetric which is $\mathcal{C}^\beta = \mathcal{C}$. Therefore, ALIGN-TWO-ASYM is able to identify the robot that has to move in order to perform the possible pending move. In this specific case, the robot that moved from $\mathcal{C}$ to $\mathcal{C}''$, is the one on the axis of symmetry. It follows that there are no pending moves and ALIGN-TWO-ASYM proceeds by applying ASYM.

Second example. The example in Figure 9 shows a case where applying REDUCE$_1$ on a symmetric configuration results in an asymmetric configuration which is adjacent to another symmetric configuration, different from the original one, with respect to REDUCE$_1$. Let us consider the configuration of Figure 9a. As $v_1 = 0$ and $\mathcal{C}$ is symmetric, then Algorithm ALIGN-TWO-SYM is performed. It first computes configuration $\mathcal{C}'$ in Figure 9b which is the one that would be obtained from $\mathcal{C}$ by applying REDUCE$_1$ on only one robot. As such configuration is asymmetric, the procedure checks whether it can be obtained by applying REDUCE$_1$ from a symmetric configuration different from $\mathcal{C}$. To this aim it computes:

- $\mathcal{C}^\alpha$ which is equivalent to $\mathcal{C}$;
- $\mathcal{C}^\beta$ given in Figure 9d;
- $\mathcal{C}^\gamma$ given in Figure 9e.

Both configurations $\mathcal{C}^\alpha$ and $\mathcal{C}^\gamma$ are symmetric, and configuration $\mathcal{C}'$ can be obtained from both of them by applying REDUCE$_1$. By Lemma 5, it follows that one of them belongs to $S_1 \setminus S_3$ or both of them belong to S_4. In fact, $\mathcal{C}^\alpha \in S_1 \setminus S_3$. Therefore, Algorithm ALIGN-TWO-SYM exploits Lemma 6 and applies REDUCE$_2$ on $\mathcal{C}$. The obtained configuration $\mathcal{C}''$ is given in Figure 9c. It can be checked that this configuration is asymmetric and it is not adjacent to any symmetric configuration different from $\mathcal{C}$ with respect to any procedure permitted by ALIGN (as proved in Lemma 6 for the general case).

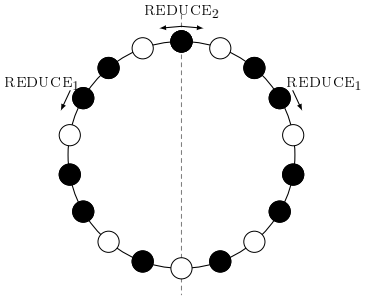
6.3 Further details on particular cases

In this section we give more details on particular configurations handled by ALIGN. First, we focus on the case of ALIGN-ONE, and then on ALIGN-TWO-SYM.

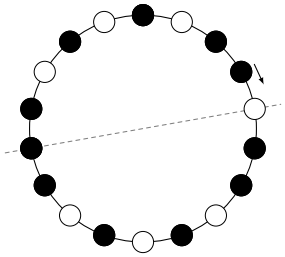
Algorithm ALIGN-ONE. Let us consider configuration $\mathcal{C} = (0,1,0,1,1,0,1,1)$. In this case, ALIGN performs procedure REDUCE$_{-1}$ which moves the robot in v_5 (lying on the axis of symmetry) toward an arbitrary directions. Such a procedure leads to the symmetric configuration $\mathcal{C}' = (0,1,0,1,0,1,1,1)$. From $\mathcal{C}'$, procedure REDUCE$_0$ is performed which leads to configuration $\mathcal{C}^b$, eventually. In fact, if both the two symmetric robots that should perform REDUCE$_0$ move synchronously, the configuration obtained is $(0,0,0,1,1,1,1,1)$. Otherwise, if only one of them actually moves, the configuration obtained is $\mathcal{C}'' = (0,0,1,0,1,1,1,1)$. As $\mathcal{C}''$ is asymmetric, Algorithm ALIGN-TWO-ASYM is performed. Such algorithm computes $\mathcal{C}^\alpha = (0,1,0,0,1,1,1,1)$, $\mathcal{C}^\beta = (0,1,0,1,1,1,0,1)$, and $\mathcal{C}^\gamma = (0,0,1,1,0,1,1,1)$ and identifies $\mathcal{C}^\beta$ as the only symmetric configuration among them. Indeed, $\mathcal{C}^\beta$ corresponds to $\mathcal{C}'$. Therefore, the algorithm performs the pending move and obtains $(0,0,0,1,1,1,1,1)$.

Algorithm ALIGN-TWO-SYM. *Configurations that generate adjacent configurations with respect to* REDUCE$_1$. We first define the following two sets:

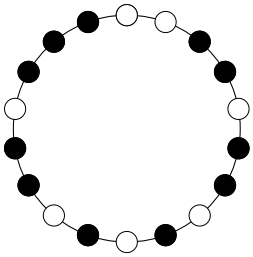
- S_{4a} are all the configurations $\mathcal{C}$ such that one of the following holds:
  1. $\mathcal{C}^{\min} = (0^i,1,1,0^x,1,1)$, for some $i \geq 3$ and $x < i$;
  2. $\mathcal{C}^{\min} = (0,0,0,0,1,1,0,0,0,1,0,0,0,1,1)$;
  3. $\mathcal{C}^{\min} = (0^i,1,1,1,0,1,0^{i-1}, (1,1,0,1,1,0^{i-1})^\ell,1,0,1,1,1)$, for some $i \geq 3$ and $\ell \geq 0$;



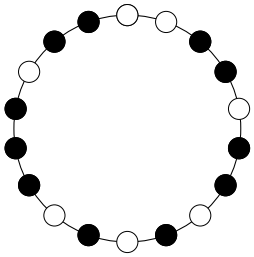
(a) Configuration $\mathcal{C}$.



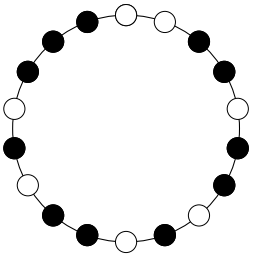
(b) Configuration $\mathcal{C}'$ computed from $\mathcal{C}$ by applying REDUCE$_1$.



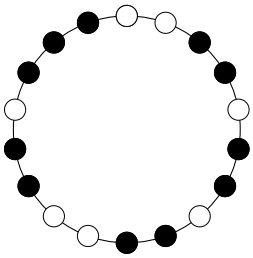
(c) Configuration $\mathcal{C}''$ computed from $\mathcal{C}$ by applying REDUCE$_2$.



(d) Configuration $\mathcal{C}^{\alpha}$ computed from $\mathcal{C}''$ in Algorithm Align-Two-Asym.

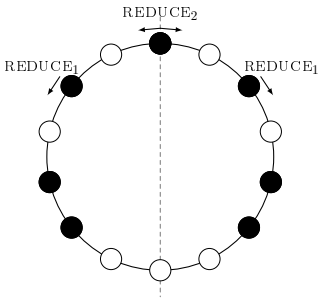


(e) Configuration $\mathcal{C}^{\gamma}$ computed from $\mathcal{C}''$ in Algorithm Align-Two-Asym.

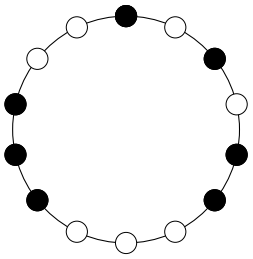


(f) Configuration $\mathcal{C}^{\delta}$ computed from $\mathcal{C}''$ in Algorithm Align-Two-Asym.

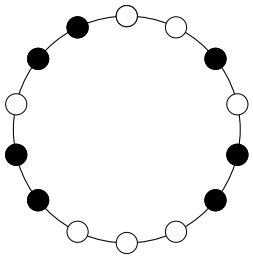
Fig. 8: First example



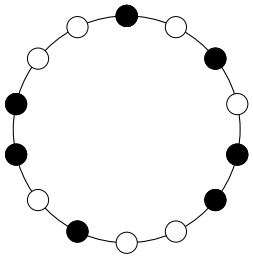
(a) Configuration $\mathcal{C}$.



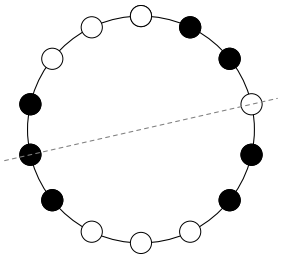
(b) Configuration $\mathcal{C}'$ computed from $\mathcal{C}$ by applying REDUCE$_1$.



(c) Configuration $\mathcal{C}''$ computed from $\mathcal{C}$ by applying REDUCE$_2$.



(d) Configuration $\mathcal{C}^{\beta}$ computed from $\mathcal{C}'$ in Algorithm Align-Two-Sym.



(e) Configuration $\mathcal{C}^{\gamma}$ computed from $\mathcal{C}'$ in Algorithm Align-Two-Sym.

Fig. 9: Second example

4. $\mathcal{C}^{\min} = (0,0,0,1,1,0,(0,1)^{\ell},0,0,1,1)$, for some $\ell \geq 1$;
5. $\mathcal{C}^{\min} = (0^{i},1,1,0^{i-1},(1,0,1,0^{i-2})^{\ell},1,0,1,0^{i-1},1,1)$, for some $i \geq 3$ and $\ell \geq 0$;
6. $\mathcal{C}^{\min} = (0^{i},1,0^{i},1,0^{i},1,0^{i},1,0^{i-1},1,0^{i},(1,0^{i-1},1,0^{i},1,0^{i},1,0^{i-1},1,0^{i})^{\ell},1,0^{i-1},1)$, for some $i \geq 2$ and $\ell \geq 1$.

– S_{4b} are all the configurations $\mathcal{C}$ such that one of the following holds:
1. $\mathcal{C}^{\min} = (0^{i-1},1,0,1,0^{x-1},1,0,1)$, for some $i \geq 3$ and $x < i$;
2. $\mathcal{C}^{\min} = (0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)$;
3. $\mathcal{C}^{\min} = (0^{i-1},1,0,1,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^{\ell},1,1,0,1,1)$, for some $i \geq 3$ and $\ell \geq 0$;
4. $\mathcal{C}^{\min} = (0,0,1,0,1,0,(0,1)^{\ell},0,1)$, for some $\ell \geq 2$;
5. $\mathcal{C}^{\min} = (0^{i-1},1,0,1,0^{i-1},(1,0,1,0^{i-2})^{\ell+1},1,0,1)$, for some $i \geq 3$ and $\ell \geq 0$;
6. $\mathcal{C}^{\min} = (0^{i},1,0^{i},1,0^{i},(1,0^{i-1},1,0^{i},1,0^{i},1,0^{i-1},1,0^{i})^{\ell},1,0^{i-1},1,0^{i},1,0^{i},1,0^{i-1},1)$, for some $i \geq 2$ and $\ell \geq 1$.

Set S_4 is given by the union of S_{4a} and S_{4b}. Observe that $S_4 \not\subseteq S_1 \setminus S_3$, that is, if $\mathcal{C} \in S_4$ either $\mathcal{C} \in S_3$ or $\mathcal{C} \notin S_1$. In particular, configurations 1–5 of S_{4a} and configuration 1 of S_{4b} do not belong to S_1, while configuration 6 of S_{4a} and configurations 2–6 of S_{4b} belong to S_3.

In Lemma 5 we show that the set S_4 contains the only configurations not in $S_1 \setminus S_3$ such that by applying REDUCE_1 on only one of the two symmetric robots that are allowed to move, we obtain a configuration that is adjacent with respect to REDUCE_1 to another symmetric configuration. The set of the configurations obtained by applying REDUCE_1 on S_4 is called S'_4. In particular, for any configuration $\mathcal{C}$ in S_{4a}, there exists a unique configuration $\mathcal{C}''$ in S_{4b} such that the configuration obtained by applying REDUCE_1 on $\mathcal{C}$ and that obtained by applying REDUCE_1 on $\mathcal{C}''$ are identical. For example, if we consider configurations $\mathcal{C}^{\min} = (0^{i},1,1,0^{x},1,1)$, and $\mathcal{C}''^{\min} = (0^{i-1},1,0,1,0^{x-1},1,0,1)$, for some $i \geq 3$ and $x < i$, then by applying REDUCE_1 we obtain $\mathcal{C}'^{\min} = (0^{i},1,0,1,0^{x-1},1,1)$ from both $\mathcal{C}$ and $\mathcal{C}''$.

We observe that in symmetric configurations in S_3 we cannot perform REDUCE_2, while in those not in S_1 procedures REDUCE_1 and REDUCE_{-1} coincide. Therefore the general strategy is to apply REDUCE_1 on configuration in S_{4a}, REDUCE_2 on configurations in $S_{4b} \setminus S_1$, and REDUCE_{-1} on configurations in $S_{4b} \cap S_3$. In detail, we distinguish the following three cases.

