HAL
archives-ouvertes.fr

Gathering and Exclusive Searching on Rings under Minimal Assumptions

Gianlorenzo d'Angelo, Alfredo Navarra, Nicolas Nisse

▶ **To cite this version:**

Gianlorenzo d'Angelo, Alfredo Navarra, Nicolas Nisse. Gathering and Exclusive Searching on Rings under Minimal Assumptions. 15th International Conference on Distributed Computing and Networking (ICDCN), Jan 2014, Coimbatore, India. pp.149-164, 10.1007/978-3-642-45249-9_10 . hal-00931514

HAL Id: hal-00931514
https://hal.inria.fr/hal-00931514

Submitted on 15 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Gathering and Exclusive Searching on Rings under Minimal Assumptions

Gianlorenzo D'Angelo[1], Alfredo Navarra[1], and Nicolas Nisse[2]

[1] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy.
gianlorenzo.dangelo@dmi.unipg.it, alfredo.navarra@unipg.it
[2] Inria and Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, France
nicolas.nisse@inria.fr

Abstract. Consider a set of mobile robots with minimal capabilities placed over distinct nodes of a discrete anonymous ring. Asynchronously, each robot takes a snapshot of the ring, determining 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. The computation also 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 minimal assumptions.

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 ring of n nodes, and we investigate two fundamental problems requiring complex coordination: *Gathering* (see, e.g., [5, 10, 13, 26]) and *Exclusive Searching* (see, e.g., [2, 19, 20]).

We assume minimal abilities for the robots. They are oblivious (without memory of the past), uniform (running the same deterministic algorithm), autonomous (without a common coordinate system, identities or chirality), asynchronous (without central coordination), 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 [17]). The Look-Compute-Move model considers that in each cycle a robot takes a snapshot

of the current global configuration (Look), then, based on the perceived configuration, takes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case it moves to this 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) 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*, has first been defined in continuous environment [18, 27]. 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 have been considered in discrete environments. Exploration with stop has been studied in paths [16], trees [15], rings [14] and general graphs [6]. More recently, the gathering problem (a.k.a. Rendez-vous) has been considered in rings [9, 11, 25] and grids [1, 7]. Exclusive perpetual exploration has been studied in rings [3] and grids [4]. The *exclusivity property* states that any node must be occupied by at most one robot. Very recently, exclusive perpetual searching has been defined and studied in trees [2] and rings [11]. 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' disposal and not on 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 towards the same node and remain there. On rings, under the Look-Compute-Move model, the gathering is unsolvable if the robots are not empowered by the so-called *multiplicity detection* capability [25], either in its *global* or *local* version. In the former type, a robot is able to perceive whether any node of the graph is occupied by a single robot or more than one (i.e., a *multiplicity* occurs) without perceiving the exact number. In the latter (and weaker) type, a robot is able to perceive the multiplicity only if it is part of it. Using the global multiplicity detection capability, in [25] some impossibility results have been proven. Then, several algorithms have been proposed for different kinds of exclusive initial configurations in [8, 24, 25]. These papers left open some cases which have been closed in [9] where a

unified strategy has been provided. With local multiplicity detection capability, an algorithm starting from rigid configurations where the number of robots k is strictly smaller than $\lfloor \frac{n}{2} \rfloor$ has been designed in [21]. In [22], the case where k is odd and strictly smaller than $n-3$ has been solved. In [23], 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 has been solved in [11]. The remaining cases are left open and the design of a unified algorithm for all the cases is still unknown.
Exclusive Searching: Graph searching has been widely studied in centralized and distributed settings (e.g., [19, 20]). 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 recontaminated 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 cleared. 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 asynchronicity. Following [2, 11], 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. 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 perpetual exclusive searching in the discrete model has been introduced in [2] for tree topologies. Concerning rings, in [11] the case of initial rigid configurations has been tackled.
Contribution: We consider the gathering with local multiplicity detection and the perpetual exclusive searching problems for k robots in an n-nodes ring.