– Let us consider the symmetric configurations $\mathcal{C}^{s_1} = (0^{i},1,1,0^{x},1,1)$ and $\mathcal{C}^{s_2} = (0^{i-1},1,0,1,0^{x-1},1,0,1)$ with $i \geq 3$ and $x < i$. Note that in both these cases REDUCE_1 and REDUCE_{-1} coincide. Moreover, in $\mathcal{C}^{s_1}$ the only procedure that reduces the supermin and does not create a multiplicity is REDUCE_1. However, performing REDUCE_1 from $\mathcal{C}^{s_2}$ and $\mathcal{C}^{s_1}$ leads to the same configuration $\mathcal{C}'$. For these reasons Align-Two-Sym performs REDUCE_1 from $\mathcal{C}^{s_1}$ and, if $x > 1$, REDUCE_2 from $\mathcal{C}^{s_2}$ (lines 16–17 of procedure Align-Two-Sym). This latter procedure cannot create a symmetric configuration nor a configuration adjacent to any symmetric configuration different from $\mathcal{C}^{s_2}$ with respect to any procedure permitted by Align (see Lemma 15 in Section 6.5). If $x = 1$, then REDUCE_2 cannot be performed on $\mathcal{C}^{s_2}$. However, even if $\mathcal{C}'$ is adjacent to $\mathcal{C}^{s_1}$, it is equal to $\mathcal{C}^{a}$ and therefore in this case algorithm Align performs REDUCE_1 also from $\mathcal{C}^{s_2}$. Then, from $\mathcal{C}' = \mathcal{C}^{a}$ algorithm Align either stops or achieves $\mathcal{C}^{b}$, if it is allowed.
– Let us consider configurations $\mathcal{C}^{s_3} = (0,0,0,0,1,1,0,0,0,1,0,0,0,1,1)$ and $\mathcal{C}^{s_4} = (0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)$. By applying REDUCE_1 on them we obtain $\mathcal{C}' = (0,0,0,0,1,0,1,0,0,1,0,0,0,1,1)$. Note that in this case the only move that can be performed on $\mathcal{C}^{s_3}$ is REDUCE_1 as any other procedure might create a multiplicity. Similarly, performing REDUCE_2 on $\mathcal{C}^{s_4}$ might create a multiplicity, while performing REDUCE_{-1} might create a periodic configuration. Therefore in this case we perform REDUCE_1 on $\mathcal{C}^{s_3}$ and we move the robot in v_2 to v_3 in $\mathcal{C}^{s_4}$. In Lemma 16 given in Section 6.5, we show that this latter procedure creates a configuration $\mathcal{C}'''$ which is asymmetric and such that the only symmetric configuration which is adjacent to $\mathcal{C}'''$ with respect to any procedure permitted by Align is $\mathcal{C}'' = (0,0,0,1,0,0,0,1,0,0,1,1,0,0,1)$ which is the one that is obtained from $\mathcal{C}^{s_4}$ if both the symmetric robots (at nodes v_2 and v_{10}) move synchronously. Moreover, we observe that the obtained configuration $\mathcal{C}''' = (0,0,1,0,0,0,1,0,0,1,0,0,0,1,1)$ has a supermin that is smaller than that of $\mathcal{C}^{s_4}$ since $\mathcal{C}'''^{\min} = (0,0,0,1,0,0,1,0,0,0,1,0,0,1,1)$.
– In any other configuration in S_4 we perform REDUCE_1 on configurations in S_{4a} and REDUCE_{-1} on configuration in S_{4b}. Note that

all such configurations in S_{4b} are also contained in S_3. In Lemma 17, we show that applying REDUCE$_{-1}$ on such configurations cannot create a symmetric configuration nor a configuration adjacent to any symmetric configuration different from the original configuration with respect to any procedure permitted by ALIGN.

Algorithm ALIGN-TWO-SYM. *Configurations that generate symmetric configurations by performing* REDUCE$_1$. Finally, let us consider the case of $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $1 \leq x \leq i$ and $\ell = 1$, that is $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, 0^i, 1, 0^{x-1}, 1)$. We distinguish two cases: $x = 1$ and $x \geq 1$. In the first case, $\mathcal{C}^{\min} = (0^i, 1, 0, 1, 0^i, 1, 1)$ and the only possible procedure that reduces the supermin is REDUCE$_1$. However, as shown in Lemma 4, such a move produces a symmetric configuration, namely $\mathcal{C}'^{\min} = (0^{i+1}, 1, 1, 0^i, 1, 1)$. Note that, the robot moved from $\mathcal{C}$ to $\mathcal{C}'$ is the one on the axis of symmetry and hence there is no pending move. Moreover, $\mathcal{C}'$ is equivalent to configuration $\mathcal{C}^{s_1}$ therefore, from this point on, the ALIGN proceeds as in the previous case. If $x \geq 1$, then by performing REDUCE$_{-1}$, we obtain configuration $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0^x, 1, 0^i, 1, 0^{x-2}, 1)$ which is asymmetric. However, $\mathcal{C}'^{\min}$ can be obtained by performing REDUCE$_1$ from configuration $\mathcal{C}''^{\min} = (0^i, 1, 0^{x+1}, 1, 0^i, 1, 0^{x-2}, 1)$. Therefore, algorithm ALIGN performs move REDUCE$_{-1}$ to any configuration $\mathcal{C}'''$ such that $\mathcal{C}'''^{\min} = (0^i, 1, 0^x, 1, 0^i, 1, 0^y, 1)$, for some $1 < y < x \leq i$. This latter procedure cannot create a symmetric configuration nor a configuration adjacent to any symmetric configuration different from $\mathcal{C}'''$ with respect to any procedure permitted by ALIGN (see Lemma 7).

6.4 Pseudo-code

In Figures 10–13 we report the pseudo-code of algorithm ALIGN.

6.5 Correctness

In this section we prove the lemmata exploited in Theorem 1 .We first give two technical lemmas.

Lemma 8 *If* $(0, \overline{R}) = (R, 0)$ *and* $(0, R) = (\overline{R}, 0)$, *then* $R \in (0^k)$, $k \in \mathbb{N}$.

Proof R may be $\emptyset$. Clearly, $R = (0)$ if $|R| = 1$. Assume $|R| > 1$.

By induction on $0 \leq j \leq \lfloor \frac{n}{2} \rfloor$, we show that $R = (0^j, X, 0^j)$. Assume that $R = (0^j, a, X, b, 0^j)$, then, by symmetries, $(0, 0^j, a, X, b, 0^j) = (0^j, b, \overline{X}, a, 0^j, 0)$ and $(0, 0^j, b, \overline{X}, a, 0^j) = (0^j, a, X, b, 0^j, 0)$, hence $a = b = 0$ and thus, $R = (0^{j+1}, X, 0^{j+1})$. Then, $X = \emptyset$, or $|X| = 1$, in which case the only possibility is $X = (0)$, or $X = (0^{j+1}, a', X', b', 0^{j+1})$ and the result holds by induction. □

Lemma 9 *Let* X *and* Y *be two sequences such that* $(Y, X) = (X, \overline{Y})$ *and* $X = \overline{X}$. *Then,* $Y = (U, V)$ *and* $X = (Y^i, U)$, *for some* $U = \overline{U}$, $V = \overline{V}$, *and* $i \geq 0$.

Proof Clearly, any pair of sequences X, Y satisfying the properties of the lemma is a valid solution. We prove that any solution has the desired form by induction on the length of X.

Assume first that there is a solution to $(Y, X) = (X, \overline{Y})$ with $|X| \leq |Y|$. Then, X is a prefix of Y because $(Y, X) = (X, \overline{Y})$. Therefore, $Y = (X, V)$. Plugging it into the equation, we get $(X, V, X) = (X, \overline{V}, X)$. Therefore, $V = \overline{V}$. Hence, X and Y have the desired form.

Now, consider a solution such that $|X| > |Y|$. Then, Y is a prefix of X because $(Y, X) = (X, \overline{Y})$. Therefore, $X = (Y, X')$. Plugging it into the equation, we get $(Y, Y, X') = (Y, X', \overline{Y})$. Moreover, because $X = \overline{X}$, we get that $(Y, \overline{X'}, \overline{Y}) = (Y, X', \overline{Y})$. All together, we get that $X = (Y, X')$ with $(Y, X') = (X', \overline{Y})$ and $X' = \overline{X'}$.

Therefore, by induction, we get that, there is $i \geq 0$ such that $X' = (Y^i, U)$ with $Y = (U, V)$ and $U = \overline{U}$ and $V = \overline{V}$. Hence, $X = (Y^{i+1}, U)$ and the lemma holds. □

Lemma 10 (Lemma 3) *Let* $\mathcal{C}$ *be a symmetric and allowed configuration with supermin* $\mathcal{C}^{\min} = (0, 1^j, 0, R)$, *for* $R = \overline{R}$ *and* $j \geq 1$, *and let* $\mathcal{C}'$ *be the configuration obtained by applying* REDUCE$_{-1}$ *on only one robot on* $\mathcal{C}$. *Then* $\mathcal{C}'$ *has no consecutive occupied nodes and either* $\mathcal{C}^{\min} = (0, 1^j, 0, 1^{j+1}, 0, 1^{j+1})$ *or* $\mathcal{C}'$ *is asymmetric and it is not adjacent with respect to* REDUCE$_0$ *and* REDUCE$_{-1}$ *to any symmetric configuration different from* $\mathcal{C}$.

Proof We first prove that $\mathcal{C}'$ has no consecutive occupied nodes. By contradiction, let us assume that $\mathcal{C}'^{\min} = (0, 0, X)$ for some X. Since $\mathcal{C}$ has no consecutive occupied nodes, then such a sequence in $\mathcal{C}'$ has been created by the REDUCE$_{-1}$ move. This implies that $\mathcal{C}^{\min} = (0, 1^j, 0, 1, 0, R', 0, 1)$, i.e.

Algorithm: ALIGN
Input: Allowed configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$

```
if v_1 = 1 then
    ALIGN-ONE(C);
else
    if C is symmetric then
        ALIGN-TWO-SYM(C);
    else
        ALIGN-TWO-ASYM(C);
```

Fig. 10: Algorithm ALIGN.

Algorithm: ALIGN-ONE
Input: Allowed configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$

```
if C is symmetric then
    if C^min = (0,1,0,R), with R = R̄ then
        REDUCE_{-1}(C);
    else
        REDUCE_0(C);
else
    Let C^min = (0,1,X,1);
    Let C' = (1,1,X,0);
    if C' is symmetric and allowed and C'^min = (0,1,1,X) then
        if The robot that moved from C' to C is not the one on the axis of symmetry of C' then
            Perform REDUCE_0 on the robot symmetrical to the one that moved from C' to C and exit;
    else
        if C' is symmetric and allowed, C'^min = (0,1,0,1,1,0,R',0,1,1), and R' = R̄' then
            if The robot that moved from C' to C is not the one on the axis of symmetry of C' then
                Perform REDUCE_{-1} on the robot symmetrical to the one that moved from C' to C and exit;
        else
            if C^min = (0,1,0,1^{j-1},0,1,R',0,1^j) for some j > 1 then
                Let C' = (0,1,0,1^j,0,R',0,1^j);
                if C' is symmetric and allowed, C'^min = (0,1,0,1^j,0,R',0,1^j), and R' = R̄' then
                    if The robot that moved from C' to C is not the one on the axis of symmetry of C' then
                        Perform REDUCE_{-1} on the robot symmetrical to the one that moved from C' to C and exit;
    ASYM(C);
```

Fig. 11: First phase of Algorithm ALIGN.

$R = (1, 0, R', 0, 1)$, for some R'. This is a contradiction as $\mathcal{C}^{\min}_{n-2} < \mathcal{C}^{\min}$.

We now prove the second part of the statement. Since R is a palindrome, there cannot exist a supermin starting in R, as otherwise $\mathcal{C}$ is periodic. Therefore, $R = (1^{j'}, 0, 1^{j'})$ or $R = (1^{j'}, 0, R', 0, 1^{j'})$, for some $j' > j$ and $R' = \overline{R'}$. In the first case $\mathcal{C}' = (0, 1^j, 0, 1^{j'+1}, 0, 1^{j'-1})$ is symmetric or adjacent with respect to REDUCE_0 and REDUCE_{-1} to any symmetric configuration different from $\mathcal{C}$ only if $j' = j + 1$, that is $\mathcal{C}^{\min} = (0, 1^j, 0, 1^{j+1}, 0, 1^{j+1})$. In the latter case, the configuration obtained by applying REDUCE_{-1} on only one robot on $\mathcal{C}$ is $\mathcal{C}' = (0, 1^j, 0, 1^{j'}, 0, R', 1, 0, 1^{j'-1})$. We distinguish the following two cases.

- $j' - 1 > j$. In this case, $\mathcal{C}'^{\min} = (0, 1^j, 0, 1^{j'-1}, 0, 1, R', 0, 1^{j'})$. If $\mathcal{C}'$ is symmetric, then it contains a sequence $(0, 1^j, 0, 1^{j'-1}, 0)$ different from that at nodes v_0—$v_{j+j'+1}$ and not overlapping with it since $j' - 1 > j$. Such a sequence must exist also in $\mathcal{C}$ but this is a contradiction to the superminimality of $\mathcal{C}^{\min}$ as such a sequence is smaller than $(0, 1^j, 0, 1^{j'}, 0)$. It follows that $\mathcal{C}'$ is asymmetric.
  Configuration $\mathcal{C}'$ can be obtained by applying REDUCE_0 or REDUCE_{-1} on a configuration $\mathcal{C}''$ only if (i) $\mathcal{C}'' = (0, 1^{j+1}, 0, 1^{j'-1}, 0, 1, R', 0, 1^{j'-1})$, or (ii) $\mathcal{C}'' = \mathcal{C}$, or (iii) $\mathcal{C}'' = (0, 1^{j+1}, 0, 1^{j'-2}, 0, 1, R', 0, 1^{j'})$. In fact, in case (i) $\mathcal{C}'$ is obtained by performing REDUCE_0 on $\mathcal{C}''$ where

Algorithm: ALIGN-TWO-SYM
Input: Allowed configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$

if $\mathcal{C} \in S_2$ *or*
  $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, *for some* $\ell \geq 1$, *OR*
  $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, *for some* $\ell \geq 2$ *and* $1 \leq x \leq i$, *OR*
  $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, 0^i, 1, 0^y, 1)$, *for some* $1 \leq y < x \leq i$, *OR*
  $\mathcal{C} \in S_{4b} \cap S_3 \setminus \{(0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)\}$ **then**
    REDUCE$_{-1}(\mathcal{C})$;
else
    Let $\mathcal{C}'$ be the configuration obtained by applying REDUCE$_1(\mathcal{C})$ to one robot of $\mathcal{C}$;
    if $\mathcal{C}'$ *is symmetric* **then**
      REDUCE$_2(\mathcal{C})$;
    else
      Let $\mathcal{C}' = (0^i, 1^j, 0, R, 1)$, for $i \geq 2$ and $j \geq 1$;
      $\mathcal{C}^\alpha := (0^{i-1}, 1, 0, 1^{j-1}, 0, R, 1)$;
      $\mathcal{C}^\beta := (0^{i-1}, 1^j, 0, R, 0, 1)$;
      if $R = (1, R')$ **then** $\mathcal{C}^\gamma := (0^i, 1^{j+1}, 0, R', 1)$;
      if *At least two among* $\mathcal{C}^\alpha$, $\mathcal{C}^\beta$, *and* $\mathcal{C}^\gamma$ *are symmetric, the procedure from both of them corresponds to* REDUCE$_1$, *and* ($\mathcal{C} \in S_1 \setminus S_3$ *or* $\mathcal{C} = (0^i, 1, 0, 1, 0^x, 1, 0, 1)$, $x > 0$) **then**
        REDUCE$_2(\mathcal{C})$;
      else
        if $\mathcal{C}^{\min} = (0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)$ **then**
          Move the robot in v_2 to v_3;
        else
          REDUCE$_1(\mathcal{C})$;

Fig. 12: Second phase of Algorithm ALIGN for symmetric configurations.

Algorithm: ALIGN-TWO-ASYM
Input: Allowed configuration $\mathcal{C}$ with $\mathcal{C}^{\min} = (v_0, v_1, \ldots, v_{n-1})$

Let $\mathcal{C} = (0^i, 1^j, 0^x, 1^{j'}, R, 1)$ with $j \geq 1$, $x \geq 1$, and $j' \geq 0$;
$\mathcal{C}^\alpha := (0^{i-1}, 1, 0, 1^{j-1}, 0^x, 1^{j'}, R, 1)$;
$\mathcal{C}^\beta := (0^{i-1}, 1^j, 0^x, 1^{j'}, R, 0, 1)$;
if $j' > 0$ **then** $\mathcal{C}^\gamma := (0^i, 1^j, 0^{x-1}, 1, 0, 1^{j'-1}, R, 1)$;
if $R = (0, 1, R')$ **then** $\mathcal{C}^\delta := (0^i, 1^j, 0^x, 1^{j'+1}, 0, R', 1)$;
for $y \in \{\alpha, \beta, \gamma, \delta\}$ **do**
  if $\mathcal{C}^y$ *is symmetric and allowed* **then**
    if $\mathcal{C}^y$ *has no consecutive occupied nodes* **then**
      Let $\mathcal{C}'$ be the configuration obtained by executing ALIGN-ONE on $\mathcal{C}^y$;
    else
      Let $\mathcal{C}'$ be the configuration obtained by executing ALIGN-TWO-SYM on $\mathcal{C}^y$;
    if $\mathcal{C} = \mathcal{C}'$ **then**
      if *The robot that moved from* $\mathcal{C}^y$ *to* $\mathcal{C}$ *is not the one on the axis of symmetry of* $\mathcal{C}^y$ **then**
        Perform the move symmetrical to the one that is performed from $\mathcal{C}'$ to $\mathcal{C}$ and exit;
ASYM($\mathcal{C}$);

Fig. 13: Second phase of Algorithm ALIGN for asymmetric configurations.

$\mathcal{C}''^{\min} = (0, 1^{j+1}, 0, 1^{j'-1}, 0, 1, R', 0, 1^{j'-1})$, or by performing REDUCE$_{-1}$ on $\mathcal{C}''$ where $\mathcal{C}''^{\min} = (0, 1^{j'-1}, 0, 1, R', 0, 1^{j'-1}, 0, 1^{j+1})$; in case (ii) $\mathcal{C}'$ is obtained by performing REDUCE$_{-1}$ on $\mathcal{C}''$ where $\mathcal{C}''^{\min} = \mathcal{C}^{\min}$; and in case (iii), $\mathcal{C}'$ is obtained by performing REDUCE$_{-1}$ on $\mathcal{C}''$ where $\mathcal{C}''^{\min} = (0, 1^{j'}, 0, R', 1, 0, 1^{j'-2}, 0, 1^{j+1})$.

In the first case of (i), as the supermin of $\mathcal{C}''$ is $(0, 1^{j+1}, 0, 1^{j'-1}, 0, 1, R', 0, 1^{j'-1})$, we have that $(0, 1^{j+1}, 0, 1^{j'-1}, 0, 1, R', 0, 1^{j'-1}) \leq (0, 1^{j+1}, 0, 1^{j'-1}, 0, R', 1, 0, 1^{j'-1})$ which implies that $(1, R') \leq (R', 1)$. These last inequalities can be satisfied only if $R = (1^{j''})$ for some $j'' > j$, which implies that $k = 4$, a contradiction. In the second case of (i), since a REDUCE$_{-1}$ move has been performed, then we

must have that the axis passes through the middle of the first sequence of $j'-1$ consecutive empty nodes and that $(1, R', 0, 1^{j'-1}, 0, 1^{j+1})$ is a palindrome, that is $(R', 0, 1^{j'-1}, 0, 1^j) = (1^j, 0, 1^{j'-1}, 0, R')$. By Lemma 9, it follows that $R' = ((1^j, 0, 1^{j'-1}, 0)^\ell, 1^j)$ for some $\ell \geq 0$. However, in this case we have that $\mathcal{C}^{\min} = (0, 1^j, 0, 1^{j'}, 0, (1^j, 0, 1^{j'-1}, 0)^\ell, 1^j, 0, 1^{j'})$ which is a contradiction as: for $\ell > 0$, $\mathcal{C}^{\min}_{j+j'+2} < \mathcal{C}^{\min}$, and, for $\ell = 0$, $\mathcal{C}$ is periodic.
The case (ii) corresponds to the procedure that has been actually performed.
In case (iii), we have that the axis passes through the middle of the first sequence of j' consecutive empty nodes and that the sequence $(R', 1, 0, 1^{j'-2}, 0, 1^{j+1})$ is a palindrome, that is $(R', 1, 0, 1^{j'-2}, 0, 1^{j+1}) = (1^{j+1}, 0, 1^{j'-2}, 0, 1, R')$. By Lemma 9, this can occur only if $j = 0$ as otherwise the sequence $(1^{j+1}, 0, 1^{j'-2}, 0, 1)$ cannot be split into two palindromic sub-sequences. We obtained a contradiction as $j \geq 1$.

– $j'-1 = j$. We first prove that $\mathcal{C}'$ is asymmetric. In this case $\mathcal{C}' = (0, 1^j, 0, 1^{j+1}, 0, R', 1, 0, 1^j)$ and either ${\mathcal{C}'}^{\min} = (0, 1^j, 0, 1^j, 0, 1^{j+1}, 0, R', 1)$ or ${\mathcal{C}'}^{\min} = (0, 1^j, 0, 1^j, 0, 1, R', 0, 1^{j+1})$. In any case R' cannot contain a sequence $(0, 1^j, 0, 1^j)$ as otherwise the superminimality of $\mathcal{C}^{\min}$ is contradicted. Therefore, if ${\mathcal{C}'}^{\min} = (0, 1^j, 0, 1^j, 0, 1^{j+1}, 0, R', 1)$, the axis of symmetry can only pass through the first sequence of j consecutive empty nodes or in the robot separating the two sequences of j consecutive empty nodes. In the first case, R' must start with a $(1^{j-1}, 0)$ but this is a contradiction to the superminimality of $\mathcal{C}^{\min}$. In the second case, we must have that $(1^{j+1}, 0, R', 1) = (1, R', 0, 1^{j+1})$ that is $(1^j, 0, R') = (R', 0, 1^j)$. By Lemma 9, this implies that $R' = ((1^j, 0)^\ell, 1^j)$ for some $\ell \geq 0$. It follows that $\mathcal{C}^{\min} = (0, 1^j, 0, 1^{j+1}, 0, (1^j, 0)^\ell, 1^j, 0, 1^{j+1})$. However, this implies that: if $\ell = 0$, then $\mathcal{C}$ is periodic, and if $\ell > 0$, then $\mathcal{C}^{\min}_{2j+3} < \mathcal{C}^{\min}$. In any case, we obtain a contradiction.
If ${\mathcal{C}'}^{\min} = (0, 1^j, 0, 1^j, 0, 1, R', 0, 1^{j+1})$, the axis of symmetry can only pass through the robot separating the two sequences of j consecutive empty nodes. Note that, in this case, ${\mathcal{C}'}^{\min} = (0, 1^j, 0, 1^j, 0, 1, R', 0, 1^{j+1}) = (0, 1^j, 0, 1^j, 0, 1^{j+1}, 0, R', 1)$ and hence the same arguments as before can be applied.
Configuration $\mathcal{C}'$ can be obtained by applying REDUCE$_0$ or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if $\mathcal{C}'' = (0, 1^{j+1}, 0, 1^j, 0, 1, R', 0, 1^j)$. In fact, $\mathcal{C}'$ can be obtained by applying REDUCE$_0$ or REDUCE$_{-1}$ to $\mathcal{C}''$ if ${\mathcal{C}''}^{\min} = (0, 1^{j+1}, 0, 1^j, 0, 1, R', 0, 1^j)$ or ${\mathcal{C}''}^{\min} = (0, 1^j, 0, 1, R', 0, 1^j, 0, 1^{j+1})$, respectively. The first case is impossible as ${\mathcal{C}''}^{\min}$ is not minimum. In the second case, we must have that $(1, R', 0, 1^j, 0, 1^{j+1}) = (1^{j+1}, 0, 1^j, 0, R', 1)$ and hence $(R', 0, 1^j, 0, 1^j) = (1^j, 0, 1^j, 0, R')$. By Lemma 9, this implies that $R' = ((1^j, 0, 1^j, 0)^\ell, 1^j)$ or $R' = ((1^j, 0, 1^j, 0)^\ell, 1^j, 0, 1^j)$. In any case, we obtain that $k = 4$, or $\mathcal{C}$ is periodic, or ${\mathcal{C}'}^{\min}$ is not minimum. □

Lemma 11 (Lemma 4) *Let $\mathcal{C}$ be a symmetric and allowed configuration with supermin $\mathcal{C}^{\min} = (0^i, 1, R)$, $i > 1$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_1$ *on only one robot on $\mathcal{C}$. If $\mathcal{C}'$ is symmetric, then $\mathcal{C} \in S_1 \backslash S_3$ or $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, for some $\ell \geq 1$, or $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $\ell \geq 1$ and $1 \leq x \leq i$.*

Proof We first show that $\mathcal{C} \in S_1$ and then that the only configuration in S_3 that leads to a symmetric $\mathcal{C}'$ is such that $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^\ell)$, for some $\ell \geq 1$, or $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^\ell)$, for some $\ell \geq 1$ and $1 \leq x \leq i$.

Let $\mathcal{C}^{\min} = (0^i, 1^j, 0, R')$ for $j \geq 1$, then $S = (0^i, 1^{j-1}, 0, 1, R')$ is a representation of $\mathcal{C}'$. We show that if $j > 1$, then ${\mathcal{C}'}^{\min} = (0^i, 1^{j-1}, 0, 1, R')$ is the unique supermin of $\mathcal{C}'$, i.e. the configuration is asymmetric. Note that $S < \mathcal{C}^{\min}$ and that $\overline{S_h} > \overline{\mathcal{C}^{\min}_h} \geq \mathcal{C}^{\min}$ for each $h \neq \ell_1$. Moreover, $\overline{S_{\ell_1}} > S$. Therefore, $\overline{S_h}$ cannot be a supermin of $\mathcal{C}'$ for each h. To obtain a contradiction, let j' be an integer such that $S_{j'} < S$. Then $\mathcal{C}^{\min}_{j'} < \mathcal{C}^{\min}$, a contradiction.

It follows that if $\mathcal{C}'$ is symmetric, then $\mathcal{C}^{\min} = (0^i, 1, 0, R')$ and the supermin of C' is $(0^{i+1}, 1, R')$, or $(0^{i+1}, \overline{R'}, 1)$ or both. In this case, $\mathcal{C}'$ has an axis of symmetry passing through the middle of the unique sequence of $i+1$ consecutive occupied nodes. This implies that $(0^{i+1}, 1, R') = (0^{i+1}, \overline{R'}, 1)$. Let us assume that the axis of symmetry of $\mathcal{C}$ passes through the middle of the initial sequence of i consecutive occupied nodes. Then, the sequence $(1, 0, R')$ is a palindrome and then $R' = (R'', 0, 1)$ with $R'' = \overline{R''}$. Since, by the symmetry of $\mathcal{C}'$, $(1, R') = (\overline{R'}, 1)$, then $(1, R'', 0, 1) = (1, 0, R'', 1)$ and therefore $(R'', 0) = (0, R'')$. By

Lemma 8, $R'' = (0^{j'})$ for some j' and then $\mathcal{C}^{\min} = (0^i, 1, 0^{j'+2}, 1)$ which is a contradiction as it implies that $k = n - 2$. Therefore, the axis of symmetry of $\mathcal{C}$ does not pass through the middle of the initial sequence of i consecutive occupied nodes and therefore there is another such sequence. Therefore, $\mathcal{C} \in S_1$.

We now consider the possibility that $\mathcal{C} \in S_3$.