For any $k < n-4$, $k \neq 4$, we fully characterize the exclusive configurations from which the gathering problem is feasible. 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 multiplicity detection, but for the unsolvable configurations that will be specified later. Similarly to the case of $k = 4$ in [9] and $(n, k) = (7, 6)$ in [8], 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 solvable. That is, we design an algorithm that solves the problem starting from any such aperiodic exclusive configurations but for the unsolvable ones. For periodic configurations, we provide some impossibility results. 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, resp.) exploit a common technique (provided in Section 3) 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. Due to space constraints, most of the proofs and the pseudo-codes of the algorithms are reported in Appendix.

2 Notation and preliminary

In this paper, we consider a 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, $1 \leq j \leq n$. Intuitively, $\mathcal{S}_i$ represents the positions of robots, starting at v_i. For any $X = (x_0, \cdots, x_r)$, let us denote $\overline{X} = (x_r, \cdots, x_0)$ and $X_i = (x_{i \bmod r}, \ldots, x_{r+i \bmod r})$. 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 $S \in \mathcal{S}_\mathcal{C}$. Note that, for any exclusive configuration $S = (s_0, \cdots, s_{n-1}) \in \mathcal{S}_\mathcal{C}$, $\sum_{i<n} s_i = n - k$. 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 $s_{n-1} = 1$.

We denote by x^h a sequence of $h \geq 0$ consecutive x, $x \in \{0,1\}$. We say that a sequence X is *palindrome* if $X = \overline{X}$, it is *symmetric* if X_i is 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). It is known that an aperiodic configuration admits at most one axis of symmetry [9]. Moreover, an aperiodic configuration has either a unique supermin representation or two symmetrical supermins [9].

Allowed configurations: Let us summarize the known feasible and unfeasible exclusive configurations for both gathering and graph searching. In [25], it is shown that gathering is not solvable for $k = 2$, for any periodic initial configuration, and for any initial configuration with an axis of symmetry passing through two edges. In [11], 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 k even starting from any configuration with an axis of symmetry passing through an empty node.

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 unsolvable, and if the number of robots does not fall in the above defined impossibility ranges. In particular, all rigid configurations with a number of robots out of the defined ranges are allowed. For gathering, the symmetric allowed configurations are all aperiodic ones with the axis of symmetry not passing through two edges and $3 \leq k < n-4$, $k \neq 4$. For exclusive searching, the symmetric allowed configurations are all aperiodic ones with k odd and those with k even where the axis does not pass through an empty node, provided that $3 < k < n - 2$ and $n > 9$.

Dealing with symmetry: The core of the technique in [11] 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.

Lemma 1 ([11]). *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, that can be uniquely distinguished, moves to an unoccupied neighbor, 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, r' should perform the same (symmetric) move. However, due to asynchronicity, 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 reached configurations that might have a pending move are asymmetric, distinguishable and the pending move is unique. Therefore, in such a case, our algorithm forces the pending move. That is, contrary to [11] 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 distinguish such configurations, we define the notion of adjacent configurations. An asymmetric configuration $\mathcal{C}$ is *adjacent* to a symmetric configuration $\mathcal{C}'$ with respect to a procedure M allowed by the algorithm if $\mathcal{C}$ can be obtained from $\mathcal{C}'$ by applying M to only one of the robots permitted to move by M. 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.
Overview of Algorithm Align: Our contribution mainly relies on Algorithm Align, described in Section 3. 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. From such configurations, we will show how to solve the gathering and the exclusive searching problems. Here, 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. Algorithm Align proceeds as follows:

- If no two robots occupy two adjacent nodes in $\mathcal{C}$, we prove that only two cases are possible. If $\mathcal{C}$ is symmetric, then Algorithm Align-One is executed by two symmetric robots. In this case, if only one of them actually moves, then the obtained configuration is asymmetric and adjacent only to $\mathcal{C}$. Then,

the possible pending move is forced. Otherwise, if $\mathcal{C}$ is asymmetric and not adjacent to a symmetric configuration, Algorithm Asym can be executed without ambiguity.