Since $\mathcal{C}'$ is asymmetric if $\mathcal{C}^{\min} = (0^i, 1^j, 0, R')$ for any R' and $j > 1$, we now show that it is asymmetric if $\mathcal{C}^{\min} = (0^i, 1, 0^{i-1}, 1, 0, R, 1)$, for some $R = \overline{R}$. Let us assume that $\mathcal{C}^{\min} = (0^i, 1, 0^{i-1}, 1, 0, R, 1)$ with $R = \overline{R}$, then $\mathcal{C}' = (0^{i+1}, 1, 0^{i-2}, 1, 0, R, 1)$. Since there is only one sequence of $i+1$ consecutive occupied nodes, then $\mathcal{C}'$ can only have an axis of symmetry passing through the initial sequence of consecutive occupied nodes, and hence $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0^{i-2}, 1, 0, R, 1)$ and the sequence $(1, 0^{i-2}, 1, 0, R, 1)$ is a palindrome. By Lemma 9, $\mathcal{C}'$ is symmetric if and only if $R = ((0^{i-2}, 1, 0)^{\ell}, 0^{i-3})$, with $i \geq 3$, that is $\mathcal{C}^{\min} = (0^i, 1, 0^{i-1}, 1, 0, (0^{i-2}, 1, 0)^{\ell}, 0^{i-3}, 1)$ and then $\mathcal{C}$ is asymmetric.

Let us now assume that $\mathcal{C}^{\min} = (0^i, 1^{j'}, 0^x, 1^j, 0^x, 1^{j'}, 0^i, Z)$ with $Z = \overline{Z}$, $j, j' > 0$, and $1 \leq x \leq i$. We can assume that $j' = 1$ as otherwise $\mathcal{C}'$ is asymmetric. After applying REDUCE$_1$, we have $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$ with $(1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$ palindromic, that is $(1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z) = (Z, 0^i, 1, 0^x, 1^j, 0^{x-1}, 1)$. By Lemma 9 it follows that $Z = ((1, 0^i, 1^j, 0^{i+1}, 1, 0^i)^{\ell}, 1)$, for some $\ell \geq 0$ and $x - 1 = i$; or $Z = ((1, 0^{i-1}, 1, 0^i, 1, 0^i)^{\ell}, 1, 0^{i-1}, 1)$, for some $\ell \geq 0$, $x = i$, and $j = 1$; or $Z = ((1^{j+1}, 0, 1, 0^i)^{\ell}, 1^{j+1})$, for some $\ell \geq 0$, $x = 1$, and $i = 1$. In the first case we obtain a contradiction with the hypothesis that $x \leq i$; in the second case, we have that $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, 0^i, 1, 0^i, (1, 0^{i-1}, 1, 0^i, 1, 0^i)^{\ell}, 1, 0^{i-1}, 1) = (0^i, 1, 0^i, 1, (0^i, 1, 0^i, 1, 0^{i-1}, 1)^{\ell+1})$, for $\ell \geq 0$; in the third case we obtain a contradiction with the hypothesis that $i > 1$.

Finally, let us now assume that $\mathcal{C}^{\min} = (0^i, 1^{j'}, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j' > 0$ and $1 \leq x \leq i$. Also in this case, we can assume that $j' = 1$ as otherwise $\mathcal{C}'$ is asymmetric. After applying REDUCE$_1$, we have $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0^{x-1}, 1, 0^i, Z)$ with $(1, 0^{x-1}, 1, 0^i, Z)$ palindromic. By Lemma 9 it follows that $Z = ((1, 0^{x-1}, 1, 0^i)^{\ell}, 1)$, for some $\ell \geq 0$ and $x - 1 = i$; or $Z = ((1, 0^{x-1}, 1, 0^i)^{\ell}, 1, 0^{x-1}, 1)$, for some $\ell \geq 0$. In the first case we obtain a contradiction with the hypothesis that $x \leq i$; in the second case, we have that $\mathcal{C}^{\min} = (0^i, 1, 0^x, 1, 0^i, (1, 0^{x-1}, 1, 0^i)^{\ell}, 1, 0^{x-1}, 1) = (0^i, 1, 0^x, 1, (0^i, 1, 0^{x-1}, 1)^{\ell+1})$, for $\ell \geq 0$. □

Lemma 12 (Lemma 5) *Let $\mathcal{C}$ be a symmetric and allowed configuration with $i > 1$ consecutive occupied nodes and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_1$ *on only one robot on $\mathcal{C}$. If $\mathcal{C}'$ is adjacent with respect to* REDUCE$_1$ *to a symmetric configuration $\mathcal{C}''$ different from $\mathcal{C}$, then $\mathcal{C} \in S_1 \setminus S_3$, or $\mathcal{C}'' \in S_1 \setminus S_3$ or $\mathcal{C}, \mathcal{C}'' \in S_4$.*

Proof Let $\mathcal{C}$ and $\mathcal{C}''$ be two different symmetric configurations and let $\mathcal{C}'$ and $\mathcal{C}'''$ be the configuration obtained by applying REDUCE$_1$ on only one robot from $\mathcal{C}$ and $\mathcal{C}''$, respectively. We show that if $\mathcal{C}' = \mathcal{C}'''$, then $\mathcal{C} \in S_1 \setminus S_3$, or $\mathcal{C}'' \in S_1 \setminus S_3$ or $\mathcal{C}' \in S'_4$. We show the following equivalent statement: if $\mathcal{C}' = \mathcal{C}'''$ and neither $\mathcal{C}$ nor $\mathcal{C}''$ belong to $S_1 \setminus S_3$, then $\mathcal{C}' \in S'_4$. The premise is satisfied when one of the following cases holds:

A. $\mathcal{C} \notin S_1$ and $\mathcal{C}'' \notin S_1$;
B. $\mathcal{C} \notin S_1$ and $\mathcal{C}'' \in S_3$;
C. $\mathcal{C} \in S_3$ and $\mathcal{C}'' \notin S_1$;
D. $\mathcal{C} \in S_3$ and $\mathcal{C}'' \in S_3$.

We observe that case B is equivalent to case C and analyze the three cases separately.

A. If a symmetric configuration $\mathcal{C}$ with $i \geq 2$ consecutive occupied nodes does not belong to S_1, then $\mathcal{C}^{\min} = (0^i, 1^j, 0, R, 0, 1^j)$ for some $j > 0$ and $R = \overline{R}$ (that is the axis of symmetry passes through the middle of the sequence of i occupied nodes and the middle of R) or $\mathcal{C}^{\min} = (0^i, 1^j, 0, 1^j)$ for some $j > 1$.
   We first analyze the case when both the configurations $\mathcal{C}$ and $\mathcal{C}''$ belong to the first type. In this case $\mathcal{C}^{\min} = (0^i, 1^j, 0, R, 0, 1^j)$ and $\mathcal{C}''^{\min} = (0^{i'}, 1^{j'}, 0, R', 0, 1^{j'})$, for some $j, j' > 0$, $R = \overline{R}$, and $R' = \overline{R'}$. The configurations obtained after applying REDUCE$_1$ on only one robot from $\mathcal{C}$ and $\mathcal{C}''$ are $\mathcal{C}' = (0^i, 1^{j-1}, 0, 1, R, 0, 1^j)$ and $\mathcal{C}''' = (0^{i'}, 1^{j'-1}, 0, 1, R', 0, 1^{j'})$, respectively. Four cases arise:
   - $j > 1$ and $j' > 1$. In this case $\mathcal{C}'^{\min} = (0^i, 1^{j-1}, 0, 1, R, 0, 1^j)$ and $\mathcal{C}'''^{\min} = (0^{i'}, 1^{j'-1}, 0, 1, R', 0, 1^{j'})$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$, $j = j'$, and $R = R'$, which implies that $\mathcal{C} = \mathcal{C}''$, a contradiction.
   - $j = 1$ and $j' > 1$. In this case $\mathcal{C}' = (0^{i+1}, 1, R, 0, 1)$ and $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0, R, 1)$, while $\mathcal{C}'''^{\min} = (0^{i'}, 1^{j'-1}, 0, 1, R', 0, 1^{j'})$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i' = i + 1$, $j' - 1 = 1$,

and $R = (1, R', 0, 1)$. Since $R = \overline{R}$, then $(R', 0) = (0, R')$ which, by Lemma 8, implies that $R' = (0^y)$, for some $y \geq 0$. Summarizing, $\mathcal{C}^{\min} = (0^i, 1, 0, 1, 0^{y+1}, 1, 0, 1)$, $\mathcal{C}''^{\min} = (0^{i+1}, 1, 1, 0^{y+2}, 1, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0^{i+1}, 1, 0, 1, 0^{y+1}, 1, 1)$, for some $0 \leq y \leq i-2$, that is, $\mathcal{C} \in S_{4b}$, $\mathcal{C}'' \in S_{4a}$, and $\mathcal{C}' \in S'_4$.

- $j > 1$ and $j' = 1$. This case is equivalent to the previous one.
- $j = 1$ and $j' = 1$. In this case $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0, R, 1)$ and $\mathcal{C}'''^{\min} = (0^{i'+1}, 1, 0, R', 1)$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$ and $R = R'$, which implies that $\mathcal{C} = \mathcal{C}''$, a contradiction.

If $\mathcal{C}^{\min} = (0^i, 1^j, 0, 1^j)$ and $\mathcal{C}''^{\min} = (0^{i'}, 1^{j'}, 0, 1^{j'})$ for some $j, j' > 1$, then $\mathcal{C}'^{\min} = (0^i, 1^{j-1}, 0, 1^{j+1})$ and $\mathcal{C}'''^{\min} = (0^{i'}, 1^{j'-1}, 0, 1^{j'+1})$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$ and $j = j'$, which implies that $\mathcal{C} = \mathcal{C}''$, a contradiction.

If $\mathcal{C}^{\min} = (0^i, 1^j, 0, 1^j)$ and $\mathcal{C}''^{\min} = (0^{i'}, 1^{j'}, 0, R', 0, 1^{j'})$, for some $j > 1$, $j' > 0$, and $R' = \overline{R'}$, then two cases arise.

- $j' > 1$. In this case, $\mathcal{C}'^{\min} = (0^i, 1^{j-1}, 0, 1^{j+1})$ and $\mathcal{C}'''^{\min} = (0^{i'}, 1^{j'-1}, 0, 1, R', 0, 1^{j'})$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$, $j = j'$ and $(1, R', 0, 1^j) = (1^{j+1})$, a contradiction.
- $j' = 1$. In this case, $\mathcal{C}'^{\min} = (0^i, 1^{j-1}, 0, 1^{j+1})$, $\mathcal{C}''' = (0^{i'+1}, 1, R', 0, 1)$, and $\mathcal{C}'''^{\min} = (0^{i'+1}, 1, 0, R', 1)$. Therefore, if $\mathcal{C}' = \mathcal{C}'''$, then $i = i' + 1$ and $(1^{j-1}, 0, 1^{j+1}) = (1, 0, R', 1)$. This implies that $j - 1 = 1$ and $R' = (1^j)$. In conclusion, $\mathcal{C}^{\min} = (0^i, 1, 1, 0, 1, 1)$, $\mathcal{C}''^{\min} = (0^{i-1}, 1, 0, 1, 1, 0, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0^i, 1, 0, 1, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$.

B. In this case $\mathcal{C}^{\min} = (0^{i'}, 1^{j''}, 0, R, 0, 1^{j''})$, for some $j'' > 0$ and $R = \overline{R}$ (we exclude the case $\mathcal{C}^{\min} = (0^i, 1^j, 0, 1^j)$ since it always generates a contradiction). For $\mathcal{C}''$ we have four cases

(a) $\mathcal{C}''^{\min} = (0^i, 1^{j'}, 0^x, 1^j, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j, j' > 0$ and $1 \leq x \leq i$;

(b) $\mathcal{C}''^{\min} = (0^i, 1^{j'}, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j' > 0$ and $1 \leq x \leq i$;

(c) $\mathcal{C}''^{\min} = (0^i, 1^j, 0^{i-1}, 1, 0, R, 1)$, $R = \overline{R}$, $j > 0$.

We give full details on the first case, the other cases can be shown by similar arguments and therefore, we only state the conditions that do not lead to a contradiction.

(a) In this case $\mathcal{C}' = (0^{i'}, 1^{j''-1}, 0, 1, R, 0, 1^{j''})$ and $\mathcal{C}''' = (0^i, 1^{j'-1}, 0, 1, 0^{x-1}, 1^j, 0^x, 1^{j'}, 0^i, Z)$ and four cases may arise:

- $j'' = 1$ and $j' > 1$. In this case, $\mathcal{C}'^{\min} = (0^{i'+1}, 1, 0, R, 1)$ and $\mathcal{C}'''^{\min} = (0^i, 1^{j'-1}, 0, 1, 0^{x-1}, 1^j, 0^x, 1^{j'}, 0^i, Z)$. If $\mathcal{C}' = \mathcal{C}'''$, then $i = i' + 1$, $j' = 2$, and $R = (1, 0^{x-1}, 1^j, 0^x, 1, 1, 0^i, 1, Z')$, where Z' is such that $Z = (1, Z', 1)$ and $Z' = \overline{Z'}$. By plugging R into $\mathcal{C}$ we obtain a contradiction because $\mathcal{C}$ cannot contain a sequence of $i = i' + 1$ consecutive occupied nodes.
- $j'' = 1$ and $j' = 1$. In this case, $\mathcal{C}'^{\min} = (0^{i'+1}, 1, 0, R, 1)$ and $\mathcal{C}''' = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$. The supermin of $\mathcal{C}'''$ is $\mathcal{C}'''^{\min} = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$ or $\mathcal{C}'''^{\min} = (0^{i+1}, Z, 0^i, 1, 0^x, 1^j, 0^{x-1}, 1)$. In both cases , if $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$. In the former case, we have that $(R, 1) = (0^{x-2}, 1^j, 0^x, 1, 0^i, Z)$ which is a contradiction since $\mathcal{C} \notin S_1$ and therefore it cannot contain a sequence of $i = i'$ consecutive occupied nodes. In the latter case, we have that $(1, 0, R) = (Z, 0^i, 1, 0^x, 1^j, 0^{x-1})$ and either R contains a sequence of $i = i'$ consecutive occupied nodes or $Z = (1)$ and $R = (0^{i-1}, 1, 0^x, 1^j, 0^{x-1})$. However, R is a palindrome only if $x = i$, and again it contains a a sequence of $i = i'$ consecutive occupied nodes.
- $j'' > 1$ and $j' > 1$. In this case, $\mathcal{C}' = (0^{i'}, 1^{j''-1}, 0, 1, R, 0, 1^{j''})$ and $\mathcal{C}''' = (0^i, 1^{j'-1}, 0, 1, 0^{x-1}, 1^j, 0^x, 1^{j'}, 0^i, Z)$. If $\mathcal{C}' = \mathcal{C}'''$, then $i = i'$, $j'' = j'$, and $(R, 0, 1^{j''}) = (0^{x-1}, 1^j, 0^x, 1^{j'}, 0^i, Z)$. In any case we obtain that R contains a sequence of $i = i'$ consecutive occupied nodes, a contradiction.
- $j'' > 1$ and $j' = 1$. In this case, $\mathcal{C}'^{\min} = (0^{i'}, 1^{j''-1}, 0, 1, R, 0, 1^{j''})$ and $\mathcal{C}''' = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$. The supermin of $\mathcal{C}'''$ is $\mathcal{C}'''^{\min} = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$ or $\mathcal{C}'''^{\min} = (0^{i+1}, Z, 0^i, 1, 0^x, 1^j, 0^{x-1}, 1)$, in both cases, $i' = i + 1$.
  - If $\mathcal{C}'''^{\min} = (0^{i+1}, 1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$, then we must have $(1^{j''-1}, 0, 1, R, 0, 1^{j''}) =$

$(1, 0^{x-1}, 1^j, 0^x, 1, 0^i, Z)$. The following cases arise. (1) $x - 1 = 0$, $j'' - 1 = j + 1$ and $(R, 0, 1^{j''}) = (0^i, Z)$, which implies that $\mathcal{C}^{\min} = (0^{i+1}, 1^{j''}, 0^{i+1}, Z)$ which is a contradiction as it contains a sequence of $i+1 = i'$ consecutive occupied nodes and therefore it belongs to S_1. (2) $j'' - 1 = 1$, $x - 1 = 1$, which implies that $(R, 0, 1, 1) = (1^{j-1}, 0, 0, 1, 0^i, Z)$. We obtain the following sub-cases. (2a) $Z = (1, 1)$, which implies that $R = (1^{j-1}, 0, 0, 1, 0^{i-1})$ and, since $R = \overline{R}$, that $j = 1$ and $i = 3$. Therefore, $\mathcal{C}^{\min} = (0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1)$, $\mathcal{C}''^{\min} = (0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$. (2b) $Z = (1, 1, 0, 1, 1)$, which implies that $R = (1^{j-1}, 0, 0, 1, 0^i, 1, 1)$ and, since $R = \overline{R}$, that $j = 3$ and $i = 2$. Therefore, $\mathcal{C}''^{\min} = (0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1)$ which is a contradiction as $\mathcal{C}''^{\min}_8 < \mathcal{C}''^{\min}$. (2c) $Z = (1, 1, 0, Z', 0, 1, 1)$ with $Z' = \overline{Z'}$, which implies that $R = (1^{j-1}, 0, 0, 1, 0^i, 1, 1, 0, Z')$. We exploit Lemma 9 to compute a closed sequence for R and Z'. The only way to divide the sequence $(1^{j-1}, 0, 0, 1, 0^i, 1, 1, 0)$ into two palindromic subsequences U and V is $U = (0, 0, 1, 0, 0)$ and $V = (0, 1, 1, 0)$, where $i = 3$ and $j = 1$. We obtain that $Z' = ((0, 0, 1, 0, 0, 0, 1, 1, 0)^\ell, 0, 0, 1, 0, 0)$ for some $\ell \geq 0$ and $\mathcal{C}''^{\min} = (0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, (0, 0, 1, 0, 0, 0, 0, 1, 1, 0)^\ell, 0, 0, 1, 0, 0, 0, 1, 1)$ which is a contradiction as $\mathcal{C}''^{\min}_{15} < \mathcal{C}''^{\min}$.

– If $\mathcal{C}'''^{\min} = (0^{i+1}, Z, 0^i, 1, 0^x, 1^j, 0^{x-1}, 1)$, then we must have that $(1^{j''-1}, 0, 1, R, 0, 1^{j''}) = (Z, 0^i, 1, 0^x, 1^j, 0^{x-1}, 1)$. Since $j'' > 1$, then $x - 1 = 0$ and $j + 1 = j''$. Hence, $(1^j, 0, 1, R) = (Z, 0^i, 1)$, two cases may arise. (1) $R = (1, 0^i, 1)$ and $Z = (1^j, 0, 1, 1)$. Since $Z = \overline{Z}$, then $j = 2$. Therefore, $\mathcal{C}^{\min} = (0^{i+1}, 1, 1, 1, 0, 1, 0^i, 1, 0, 1, 1, 1)$, $\mathcal{C}''^{\min} = (0^i, 1, 0, 1, 1, 0, 1, 0^i, 1, 1, 0, 1, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0^{i+1}, 1, 1, 0, 1, 1, 0^i, 1, 0, 1, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$. (2) $R = (1, 0^i, R', 0^i, 1)$ for some $R' = \overline{R'}$ and $Z = (1^j, 0, 1, 1, 0^i, R')$. Again we exploit Lemma 9 to compute a closed sequence for R and Z. The only way to divide the sequence $(1^j, 0, 1, 1, 0^i)$ into two palindromic subsequences U and V is $U = (1, 1, 0, 1, 1)$ and $V = (0^i)$, where $j = 2$. We obtain that $R' = ((1, 1, 0, 1, 1, 0^i)^\ell, 1, 1, 0, 1, 1)$, $R = (1, 0^i, (1, 1, 0, 1, 1, 0^i)^\ell, 1, 1, 0, 1, 1, 0^i, 1) = (1, 0^i, (1, 1, 0, 1, 1, 0^i)^{\ell+1}, 1)$, and $Z = ((1, 1, 0, 1, 1, 0^i)^{\ell+1}, 1, 1, 0, 1, 1)$, for some $\ell \geq 0$. Therefore, $\mathcal{C}^{\min} = (0^{i+1}, 1, 1, 1, 0, 1, 0^i, (1, 1, 0, 1, 1, 0^i)^{\ell+1}, 1, 0, 1, 1, 1)$, $\mathcal{C}''^{\min} = (0^i, 1, 0, 1, 1, 0, 1, 0^i, (1, 1, 0, 1, 1, 0^i)^{\ell+1}, 1, 1, 0, 1, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0^{i+1}, 1, 1, 0, 1, 1, 0^i, (1, 1, 0, 1, 1, 0^i)^{\ell+1}, 1, 0, 1, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$.

(b) In this case $\mathcal{C}' = (0^{i'}, 1^{j''-1}, 0, 1, R, 0, 1^{j''})$ and $\mathcal{C}''' = (0^i, 1^{j'-1}, 0, 1, 0^{x-1}, 1^{j'}, 0^i, Z)$. By using arguments similar to those used in the previous case, we obtain that if $\mathcal{C}' = \mathcal{C}'''$, then one of the following cases is possible.

– $i' = 3$, $j'' = 2$, $R = (0, (1, 0, 1, 0)^\ell, 1, 0)$, $i = 2$, $j' = 1$, $x = 1$, and $Z = ((1, 0, 1, 0)^{\ell+1}, 1)$, for some $\ell \geq 0$. Therefore, $\mathcal{C}^{\min} = (0, 0, 0, 1, 1, 0, 0, (1, 0, 1, 0)^\ell, 1, 0, 0, 1, 1) = (0, 0, 0, 1, 1, 0, (0, 1)^{2\ell+1}, 0, 0, 1, 1)$, $\mathcal{C}''^{\min} = (0, 0, 1, 0, 1, 0, 0, (1, 0, 1, 0)^{\ell+1}, 1) = (0, 0, 1, 0, 1, 0, (0, 1)^{2\ell+2}, 0, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0, 0, 0, 1, 0, 1, 0, (1, 0, 1, 0)^\ell, 1, 0, 0, 1, 1) = (0, 0, 0, 1, 0, 1, (0, 1)^{2\ell+1}, 0, 0, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$.

– $i' = i + 1$, $j'' = 2$, $R = (0^{i-1}, (1, 0, 1, 0^{i-1})^\ell, 1, 0, 1, 0^{i-1})$,

$j' = 1$, $x = 1$, and $Z = ((1,0,1,0^{i-1})^{\ell+1}, 1, 0, 1)$, for some $\ell \geq 0$. Therefore, $\mathcal{C}^{\min} = (0^{i+1}, 1, 1, 0^i, (1,0,1,0^{i-1})^{\ell}, 1, 0, 1, 0^i, 1, 1)$, $\mathcal{C}''^{\min} = (0^i, 1, 0, 1, 0^i, (1,0,1,0^{i-1})^{\ell+1}, 1, 0, 1)$, and $\mathcal{C}' = \mathcal{C}''' = (0^{i+1}, 1, 0, 1, 0^{i-1}, (1,0,1,0^{i-1})^{\ell}, 1, 0, 1, 0^i, 1, 1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$.

(c) In this case $\mathcal{C}' = (0^{i'}, 1^{j''-1}, 0, 1, R, 0, 1^{j''})$ and $\mathcal{C}''' = (0^i, 1^{j-1}, 0, 1, 0^{i-2}, 1, 0, R', 1)$. By using arguments similar to those used in the previous case, we obtain that if $\mathcal{C}' = \mathcal{C}'''$, then $i = 2$, $i' = 3$, $j = 1$, $j'' = 2$, $R = ((0,1)^{\ell}, 0)$, $R' = ((0,1)^{\ell+1}, 0)$, for some $\ell \geq 0$. Therefore, $\mathcal{C}^{\min} = (0,0,0,1,1,0,(0,1)^{\ell},0,0,1,1)$, $\mathcal{C}''^{\min} = (0,0,1,0,1,0,(0,1)^{\ell+1},0,1)$, and $\mathcal{C}' = \mathcal{C}''' = (0,0,0,1,0,1,(0,1)^{\ell},0,0,1,1)$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$.

C. This case is symmetrical to the previous one.

D. Let us define the following three configurations $\mathcal{C}^r = (0^i, 1^{j'}, 0^x, 1^j, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j, j' > 0$ and $1 \leq x \leq i$; $\mathcal{C}^s = (0^i, 1^{j'}, 0^x, 1^{j'}, 0^i, Z)$ for some $Z = \overline{Z}$, $j' > 0$ and $1 \leq x \leq i$; and $\mathcal{C}^t = (0^i, 1^j, 0^{i-1}, 1, 0, R, 1)$, $R = \overline{R}$, $j > 0$. The following cases arise:

(a) $\mathcal{C} = \mathcal{C}^r$ and $\mathcal{C}'' = \mathcal{C}^r$;
(b) $\mathcal{C} = \mathcal{C}^r$ and $\mathcal{C}'' = \mathcal{C}^s$;
(c) $\mathcal{C} = \mathcal{C}^r$ and $\mathcal{C}'' = \mathcal{C}^t$;
(d) $\mathcal{C} = \mathcal{C}^s$ and $\mathcal{C}'' = \mathcal{C}^s$;
(e) $\mathcal{C} = \mathcal{C}^s$ and $\mathcal{C}'' = \mathcal{C}^t$;
(f) $\mathcal{C} = \mathcal{C}^t$ and $\mathcal{C}'' = \mathcal{C}^t$.

Since the arguments used to analyze these cases are similar to those already used, we only give the conditions that do not lead to a contradiction. In particular, the only case that do not lead to a contradiction is case (b) when $\mathcal{C} = (0^i, 1, 0^i, 1, 0^i, 1, 0^i, 1, 0^{i-1}, 1, 0^i, (1, 0^{i-1}, 1, 0^i, 1, 0^i, 1, 0^{i-1}, 1, 0^i)^{\ell}, 1, 0^{i-1}, 1)$, $\mathcal{C}'' = (0^i, 1, 0^i, 1, 0^i, (1, 0^{i-1}, 1, 0^i, 1, 0^i, 1, 0^{i-1}, 1, 0^i)^{\ell}, 1, 0^{i-1}, 1, 0^i, 1, 0^i, 1, 0^{i-1}, 1)$, for some $\ell > 0$, that is, $\mathcal{C} \in S_{4a}$, $\mathcal{C}'' \in S_{4b}$, and $\mathcal{C}' \in S'_4$. □

Lemma 13 (Lemma 6) *Let $\mathcal{C}$ be a configuration in $S_1 \setminus S_3$ with supermin $\mathcal{C}^{\min} = (0^i, 1, R)$, $i > 1$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_2$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_1$ *or* REDUCE$_2$ *to any symmetric configuration different from $\mathcal{C}$.*

Proof We first show that $\mathcal{C}'$ is asymmetric and that cannot it be obtained by applying REDUCE$_1$ on a configuration different from $\mathcal{C}$

Note that if $\mathcal{C}'^{\min} = (0^i, 1^j, 0, X, 1)$, for some $j \geq 1$, then it can be obtained by performing REDUCE$_1$ on a configuration $\mathcal{C}''$ such that (i) $\mathcal{C}'' = (0^{i-1}, 1, 0, 1^{j-1}, 0, X, 1)$, or (ii) $\mathcal{C}'' = (0^{i-1}, 1^j, 0, X, 0, 1)$, or (iii) $\mathcal{C}'' = (0^i, 1^{j+1}, 0, X', 1)$, where this last case can occur only if $X = (1, X')$ for some X'.