– If two robots occupy two adjacent nodes in $\mathcal{C}$ (i.e.,the supermin representation of $\mathcal{C}$ starts by 0^2) and $\mathcal{C}$ is symmetric, then moving only one robot can lead to a configuration which is symmetric or adjacent to a different symmetric configuration. 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 a way that it is possible to univocally determine the possible pending move in the case that only one of two symmetric robots actually moves (Algorithm Align-Two-Sym). If $\mathcal{C}$ is asymmetric, there are two cases: either $\mathcal{C}$ is not adjacent to any symmetric configuration and Algorithm Asym is executed or we force to perform the unique possible pending move (Algorithm Align-Two-Asym).

In detail, if the initial allowed configuration is symmetric and k is even, Align 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 achieved, if they are allowed. If k is odd, then the configuration achieved 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 in an arbitrary direction the unique robot lying on the axis of symmetry.

– 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 the algorithm in [11] 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 ALIGN always reduce the supermin and that only allowed configuration are reached.

3 ALIGN algorithm

In this section, we devise 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. 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.[3] Let ℓ_1 be the smallest integer such that $\ell_1 > 0$, $v_{\ell_1} = 0$ and $v_{\ell_1-1} = 1$; let ℓ_2 be the smallest integer such that $\ell_2 > \ell_1$, $v_{\ell_2} = 0$ and $v_{\ell_2-1} = 1$; let ℓ_{-1} be the largest integer such that $\ell_{-1} < n$ and $v_{\ell_{-1}} = 0$. The four procedures permitted by ALIGN are the following:

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

Algorithm ALIGN works in two phases: the first phase (Algorithm ALIGN-ONE) copes with configurations without any consecutive occupied nodes (i.e. $v_1 = 1$) while the second phase copes with configurations having at least two consecutive occupied nodes (Algorithm ALIGN-TWO-SYM, if the configuration is symmetric, and ALIGN-TWO-ASYM otherwise).

Algorithm ALIGN-ONE. If $v_1 = 1$ and the configuration $\mathcal{C}$ is symmetric, the general strategy is to reduce the supermin by performing REDUCE$_0$. If the two symmetric robots that should move perform their Look-Compute-Move cycles synchronously, then the obtained configuration $\mathcal{C}'$ is symmetric where the supermin is reduced and the axis of symmetry of $\mathcal{C}$ is preserved. 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 with respect to any possible procedure that allows at most two robots to move.

Lemma 2 ([9]). *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 $\mathcal{C}$.*

[3] By v_i we denote both the i-th node and the i-th value of sequence $\mathcal{C}^{\min}$.

It follows that robots can recognize whether $\mathcal{C}'$ has been obtained by performing REDUCE$_0$ from $\mathcal{C}$. In the affirmative case, ALIGN forces to perform the possible pending move.

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$_{-1}$. In [12] (Lemma 5 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 symmetric configuration and the configuration obtained is not adjacent with respect to any possible procedures performed by ALIGN.[4] Therefore, 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.

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 the algorithm in [11] (Algorithm ASYM). 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.

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 consists in moving r (in any arbitrary direction). The obtained configuration is asymmetric and not adjacent to any other symmetric configuration with respect to the procedures of ALIGN. Then, we can safely perform ASYM as there are no pending moves.

It follows that ALIGN-ONE leads to a configuration with two consecutive occupied nodes. In detail, we can obtain: (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 procedure among REDUCE$_1$, REDUCE$_2$ or REDUCE$_{-1}$ is performed.

[4] Configuration $\mathcal{C} = (0, 1, 0, 1, 1, 0, 1, 1)$ is the only exception, see [12].

In symmetric configurations, we perform REDUCE_1 every time it is possible. This occurs when the asynchronous execution of the two symmetric robots that should perform the procedure cannot generate a symmetric configuration with a different axis of symmetry or a configuration which is adjacent to a different symmetric configuration with respect to any procedure permitted by ALIGN.