We obtain the following cases.

- $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, 0, R')$. In this case, $\mathcal{C}'$ has a representation $\mathcal{C}' = (0^i, 1, 0^{i+1}, 1, R')$ and, since there is only one sequence of $i+1$ consecutive occupied nodes, the axis of symmetry of $\mathcal{C}'$ passes through such a sequence and $\mathcal{C}'^{\min} = (0^{i+1}, 1, R', 0^i, 1) = (0^{i+1}, 1, 0^i, \overline{R'}, 1)$. However, this implies that R' starts with 0^i which is a contradiction to the superminimality of $\mathcal{C}^{\min}$ as in this case $\mathcal{C}^{\min}$ contains a sequence of $i+1$ consecutive occupied nodes. This also implies that $\mathcal{C}'^{\min} = (0^{i+1}, 1, 0^i, \overline{R'}, 1)$.
  If C' can been obtained by applying REDUCE$_1$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$, then (i) $\mathcal{C}'' = (0^i, 1, 0^{i+1}, \overline{R'}, 1)$, or (ii) $\mathcal{C}'' = (0^i, 1, 0^i, \overline{R'}, 0, 1)$. The case (iii) cannot occur as in this case X starts with 0. In both cases the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_1$, a contradiction.
- $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1^j, 0, R')$, $j > 1$. In this case, $\mathcal{C}'^{\min} = (0^i, 1, 0^i, 1^{j-1}, 0, 1, R')$. Moreover, such sequence is the only supermin sequence as otherwise we obtain a contradiction to the superminimality of $\mathcal{C}^{\min}$. Therefore, $\mathcal{C}'$ is asymmetric. If $\mathcal{C}'$ has been obtained by applying REDUCE$_1$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$, then (i) $\mathcal{C}'' = (0^{i-1}, 1, 0^{i+1}, 1^{j-1}, 0, 1, R')$, or (ii) $\mathcal{C}'' = (0^{i-1}, 1, 0^i, 1^{j-1}, 0, 1, R'', 0, 1)$ with $R'' = (R', 1)$. The case (iii) cannot occur as in this case X starts with 0. In both cases (i) and (ii) the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_1$, a contradiction.
- $\mathcal{C}^{\min} = (0^i, 1^j, 0^x, 1^{j'}, 0, R')$, $x \leq i$, $j > 0$, and $j' > 0$. We exclude the case $x = i$ and $j = 1$ because it has been already analyzed. In this case, $\mathcal{C}' = (0^i, 1^j, 0^x, 1^{j'-1}, 0, 1, R')$.
  If $x < i - 1$ or $j' > 1$, then $\mathcal{C}'^{\min} = (0^i, 1^j, 0^x, 1^{j'-1}, 0, 1, R')$. Moreover, such sequence is the only supermin sequence as otherwise we obtain a contradiction to the superminimality of $\mathcal{C}^{\min}$. Therefore, $\mathcal{C}'$ is asymmetric. If $\mathcal{C}'$ has been obtained by applying REDUCE$_1$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$, then (i) $\mathcal{C}'' =$

$(0^{i-1},1,0,1^{j-1},0^x,1^{j'-1},0,1,R')$, (ii) $\mathcal{C}'' = (0^{i-1},1^j,0^x,1^{j'-1},0,1,R'',0,1)$ with $R' = (R'',1)$, or (iii) $\mathcal{C}'' = (0^i,1^{j+1},0,1^{j'-2},0,1,R')$ where $x = 1$ and $j' > 1$ as otherwise such case cannot occur. Since $\mathcal{C} \in S_1$, in cases (i) and (ii) R' contains a sequence of i consecutive occupied nodes and hence the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_1$, a contradiction. In case (iii) $\mathcal{C}''^{\min} \neq (0^i,1^{j+1},0,1^{j'-2},0,1,R')$ as the superminimality of $\mathcal{C}$ implies that either R' contains a sequence $(0,1^j,0^i)$ or it finishes by $(0,1^j)$. Therefore also in this case, the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_1$.

If $x = i-1$ and $j' = 1$, then $\mathcal{C}^{\min} = (0^i,1^j,0^{i-1},1,0,R')$ then $\mathcal{C} \in S_3$ since R' must finish by 1.

If $x = i$, $j' = 1$, and $j > 1$, then $\mathcal{C}^{\min} = (0^i,1^j,0^i,1,0,R')$ is a contradiction as $(\mathcal{C}^{\min})_{i+j} < \mathcal{C}^{\min}$.

We conclude the proof by showing that $\mathcal{C}'$ cannot be obtained by applying REDUCE$_2$ on a configuration different from $\mathcal{C}$.

Let us assume that $\mathcal{C}^{\min} = (0^i,1^j,0^x,1^{j'},0^y,1,X)$. After performing REDUCE$_2$ on $\mathcal{C}$ we have $\mathcal{C}' = (0^i,1^j,0^x,1^{j'-1},0,1,0^{y-1},1,X)$. Let us assume that $\mathcal{C}'$ can be obtained by performing REDUCE$_2$ on symmetric configuration $\mathcal{C}''$ different from $\mathcal{C}$. The following cases may arise.

- $X = (X',0^i,1^{j''})$ and the supermin of $\mathcal{C}''$ starts from the sequence of i consecutive occupied nodes in X. In this case we have that either $\mathcal{C}'' = (0^{i-1},1,0,1^{j-1},0^x,1^{j'-1},0,1,0^{y-1},1,X',0^i,1^{j''})$ or $x = 1$ and $\mathcal{C}'' = (0^i,1^{j+1},0,1^{j'-2},0,1,0^{y-1},1,X',0^i,1^{j''})$.
  The supermin of $\mathcal{C}''$ is either $\mathcal{C}''^{\min} = (0^i,1^{j''},0^{i-1},1,0,1^{j-1},0^x,1^{j'-1},0,1,0^{y-1},1,X')$ or $\mathcal{C}''^{\min} = (0^i,1^{j''},0^i,1^{j+1},0,1^{j'-2},0,1,0^{y-1},1,X')$, respectively. In any case, we must have that $j'' > j$ otherwise we obtain a contradiction to the superminimality of $\mathcal{C}^{\min}$. It follows that the axis of symmetry of $\mathcal{C}$ does not pass through the middle of the initial sequence of i consecutive occupied nodes. Therefore, by symmetry X' contains a sequence $(0^i,1^j,0)$ which is a contradiction to the superminimality of $\mathcal{C}''^{\min}$.
- The supermin of $\mathcal{C}''$ starts from the same node as $\mathcal{C}'$. In this case $\mathcal{C}''^{\min} = (0^i,1^j,0^{x-1},1,0,1^{j'-2},0,1,0^{y-1},1,X)$ but, since $\mathcal{C}$ is symmetric, there must exist in $\mathcal{C}$ a sequence which starts by $(0^i,1^j,0^x)$ different from that at nodes v_0—$v_{i+j+x-1}$. As $\mathcal{C} \notin S_3$, such a sequence must belong to X (or $\overline{X}$) and therefore it is still in $\mathcal{C}''$ and induces a view which is smaller than $\mathcal{C}''^{\min}$, a contradiction.
- The supermin of $\mathcal{C}''$ starts from the sequence of consecutive occupied nodes corresponding to position $i+j$ of $\mathcal{C}$. Three cases may arise.
  - $x = i$, $y = 2$ and $j'-1 < j$. In this case $\mathcal{C}'' = (0^i,1^j,0^i,1^{j'-1},0,1,1,0,X)$ and $\mathcal{C}''^{\min} = (0^i,1^{j'-1},0,1,1,0,X,0^i,1^j)$. As $j' \geq j$ by the superminimality of $\mathcal{C}^{\min}$, then $j' = j$ and hence $\mathcal{C}''^{\min} = (0^i,1^{j-1},0,1,1,0,X,0^i,1^j)$. It follows that either X contains a sequence $(0^i,1^{j-1},0)$ or that X starts by 0^{i-1} and $j = 3$. In the first case we obtain a contradiction with the superminimality of $\mathcal{C}^{\min}$, in the second case the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_2$, a contradiction.
  - $x = i-1$, $j' = 1$, and $j > 1$. In this case, $\mathcal{C}' = (0^i,1^j,0^i,1,0^{y-1},1,X)$ with $\mathcal{C}'^{\min} = (0^i,1,0^{y-1},1,X,0^i,1^j)$. We assume that $X = (1^{j''-1},0,X')$, for some $j'' \geq 1$, which implies that $\mathcal{C}''^{\min} = (0^i,1,0^{y-1},1^{j''-1},0,1,X',0^i,1^j)$. If $j'' > 1$, in order to be symmetric, $\mathcal{C}''^{\min}$ must contain another sequence starting by $(0^i,1,0)$ but this implies a contradiction to the superminimality of $\mathcal{C}^{\min}$. If $j'' = 1$, then $\mathcal{C}''^{\min} = (0^i,1,0^y,1,X',0^i,1^j)$. Note that in this case $y < i$ as otherwise the superminimality of $\mathcal{C}^{\min}$ is contradicted. Therefore also in this case, $\mathcal{C}''^{\min}$ must contain another sequence starting by $(0^i,1,0)$ and the same arguments as for $j'' > 0$ hold.
  - $x = i$, $j' = 1$, and $j > 1$. In this case, $\mathcal{C}' = (0^i,1^j,0^{i+1},1,0^{y-1},1,X)$ and $\mathcal{C}'^{\min} = (0^{i+1},1,0^{y-1},1,X,0^i,1^j)$. Again, we assume that $X = (1^{j''-1},0,X')$ and this implies that $\mathcal{C}''^{\min} = (0^{i+1},1,0^{y-1},1^{j''-1},0,1,X',0^i,1^j)$ which cannot be symmetric as there is only one sequence of $i+1$ consecutive occupied nodes and $j > 1$. □

Lemma 14 (Lemma 7) *Let $\mathcal{C}$ be a configuration in S_2, or such that $\mathcal{C}^{\min} = (0^i,1,0^i,1,(0^i,1,0^i,1,0^{i-1},1)^\ell)$, for some $\ell \geq 1$, or such that $\mathcal{C}^{\min} = (0^i,1,0^x,1,(0^i,1,0^{x-1},1)^\ell)$, for some $\ell \geq 2$ and $1 \leq x \leq i$, or such that $\mathcal{C}^{\min} = (0^i,1,0^x,1,0^i,1,0^y,1)$, for some $1 < y < x \leq i$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_{-1}$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent*

with respect to REDUCE$_1$, REDUCE$_2$, *or* REDUCE$_{-1}$ *to any symmetric configuration different from* $\mathcal{C}$.

Proof If $\mathcal{C}^{\min} = (0^i,1,0^i,1,(0^i,1,0^i,1,0^{i-1},1)^\ell)$ for some $\ell \geq 1$, then $\mathcal{C}'^{\min} = (0^{i+1},1,0^i,1,(0^i,1,0^i,1,0^{i-1},1)^{\ell-1},0^i,1,0^i,1,0^{i-2},1)$. Therefore, $\mathcal{C}'^{\min}$ is asymmetric and can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if $\mathcal{C}'' = (0^i,1,0^{i+1},1,(0^i,1,0^i,1,0^{i-1},1)^{\ell-1},0^i,1,0^i,1,0^{i-2},1)$, or $\mathcal{C}'' = (0^{i+1},1,0^{i-1},1,0^{i+1},1,0^i,1,0^{i-1},1,(0^i,1,0^i,1,0^{i-1},1)^{\ell-2},0^i,1,0^i,1,0^{i-2},1)$ (if $\ell \geq 2$), or $\mathcal{C}'' = (0^{i+1},1,0^{i-1},1,0^{i+1},1,0^i,1,0^{i-2},1)$ (if $\ell = 1$). In any case $\mathcal{C}''$ is asymmetric.

If $\mathcal{C}^{\min} = (0^i,1,0^x,1,(0^i,1,0^{x-1},1)^\ell)$ for some $\ell \geq 2$ and $1 \leq x \leq i$, then we distinguish the following cases:

- $x = 1$. In this case $\mathcal{C}^{\min} = (0^i,1,0,1,(0^i,1,1)^\ell)$ and $\mathcal{C}'^{\min} = (0^i,1,0,1,(0^i,1,1)^{\ell-1},0^{i-1},1,0,1)$ which is always asymmetric for $\ell \geq 2$, and can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if $\mathcal{C}'' = (0^{i-1},1,0,0,1,(0^i,1,1)^{\ell-1},0^{i-1},1,0,1)$, or $\mathcal{C}'' = (0^i,1,1,0,(0^i,1,1)^{\ell-1},0^{i-1},1,0,1)$, or $\mathcal{C}'' = (0^i,1,0,1,0^{i-1},1,0,1,(0^i,1,1)^{\ell-2},0^{i-1},1,0,1)$. In the first two cases $\mathcal{C}''$ is asymmetric. In the third case, if $\ell > 2$, then $\mathcal{C}''$ is asymmetric, otherwise the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to the any of REDUCE$_1$, REDUCE$_2$, and REDUCE$_{-1}$.
- $x \geq 2$. In this case $\mathcal{C}'^{\min} = (0^{i+1},1,0^x,1,(0^i,1,0^{x-1},1)^{\ell-1},0^i,1,0^{x-2},1)$ which is asymmetric and can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if $\mathcal{C}''^{\min} = (0^i,1,0^{x+1},1,(0^i,1,0^{x-1},1)^{\ell-1},0^i,1,0^{x-2},1)$ or $\mathcal{C}''^{\min} = (0^{i+1},1,0^{x-1},1,0^{i+1},1,0^{x-1},1,(0^i,1,0^{x-1},1)^{\ell-2},0^i,1,0^{x-2},1)$. In any case $\mathcal{C}''$ is asymmetric.

If $\mathcal{C}^{\min} = (0^i,1,0^x,1,0^i,1,0^y,1)$, for some $1 < y < x \leq i$, then $\mathcal{C}'^{\min} = (0^{i+1},1,0^x,1,0^i,1,0^{y-1},1)$ which is asymmetric and can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if $\mathcal{C}''^{\min} = (0^i,1,0^{x+1},1,0^i,1,0^{y-1},1)$ or $\mathcal{C}''^{\min} = (0^{i+1},1,0^{x-1},1,0^{i+1},1,0^{y-1},1)$. In any case the move performed by ALIGN-TWO-SYM from $\mathcal{C}''$ is REDUCE$_{-1}$ while the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_{-1}$.

If $\mathcal{C}^{\min} \in S_2$, then $\mathcal{C}^{\min} = (0^i,1^j,0^i,Z)$ with $i > 1$, $Z = \overline{Z}$ and $j \geq 1$. Let us assume without loss of generality that $Z = (1^{j'},0,Z',0,1^{j'})$ with $Z' = \overline{Z'}$ and $j' \geq j$. Then, $\mathcal{C}^{\min} = (0^i,1^j,0^i,1^{j'},0,Z',0,1^{j'})$. We distinguish the following cases

- $j' = 1$. In this case $j = 1$ and hence $\mathcal{C}^{\min} = (0^i,1,0^i,1,0,Z',0,1)$ and $\mathcal{C}'^{\min} = (0^{i+1},1,0^i,1,0,Z',1)$. Since in such configuration there is only one sequence of $i+1$ consecutive occupied nodes, then $\mathcal{C}'$ can be symmetric only if $(0^i,1,0,Z') = (Z',0,1,0^i)$. As $Z' = \overline{Z'}$ we can apply Lemma 9 (with $Y = (0^i,1,0)$ and $X = Z'$) and the only possibility is that $Z' = (0,1,0)^\ell$ and $i = 1$, a contradiction as $i > 1$.
  If $\mathcal{C}'$ can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$, then $\mathcal{C}'' = (0^i,1,0^{i+1},1,0,Z',1)$, or $\mathcal{C}'' = (0^{i+1},1,0^{i-1},1,0,0,Z',1)$, or $\mathcal{C}'' = (0^{i+1},1,0^i,1,1,Z'',1,1)$ where $Z' = (1,Z'',1)$. In the first case, the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to any of REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$. In the second case $\mathcal{C}''$ is symmetric only if $Z' = ((0,0,1,0,0)^\ell)$ and $i = 3$, in which case $\mathcal{C}^{\min} = (0,0,0,1,0,0,0,1,0,(0,0,1,0,0)^\ell,0,1)$ which is periodic. In the third case $\mathcal{C}''$ is asymmetric.
- $j' > 1$. In this case $\mathcal{C}' = (0^i,1^j,0^i,1^{j'},0,Z',1,0,1^{j'-1})$.
  If $j' = j$, then $\mathcal{C}'^{\min} = (0^i,1^{j-1},0,1,Z',0,1^j,0^i,1^j)$. It follows that $\mathcal{C}'$ is asymmetric and that it can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if: $\mathcal{C}'' = (0^{i-1},1,0,1^{j-2},0,1,Z',0,1^j,0^i,1^j)$ or $\mathcal{C}'' = (0^{i-1},1^{j-1},0,1,Z',0,1^j,0^i,1^{j-1},0,1)$. In the first case $\mathcal{C}''$ is asymmetric, in the second case the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to any of REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$.
  If $j' > j$, then $\mathcal{C}'^{\min} = (0^i,1^j,0^i,1^{j'-1},0,1,Z',0,1^{j'})$ and $\mathcal{C}'$ is asymmetric as another supermin would imply a contradiction to the superminimality of $\mathcal{C}$. $\mathcal{C}'$ can be obtained by applying REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$ on a configuration $\mathcal{C}''$ different from $\mathcal{C}$ only if: $\mathcal{C}'' = (0^{i-1},1,0,1^{j-1},0^i,1^{j'-1},0,1,Z',0,1^{j'})$, or

$\mathcal{C}'' = (0^{i-1}, 1^j, 0^i, 1^{j'-1}, 0, 1, Z', 0, 1^{j'-1}, 0, 1)$, or $\mathcal{C}''^{\min} = (0^i, 1^j, 0^{i-1}, 1, 0, 1^{j'-2}, 0, 1, Z', 0, 1^{j'})$. In any case the step from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to any of REDUCE$_1$, or REDUCE$_2$, or REDUCE$_{-1}$.

Here we show that the move from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to REDUCE$_2$ in the case when $\mathcal{C}'' = (0^{i-1}, 1^j, 0^i, 1^{j'-1}, 0, 1, Z', 0, 1^{j'-1}, 0, 1)$ and $\mathcal{C}''^{\min} = (0^i, 1^j, 0^{i-1}, 1, 0, 1^{j'-1}, 0, Z', 1, 0, 1^{j'-1})$. According to the algorithm, in order to apply REDUCE$_2$ to $\mathcal{C}''$ it must hold that $\mathcal{C}''$ is symmetric and one of the following hold:

- Applying REDUCE$_1$ to $\mathcal{C}''$ we obtain a symmetric configuration. The configuration obtained by applying REDUCE$_1$ to $\mathcal{C}''$ is $\mathcal{C}''' = (0^i, 1^{j-1}, 0, 1, 0^{i-2}, 1, 0, 1^{j'-1}, 0, Z', 1, 0, 1^{j'-1})$ and it is asymmetric since $(0^i, 1^{j-1}) < (0^i, 1^j)$ and such sequence must be contained in Z', a contradiction to the superminimality of $\mathcal{C}^{\min}$.
- At least two among $\mathcal{C}^\alpha$, $\mathcal{C}^\beta$, and $\mathcal{C}^\gamma$ of lines 13–15 of Procedure ALIGN-TWO-SYM are symmetric, the procedure from both of them to the configuration $\mathcal{C}''' = (0^i, 1^{j-1}, 0, 1, 0^{i-2}, 1, 0, 1^{j'-1}, 0, Z', 1, 0, 1^{j'-1})$ obtained by applying REDUCE$_1$ to $\mathcal{C}''$ corresponds to REDUCE$_1$, and $\mathcal{C}'' \in S_1 \setminus S_3$ (see line 16 of Procedure ALIGN-TWO-SYM). In this case $\mathcal{C}^\alpha = (0^{i-1}, 1, 0, 1^{j-2}, 0, 1, 0^{i-2}, 1, 0, 1^{j'-1}, 0, Z', 1, 0, 1^{j'-1})$ and $\mathcal{C}^\beta = (0^{i-1}, 1^{j-1}, 0, 1, 0^{i-2}, 1, 0, 1^{j'-1}, 0, Z', 1, 0, 1^{j'-2}, 0, 1)$. If $\mathcal{C}'' \in S_1 \setminus S_3$, then both $\mathcal{C}^\alpha$ and $\mathcal{C}^\beta$ contain a sequence of i consecutive occupied nodes and therefore the supermin of such configurations does not start with $(0^{i-1}, 1)$. It follows that the move from $\mathcal{C}^\alpha$ or $\mathcal{C}^\beta$ to $\mathcal{C}'''$ does not correspond to REDUCE$_1$. □

Lemma 15 *Let $\mathcal{C}$ be a configuration with supermin $\mathcal{C}^{\min} = (0^i, 1, 0, 1, 0^x, 1, 0, 1)$, $i > 1$ and $0 < x < i$, and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_2$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_1$ *and* REDUCE$_2$ *to any symmetric configuration different from $\mathcal{C}$.*

Proof The configuration obtained by applying REDUCE$_2$ on only one robot on $\mathcal{C}$ is $\mathcal{C}'^{\min} = (0^i, 1, 0, 0, 1, 0^{x-1}, 1, 0, 1)$ which is asymmetric as there exists only one sequence of i consecutive occupied nodes and the axis of symmetry cannot pass in the middle of it. Let us assume that $\mathcal{C}'$ can be obtained by applying REDUCE$_1$ or REDUCE$_2$ on a configuration different from $\mathcal{C}$. Then, three cases may arise.

- $\mathcal{C}'' = (0^{i-1}, 1, 0, 0, 0, 1, 0^{x-1}, 1, 0, 1)$. In this case either $\mathcal{C}''^{\min} = (0^{i-1}, 1, 0, 0, 0, 1, 0^{x-1}, 1, 0, 1)$ or $\mathcal{C}''^{\min} = (0, 0, 0, 1, 0^{i-1}, 1, 0, 1, 0^{x-1}, 1)$. In the former case $\mathcal{C}''$ is asymmetric if $i-1 \geq 4$, or $i-1 = 3$ and $x-1 > 1$, or $i-1 = 3$ and $x-1 = 0$, while the step from C'' to C' does not correspond to REDUCE$_1$ or to REDUCE$_2$ if $i-1 = 3$ and $x-1 = 1$. The latter case can occur only if $i-1 < 3$, or $i-1 = 3$ and $x-1 \in \{0, 1\}$. If $i-1 < 3$, then we can have that $i-1 = 1$ and $x = 1$, or $i-1 = 2$ and $x = 2$, or $i-1 = 2$ and $x = 1$. In any of this cases either $\mathcal{C}''$ is asymmetric or the step from C'' to C' does not correspond to REDUCE$_1$ or to REDUCE$_2$.
- $\mathcal{C}'' = (0^{i-1}, 1, 0, 0, 1, 0^{x-1}, 1, 0, 0, 1)$. In this case, if $i-1 > 2$, then $\mathcal{C}''^{\min} = (0^{i-1}, 1, 0, 0, 1, 0^{x-1}, 1, 0, 0, 1)$ but the step from C'' to C' does not correspond to REDUCE$_1$ or to REDUCE$_2$. If $i-1 = 2$ then $\mathcal{C}''^{\min} = (0, 0, 1, 0, 0, 1, 0, 0, 1, 0^{x-1}, 1)$ and step from C'' to C' does not correspond to REDUCE$_1$, moreover REDUCE$_2$ cannot be performed on $\mathcal{C}''$ as it is in S_3. If $i-1 = 1$, then $x = 1$, $\mathcal{C}'' = (0, 1, 0, 0, 1, 1, 0, 0, 1)$ and $\mathcal{C}''^{\min} = (0, 0, 1, 0, 1, 0, 0, 1, 1)$ and hence the step from C'' to C' does not correspond to REDUCE$_1$ or to REDUCE$_2$.
- $\mathcal{C}'' = (0^i, 1, 0, 0, 1, 1, 0, 0, 1)$ with $x = 2$. In this case the step from C'' to C' corresponds to REDUCE$_2$ but it is not performed by ALIGN as from $\mathcal{C}''$ only REDUCE$_1$ can be performed. □

Lemma 16 *Let $\mathcal{C}$ be a configuration with supermin $\mathcal{C}^{\min} = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1)$. Then $\mathcal{C}$ is asymmetric and the only symmetric configuration that is adjacent to $\mathcal{C}$ with respect to any procedure permitted by* ALIGN *is $\mathcal{C}' = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1)$.*

Proof It is easy to see that $\mathcal{C}$ is asymmetric. Let us assume that $\mathcal{C}$ can be obtained by applying REDUCE$_1$, REDUCE$_2$, or REDUCE$_{-1}$ on a symmetric configuration $\mathcal{C}''$ different from $\mathcal{C}'$. By analyzing all the possible moves of the robots in the direction opposite to the reduction of the supermin, four cases may arise: (i) $\mathcal{C}'' = (0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1)$, (ii) $\mathcal{C}'' = (0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1)$, (iii) $\mathcal{C}'' = (0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1)$, and (iv)

$\mathcal{C}'' = (0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)$. In case (i), $\mathcal{C}'' = \mathcal{C}'$, in cases (ii) and (iii) $\mathcal{C}''$ is asymmetric, in case (iv) the move from $\mathcal{C}''$ to $\mathcal{C}$ corresponds to the move performed by algorithm ALIGN. □

Lemma 17 *Let $\mathcal{C}$ be a configuration in $S_{4b} \setminus \{(0^i,1,0,1,0^x,1,0,1),(0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)\}$ and let $\mathcal{C}'$ be the configuration obtained by applying* REDUCE$_{-1}$ *on only one robot on $\mathcal{C}$. Then $\mathcal{C}'$ is asymmetric and it is not adjacent with respect to* REDUCE$_1$ *and* REDUCE$_2$ *to any procedure permitted by* ALIGN *to any symmetric configuration different from $\mathcal{C}$.*