If it is not possible to perform REDUCE_1, we perform REDUCE_2. It can be proven that asynchronous executions cannot generate other symmetries or configurations adjacent to symmetric ones potentially reachable.

There are cases when we cannot perform REDUCE_1 and REDUCE_2. For instance this can happen 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 move the robot at v_{i+j} which is symmetric to that at v_{i-1}. Similar instances where it is not possible to perform REDUCE_2 can occur. In such cases, we perform REDUCE_{-1} and show that this cannot create any different symmetry or configuration adjacent to symmetric ones with respect to any procedure permitted by ALIGN.

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

The sets S_2 and S_3 contain the configurations where it is not possible to perform REDUCE_1 or REDUCE_2, respectively. In Lemmata 6-10 of [12], we identify the procedures that can be safely performed on the configurations in such sets. 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, we can safely perform REDUCE_{-1}. 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. If $\mathcal{C}'$ is symmetric, then we know that $\mathcal{C} \in S_1 \setminus S_3$, $\mathcal{C}^{\min} = (0^i, 1, 0, 0, 1, 0^i, (1, 0, 1, 0^i)^\ell, 1, 0, 1)$, or $\mathcal{C}^{\min} = (0^i, 1, 0^i, 1, 0^i, 1, 0^i, (1, 0^{i-1}, 1, 0^i, 1, 0^i)^\ell, 1, 0^{i-1}, 1)$, for some $\ell > 0$. In the former case, we can safely perform REDUCE_2 as the obtained configuration is neither symmetric nor adjacent to any other symmetric configuration. In the latter two cases, we cannot perform REDUCE_2 but we can safely perform REDUCE_{-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 $j \geq 1$, 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 $R = (1, R')$, $\mathcal{C}^\gamma := (0^i, 1^{j+1}, 0, R', 1)$. 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 at least one of them must belong to $S_1 \setminus S_3$. Therefore, we can safely perform REDUCE$_2$ on such configuration and REDUCE$_1$ on the other one.

In any other symmetric configuration, ALIGN-TWO-SYM applies REDUCE$_1$.[5]

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 of such procedures only if $\mathcal{C}'$ is one of the following configurations: $\mathcal{C}^\alpha := (0^{i-1}, 1, 0, 1^{j-1}, 0^x, 1^{j'}, R, 1)$, $\mathcal{C}^\beta := (0^{i-1}, 1^j, 0^x, 1^{j'}, R, 0, 1)$, if $j' > 0$, $\mathcal{C}^\gamma := (0^i, 1^j, 0^{x-1}, 1, 0, 1^{j'-1}, R, 1)$, or, if $R = (0, 1, R')$, $\mathcal{C}^\delta := (0^i, 1^j, 0^x, 1^{j'+1}, 0, R', 1)$. Note that, at most one of the above configurations can be symmetric. Let $\mathcal{C}^i$ be such a configuration, if by applying ALIGN-TWO-SYM (or ALIGN-ONE if $\mathcal{C}^i$ has no consecutive occupied nodes) on a single robot of $\mathcal{C}^i$ we obtain $\mathcal{C}$, then $\mathcal{C}$ has been possibly obtained from $\mathcal{C}^i$ and then ALIGN-TWO-ASYM performs the possible pending move. If none of $\mathcal{C}^i$, $i \in \{\alpha, \beta, \gamma, \delta\}$, is symmetric, then $\mathcal{C}$ has not been obtained from any symmetric configurations and then ALIGN-TWO-ASYM performs ASYM. As in the case of ALIGN-ONE, if the robot leading from $\mathcal{C}^i$ to $\mathcal{C}$ is that on the axis of symmetry of $\mathcal{C}^i$, then Algorithm ASYM is performed. The next theorem shows the correctness of ALIGN.