Proof We only show the case when $\mathcal{C}^{\min} = (0^{i-1},1,0,1,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^\ell,1,1,0,1,1)$, for some $i \geq 3$ and $\ell \geq 0$, the other cases can be proven by using similar arguments. The configuration obtained by applying REDUCE$_{-1}$ on only one robot on $\mathcal{C}$ is $\mathcal{C}'^{\min} = (0^{i-1},1,0,1,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^\ell,1,1,1,0,1)$ which is asymmetric as there exists only one sequence of 3 consecutive empty nodes and the axis of symmetry cannot pass in the middle of it. Let us assume that $\mathcal{C}'$ can be obtained by applying REDUCE$_1$, REDUCE$_2$, or REDUCE$_{-1}$ on a configuration different from $\mathcal{C}$. Then, four cases may arise: $\mathcal{C}'' = (0^{i-2},1,0,0,1,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^\ell,1,1,1,0,1)$, $\mathcal{C}'' = (0^{i-2},1,0,1,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^\ell,1,1,1,0,0,1)$, $\mathcal{C}'' = (0^{i-1},1,1,0,1,0,1,0^{i-1},(1,1,0,1,1,0^{i-1})^\ell,1,1,1,0,1)$, $\mathcal{C}'' = (0^{i-1},1,0,1,1,1,0^i,(1,1,0,1,1,0^{i-1})^\ell,1,1,1,0,1)$. In any case the obtained configuration is asymmetric, except for the last case when $\ell = 0$. However, in such a case $\mathcal{C}'' = (0^{i-1},1,0,1,1,1,0^i,1,1,1,0,1)$ the move from $\mathcal{C}''$ to $\mathcal{C}'$ does not correspond to any procedure permitted by ALIGN. □

Lemma 18 *Let $\mathcal{C}$ be a symmetric and allowed configuration and let $\mathcal{C}'$ be the configuration obtained after applying Algorithm* ALIGN*, then $\mathcal{C}'^{\min} < \mathcal{C}^{\min}$.*

Let $\mathcal{C}$ and $\mathcal{C}'$ be two configurations such that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying Algorithm ASYM*, and $\mathcal{C}'$ is adjacent to a symmetric configuration with respect to a move permitted by* ALIGN*. If $\mathcal{C}''$ is the symmetric configuration obtained from $\mathcal{C}'$ after applying Algorithm* ALIGN *(i.e. by forcing a pending move on $\mathcal{C}'$), then $\mathcal{C}''^{\min} < \mathcal{C}^{\min}$.*

Proof For the first statement, we analyze each move permitted by ALIGN separately.

- If $\mathcal{C} = (0,1^j,0,R)$ and the move from $\mathcal{C}$ to $\mathcal{C}'$ is REDUCE$_0$, then $\mathcal{C}' = (0,1^{j-1},0,1,R) < \mathcal{C}^{\min}$.
- If $\mathcal{C}^{\min} = (0^i,1^j,0,R)$ and the move from $\mathcal{C}$ to $\mathcal{C}'$ is REDUCE$_1$, then $\mathcal{C}' = (0^i,1^{j-1},0,1,R) < \mathcal{C}^{\min}$.
- If $\mathcal{C}^{\min} = (0^i,1^j,0^{i'},1^{j'},0,R)$ and the move from $\mathcal{C}$ to $\mathcal{C}'$ is REDUCE$_2$, then $\mathcal{C}' = (0^i,1^j,0^{i'},1^{j'-1},0,1,R) < \mathcal{C}^{\min}$.
- If $\mathcal{C}^{\min} = (0^i,R,0,1^j)$ and the move from $\mathcal{C}$ to $\mathcal{C}'$ is REDUCE$_{-1}$, then $\mathcal{C}' = (0^i,R,1,0,1^{j-1})$. Let ℓ be such that $\mathcal{C}^{\min} = \overline{\mathcal{C}^{\min}_\ell}$, then $\overline{\mathcal{C}'_\ell} < \mathcal{C}^{\min}$.
- If $\mathcal{C}^{\min} = (0,0,0,1,0,0,1,0,0,1,0,0,0,1,1)$, then $\mathcal{C}' = (0,0,1,0,0,0,1,0,0,1,0,0,0,1,1)$ (see line 20 of Procedure ALIGN-TWO-SYM), and $\mathcal{C}'_3 < \mathcal{C}^{\min}$.

To prove the second statement, we have to go into the behavior of ASYM. In particular, the only possibilities for $\mathcal{C}$ and $\mathcal{C}'$ are the following (see [14]).

1. $\mathcal{C} = (0,1^j,0,R)$ and $\mathcal{C}' = (0,1^{j-1},0,1,R)$;
2. $\mathcal{C} = (0^i,1^j,0,R)$ and $\mathcal{C}' = (0^i,1^{j-1},0,1,R)$;
3. $\mathcal{C} = (0^i,1^j,0^{i'},1^{j'},0,R)$ and $\mathcal{C}' = (0^i,1^j,0^{i'},1^{j'-1},0,1,R)$;
4. $\mathcal{C} = (0^i,1,R,0,1)$ and $\mathcal{C}' = (0^{i+1},1,R,1)$.

In the first case, the move from $\mathcal{C}$ to $\mathcal{C}'$ corresponds to REDUCE$_0$ and therefore, by Lemma 2, $\mathcal{C}'$ is not adjacent to a symmetric configuration with respect to any move permitted by ALIGN.

In any other case, note that $\mathcal{C}' < \mathcal{C}$. In what follows we show that $\mathcal{C}''$ contains a sequence that is smaller than or equal to the sequence A that precedes R in $\mathcal{C}'$ and this implies that $\mathcal{C}''^{\min} < \mathcal{C}^{\min}$. By contradiction, let us assume that such a sequence is not in $\mathcal{C}''$, it follows that one of the robots in A moved toward R. Note that no forced pending move can involve the initial sequence of consecutive 0. This directly implies a contradiction for case 4.

In case 2, $A = (0^i,1^{j-1},0,1)$ and then $\mathcal{C}'' = (0^i,1^j,0,R) = \mathcal{C}$ which is asymmetric, a contradiction.

In case 3, $A = (0^i,1^j,0^{i'},1^{j'-1},0,1)$ and either $\mathcal{C}'' = (0^i,1^j,0^{i'},1^{j'},0,R)$ or $\mathcal{C}'' = (0^i,1^j,0^{i'-1},1,0,1^{j'-2},0,1,R)$. In the first case $\mathcal{C}'' = \mathcal{C}$ which is asymmetric, a contradiction. In the second case, the only possibility is that the move from $\mathcal{C}'$ to $\mathcal{C}''$ is REDUCE$_2$ and hence there exists a supermin in R, that is $R = (1^{j''},0^{i''},R')$, for some $i'' \geq i$, j'', and R'. Since sequence R is in $\mathcal{C}$, then $i'' = i$ as otherwise we obtain a contradiction to the superminimality of $\mathcal{C}$. Therefore $\mathcal{C}'' = (0^i,1^j,0^{i'-1},1,0,1^{j'-2},0,1^{j''+1},0^i,R')$. Since $(0^i,1^{j''+1},0,1^{j'-2},0,1,0^{i'-1},1^j,0^i)$ must be the prefix of a supermin, then $j''+1 \leq j$. However,

this implies that $\mathcal{C} = (0^i, 1^j, 0^{i'}, 1^{j'}, 0, 1^{j''}, 0^i, R')$, where $j'' \leq j - 1$, which is a contradiction to the superminimality of $\mathcal{C}$ since the sequence $(0^i, 1^{j''}, 0)$ is smaller than $(0^i, 1^j, 0)$. □

7 Conclusions

We have proposed two algorithms to solve two problems that, in the last decade, received main attention in the context of the Look-Compute-Move model of computation: the gathering and the exclusive searching. Our algorithms work under very weak assumptions. The results provided here constitute a characterization of the two problems that leaves open only few marginal cases. For the gathering, this paper closes a long standing open problem. In fact, almost all the cases left open by the literature are now closed. Moreover, we have also addressed the lack of a unified algorithm that works for any gatherable configuration. For the exclusive searching, our algorithm handles the missing cases of our previous work on the same subject, leaving open some periodic configurations and some specific cases.

One of our main contributions consists of Algorithm ALIGN that is in common to the two strategies adopted to solve the gathering and the exclusive searching. The same algorithm might be used as a preliminary step also to solve other problems in the same settings like e.g. exploration with stop or perpetual exploration. In fact, Algorithm ALIGN permits to restrict the attention to a very limited subset of configurations, hence simplifying the design of new algorithms.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM Journal on Computing **36**(1), 58–82 (2006)
2. Bampas, E., Czyzowicz, J., Gąsieniec, L., Ilcinkas, D., Labourel, A.: Almost optimal asynchronous rendezvous in infinite multidimensional grids. In: Proc. of the 24th International Symposium on Distributed Computing (DISC), *Lecture Notes in Computer Science*, vol. 6343, pp. 297–311 (2010)
3. Blin, L., Burman, J., Nisse, N.: Exclusive graph searching. In: Proc. of the 21st Annual European Symposium on Algorithms (ESA), *Lecture Notes in Computer Science*, vol. 8125, pp. 181–192. Springer (2013)
4. Blin, L., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Exclusive perpetual ring exploration without chirality. In: Proc. of the 24th International Symposium on Distributed Computing (DISC), *Lecture Notes in Computer Science*, vol. 6343, pp. 312–327 (2010)
5. Bonnet, F., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Asynchronous exclusive perpetual grid exploration without sense of direction. In: 15th Int. Conf. On Principles Of Distributed Systems (OPODIS), *Lecture Notes in Computer Science*, vol. 7109, pp. 251–265. Springer (2011)
6. Chalopin, J., Das, S.: Rendezvous of mobile agents without agreement on local orientation. In: Proc of the 37th International Colloquium on Automata, Languages and Programming (ICALP), vol. 6199, pp. 515–526 (2010)
7. Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Network exploration by silent and oblivious robots. In: 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG), *Lecture Notes in Computer Science*, vol. 6410, pp. 208–219. Springer (2010)
8. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: Gathering. SIAM Journal on Computing **41**(4), 829–879 (2012)
9. Czyzowicz, J., Gąsieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theoretical Computer Science **410**(6–7), 481 – 499 (2009)
10. D'Angelo, G., Di Stefano, G., Klasing, R., Navarra, A.: Gathering of robots on anonymous grids and trees without multiplicity detection. Theoretical Computer Science **610**, 158–168 (2016)
11. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model. In: S. Alpern, R. Fokkink, L. Gąsieniec, R. Lindelauf, V. Subrahmanian (eds.) Search Theory: A Game Theoretic Perspective, pp. 197–222. Springer (2013)
12. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering on rings under the look-compute-move model. Distributed Computing **27**(4), 255–285 (2014)
13. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering six oblivious robots on anonymous symmetric rings. Journal of Discrete Algorithms **26**, 16–27 (2014)
14. D'Angelo, G., Di Stefano, G., Navarra, A., Nisse, N., Suchan, K.: Computing on rings by oblivious robots: A unified approach for different tasks. Algorithmica **4**(72), 1055–1096 (2015)
15. D'Angelo, G., Navarra, A., Nisse, N.: Gathering and exclusive searching on rings under minimal assumptions. In: Proc. of the 15th International Conference on Distributed Computing and Networking (ICDCN), *Lecture Notes in Computer Science*, vol. 8314, pp. 149–164. Springer (2014)
16. Di Stafano, G., Navarra, A.: Gathering of oblivious robots on infinite grids with minimum traveled distance. Information and Computation To appear
17. Di Stefano, G., Montanari, P., Navarra, A.: About ungatherability of oblivious and asynchronous robots on anonymous rings. In: Proc. of the 26th International Workshop on Combinatorial Algorithms (IWOCA'15), *Lecture Notes in Computer Science*, vol. 9538, pp. 136–147. Springer (2016)
18. Dieudonne, Y., Pelc, A., Peleg, D.: Gathering despite mischief. In: Proc. of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 527–540 (2012)
19. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Remembering without memory: Tree exploration by asynchronous oblivious robots. Theoretical Computer Science **411**(14-15), 1583–1598 (2010)
20. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: How many oblivious robots can explore a line. Information Processing Letters **111**(20), 1027–1031 (2011)

21. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing without communicating: Ring exploration by asynchronous oblivious robots. Algorithmica **65**(3), 562–583 (2013)
22. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by oblivious mobile robots. Morgan and Claypool (2012)
23. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: 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), *Lecture Notes in Computer Science*, vol. 1741, pp. 93–102. Springer (1999)
24. Fomin, F.V., Thilikos, D.M.: An annotated bibliography on guaranteed graph searching. Theoretical Computer Science **399**(3), 236–245 (2008)
25. Ilcinkas, D., Nisse, N., Soguet, D.: The cost of monotonicity in distributed graph searching. Distributed Computing **22**(2), 117–127 (2009)
26. Izumi, T., Izumi, T., Kamei, S., Ooshita, F.: Mobile robots gathering algorithm with local weak multiplicity in rings. In: Proc. of the 17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 6058, pp. 101–113 (2010)
27. Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., Yamashita, M.: The gathering problem for two oblivious robots with unreliable compasses. SIAM Journal on Computing **41**(1), 26–46 (2012)
28. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S.: Asynchronous mobile robot gathering from symmetric configurations. In: Proc. of the 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 6796, pp. 150–161 (2011)
29. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S.: Gathering an even number of robots in an odd ring without global multiplicity detection. In: Proc. of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS), *Lecture Notes in Computer Science*, vol. 7464, pp. 542–553 (2012)
30. Klasing, R., Kosowski, A., Navarra, A.: Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. Theoretical Computer Science **411**, 3235–3246 (2010)
31. Klasing, R., Markou, E., Pelc, A.: Gathering asynchronous oblivious mobile robots in a ring. Theoretical Computer Science **390**, 27–39 (2008)
32. Kranakis, E., Krizanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. Morgan & Claypool (2010)
33. Prencipe, G.: Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In: Proc. of the 7th Italian Conference on Theoretical Computer Science (ICTCS), pp. 154–171 (2001)