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

[5] With the exception of configurations $\mathcal{C}^{s_1} = (0^{i_1}, 1, 0, 1, 0^x, 1, 0, 1)$ and $\mathcal{C}^{s_2} = (0^{i_2}, 1, 1, 0^y, 1, 1)$ with $x < i_1$ and $y < i_2$, see [12].

the remaining asymmetric configurations. We denote such sets as S, $AS1$ and $AS2$, respectively. Lemmata 1, 2 and 5-10 in [12] imply the following properties.

- A node in S representing a configuration $\mathcal{C}$ has either one or two outgoing arcs. 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. 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 [12] (Lemmata 5-10), there exists only one of such nodes. 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. 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 $AS2$.

It follows that any execution path performed by the algorithm is made of nodes representing allowed configurations. Moreover, each allowed configuration has an outgoing arc that is traversed by the execution path of the algorithm. Moreover, any procedure performed by the algorithm reduces the supermin of a configuration. This implies that the graph is acyclic, as we can define a topological ordering of the nodes on the basis of the 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. □

4 Gathering in a ring

In this section, we provide the full strategy for achieving the gathering. We make use of procedure ALIGN to reach one of the following configurations: $\mathcal{C}^a = (0^{k-1}, 1, 0, 1^{n-k-1})$, $\mathcal{C}^b = (0^k, 1^{n-k})$, 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.

Algorithm ALIGN terminates when either the obtained configuration is one of the three above, or it is one of the configurations generated by Algorithm GATHERING below.

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 $\mathcal{C}^d = (0^{k-1}, 1, 1, 0, 1^{n-k-2})$. 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 performs REDUCE$_0$, hence creating a multiplicity, and still obtaining configuration $\mathcal{C}^d$. 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 towards the other occupied node 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 iterating procedure COMPACT$_0$ as defined below. Let $\mathcal{C} = (v_0, v_1, \ldots, v_n)$ be a configuration of type $(0^{\frac{k}{2}-i}, 1, 0^i, 1^j, 0^i, 1, 0^{\frac{k}{2}-i}, 1^{n-k-j-2})$ where $1 \leq i \leq \frac{k}{2}$ and $j < \frac{n-k-2}{2}$. Note that for $i = \frac{k}{2}$, $\mathcal{C} = \mathcal{C}^c$. Procedure COMPACT$_0$ moves the robot at $v_{\frac{k}{2}-i-1}$ towards $v_{\frac{k}{2}-i}$. As $\mathcal{C}$ is symmetric, COMPACT$_0$ permits two robots to move. If both move synchronously, the resulting configuration $\mathcal{C}'$ is similar to $\mathcal{C}$ but with i increased by one. 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, and hence $\mathcal{C}'$ can be easily obtained. Once $\mathcal{C}^c$ with $j = 1$ is reached, again COMPACT$_0$ is applied. If both the permitted robots move, a symmetric configuration $\mathcal{C}'' = (0^{\frac{k}{2}-1}, 1, 0, 1, 0^{\frac{k}{2}-1}, 1^{n-k-1})$, with $v_{\frac{k}{2}}$ being a multiplicity, is reached. This equals to the case of symmetric configurations with k odd that will be discussed later. 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, then $4 < k < n-4$ and hence, such a configuration is asymmetric and not adjacent to any other symmetric configuration. Then, $\mathcal{C}''$ can be easily obtained.

If k is even and n is odd, then ALIGN either reaches configuration $\mathcal{C}^b$ or $\mathcal{C}^c$ with either j or $n-k-j$ odd. In this case, GATHERING behaves as above but creating the multiplicity at the central node of the only odd sequence of consecutive empty nodes among j and $n-k-j$. Eventually, GATHERING achieves configuration $\mathcal{C}''$. Again, this equals to the case of symmetric configurations with k odd. Note that, this case is similar to the technique presented in [23] where the solved configurations are only those with k even and n odd.

If k is odd, then ALIGN always reaches configuration $\mathcal{C}^b$. In this case, the used technique is similar to that presented in [22] 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 towards $v_{\frac{k-1}{2}}$. 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', configuration $\mathcal{C}''$ is achieved subsequently. If both robots move synchronously, again configuration $\mathcal{C}''$ is reached. From here, GATHERING performs procedure COMPACT$_1$ defined as follows. Let $\mathcal{C} = (v_0, v_1, \ldots, v_n)$ be a configuration of type $(0^{\frac{k-i}{2}}, 1, 0^i, 1, 0^{\frac{k-i}{2}}, 1^{n-k-2})$ where $1 \leq i \leq k$, then COMPACT$_1$ moves the robot at $v_{\frac{k-i}{2}-1}$ towards $v_{\frac{k-i}{2}}$. As $\mathcal{C}$ is symmetric, COMPACT$_1$ permits two robots to move. If both move synchronously, the resulting configuration $\mathcal{C}'$ is similar to $\mathcal{C}$ but with i increased by two. If only one robot moves, as before, the obtained configuration is asymmetric and not adjacent to any other symmetric configuration, and $\mathcal{C}'$ can be easily obtained. By iterating this process, GATHERING achieves configuration $\mathcal{C}^b$ with the number of occupied nodes decreased by two. Eventually, this process terminates with only one occupied node.

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.*

5 Exclusive searching in a ring

In this section, we present an algorithm that allows a team of robots to exclusively search a ring.

If k is even and there exists an axis of symmetry passing through an empty node, the searching is clearly unsolvable 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. Moreover, we prove that graph searching is impossible starting from any periodic configuration with one or two empty nodes per period.

In [11], an algorithm is designed allowing $5 \leq k \leq n-3$ robots to search exclusively a ring with $n \geq 10$ nodes (but for $(k,n) = (5,10)$), for rigid initial configurations. Here, we improve over this algorithm by addressing also aperiodic symmetric configurations. We use two sub-procedures: Algorithm COMPACT-ALIGN is used after ALIGN to achieve configuration $\mathcal{C}^b$, when ALIGN reaches configuration $\mathcal{C}^c$, and Algorithm BREAK-SYMMETRY is used to achieve $\mathcal{C}^a$ in the case that k is odd.

Algorithm SEARCH-RING. 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 [11] can be applied and the ring is searched. If k is odd, we first use Algorithm BREAK-SYMMETRY to break the potential symmetry and then use the algorithm of [11]. Each of these configurations used during the searching phase of the algorithm of [11] 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. Algorithm COMPACT-ALIGN is

first applied until one of the configurations in $\mathcal{A}$ (described in [12]) 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 which actually performs the searching.

The intuitive explication of the Searching algorithm (Phase 2) is as follows. All robots are aligned on consecutive nodes. Then, both robots r and r' at the ends of this segment move (one clockwise and the other anti-clockwise) to reach the two adjacent nodes opposite to the occupied segment. Then, the two robots q and q' occupying the ends of the "long" occupied segment move to their empty neighbors. These moves indicate to r and r' that it is time to go back toward the "long" segment, and that is what happens. Finally, when r is adjacent to q and r' is adjacent to q', then q and q' move to their empty neighbors in order to re-build the original segment. Then, the process is repeated perpetually. Such a sequence of performed moves actually searches the ring. Moreover, by definition of the configurations met during the process (configurations in $\mathcal{A}$), there is no ambiguity in the choice of the robot(s) that must move. Finally, there are no conflicts between the different phases of our procedure because any configuration in $\mathcal{A}$ is not adjacent to any symmetric configuration not in $\mathcal{A}$. We then get:

Theorem 3. *Let $4 < k \leq n-3$ robots, forming an allowed configuration in an n-node ring, Algorithm* Search-Ring *perpetually searches the ring, but for $n = 10$ and $k = 5$, and for symmetric configurations with $n = 10$ and $k = 6$.*

References

1. E. Bampas, J. Czyzowicz, L. Gąsieniec, D. Ilcinkas, and A. Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *Proc. of 24th DISC*, volume 6343 of *LNCS*, pages 297–311, 2010.
2. L. Blin, J. Burman, and N. Nisse. Brief announcement: Distributed exclusive and perpetual tree searching. In *Proc of 26th DISC*, volume 7611 of *LNCS*, pages 403–404, 2012.
3. L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *Proc. of 24th DISC*, volume 6343 of *LNCS*, pages 312–327, 2010.
4. F. Bonnet, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Asynchronous exclusive perpetual grid exploration without sense of direction. In *15th Int. Conf. On Principles Of Distributed Systems (OPODIS)*, volume 7109 of *LNCS*, pages 251–265, 2011.
5. J. Chalopin and S. Das. Rendezvous of mobile agents without agreement on local orientation. In *Proc of 37th ICALP*, volume 6199, pages 515–526, 2010.
6. J. Chalopin, P. Flocchini, B. Mans, and N. Santoro. Network exploration by silent and oblivious robots. In *36th International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 6410 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 2010.
7. G. D'Angelo, G. Di Stefano, R. Klasing, and A. Navarra. Gathering of robots on anonymous grids without multiplicity detection. In *Proc. of 19th SIROCCO*, volume 7355 of *LNCS*, pages 327–338, 2012.
8. G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering of six robots on anonymous symmetric rings. In *18th SIROCCO*, volume 6796 of *LNCS*, pages 174–185, 2011.

9. G. D'Angelo, G. Di Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. In *Proc. of 26th DISC*, volume 7611 of *LNCS*, pages 330–344, 2012.
10. G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model. In S. Alpern, R. Fokkink, L. Gąsieniec, R. Lindelauf, and V. Subrahmanian, editors, *Search Theory: A Game Theoretic Perspective*, pages 197–222. Springer, 2013.
11. G. D'Angelo, G. Di Stefano, A. Navarra, N. Nisse, and K. Suchan. A unified approach for different tasks on rings in robot-based computing systems. In *Proc. of 15th IEEE IPDPS APDCM*, 2013. to appear.
12. G. D'Angelo, A. Navarra, and N. Nisse. Robot Searching and Gathering on Rings under Minimal Assumptions, 2013. Tech. Rep. RR-8250, Inria.
13. Y. Dieudonne, A. Pelc, and D. Peleg. Gathering despite mischief. In *Proc. of 23rd SODA*, pages 527–540, 2012.
14. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *11th International Conference on Principles of Distributed Systems (OPODIS)*, volume 4878 of *Lecture Notes in Computer Science*, pages 105–118. Springer, 2007.
15. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci.*, 411(14-15):1583–1598, 2010.
16. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. How many oblivious robots can explore a line. *Inf. Process. Lett.*, 111(20):1027–1031, 2011.
17. P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by oblivious mobile robots*. Morgan and Claypool, 2012.
18. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *10th Int. Symp. on Algorithms and Computation (ISAAC)*, volume 1741 of *LNCS*, pages 93–102. Springer, 1999.
19. F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.
20. D. Ilcinkas, N. Nisse, and D. Soguet. The cost of monotonicity in distributed graph searching. *Distributed Computing*, 22(2):117–127, 2009.
21. T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *Proc. of 17th SIROCCO*, volume 6058 of *LNCS*, pages 101–113, 2010.
22. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations. In *Proc. of 18th SIROCCO*, volume 6796 of *LNCS*, pages 150–161, 2011.
23. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *Proc. of 37th MFCS*, volume 7464 of *LNCS*, pages 542–553, 2012.
24. R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci.*, 411:3235–3246, 2010.
25. R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390:27–39, 2008.
26. E. Kranakis, D. Krizanc, and E. Markou. *The Mobile Agent Rendezvous Problem in the Ring*. Morgan & Claypool, 2010.
27. G. Prencipe. Instantaneous actions vs. full asynchronicity : Controlling and coordinating a set of autonomous mobile robots. In *ICTCS*, pages 154–171, 2001.