How to gather asynchronous oblivious robots on anonymous rings

Gianlorenzo d'Angelo, Gabriele Di Stefano, Alfredo Navarra

How to gather asynchronous oblivious robots on anonymous rings

Gianlorenzo D'Angelo[1], Gabriele Di Stefano[2], and Alfredo Navarra[3]

[1] MASCOTTE Project, INRIA/I3S(CNRS/UNSA), France.
gianlorenzo.d_angelo@inria.fr
[2] Dipartimento di Ingegneria e Sceinze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Italy. gabriele.distefano@univaq.it
[3] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy.
alfredo.navarra@unipg.it

Abstract. A set of robots arbitrarily placed on different nodes of an anonymous ring have to meet at one common node and remain in there. This problem is known in the literature as the *gathering*. Anonymous and oblivious robots operate in Look-Compute-Move cycles; in one cycle, a robot takes a snapshot of the current configuration (Look), decides whether to stay idle or to move to one of its neighbors (Compute), and in the latter case makes the computed move instantaneously (Move). Cycles are asynchronous among robots. Moreover, each robot is empowered by the so called *multiplicity detection* capability, that is, it is able to detect during its Look operation whether a node is empty, or occupied by one robot, or occupied by an undefined number of robots greater than one.
The described problem has been extensively studied during the last years. However, the known solutions work only for specific initial configurations and leave some open cases. In this paper, we provide an algorithm which solves the general problem, and is able to detect all the ungatherable configurations. It is worth noting that our new algorithm makes use of a unified and general strategy for any initial configuration, even those left open by previous works.

1 Introduction

We study one of the most fundamental problems of self-organization of mobile entities, known in the literature as the *gathering* problem (see e.g., [8, 10, 14] and references therein). In particular, we consider oblivious robots initially located at different nodes of an anonymous ring that have to gather at a common node and remain in there. Neither nodes nor links are labeled. Initially, each node of the ring is either occupied by one robot or empty. Robots operate in Look-Compute-Move cycles. In each cycle, a robot takes a snapshot of the current global configuration (Look), then, based on the perceived configuration, takes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case it moves to this neighbor (Move), eventually. Cycles are performed asynchronously for each robot. This means that the time between Look,

Compute, and Move operations is finite but unbounded, and it is decided by the adversary for each robot. Hence, robots may move based on significantly outdated perceptions. Moves are instantaneous, and hence during a Look operation robots are seen at nodes and not on edges. Robots are identical, execute the same deterministic algorithm and are empowered by the so-called *multiplicity detection* capability [15]. That is, a robot is able to perceive whether a node of the network is empty, occupied by a single robot or by more than one (i.e., a *multiplicity* occurs), but not the exact number. Without multiplicity detection the gathering has been shown to be impossible on rings [20].

Related Work. The problem of let meet mobile entities on graphs [2, 11, 20] or open spaces [5, 10, 22] has been extensively studied in the last decades. When only two robots are involved, the problem is referred to as the *rendezvous* problem [1, 4, 6, 11, 23]. Under the Look-Compute-Move model, many problems have been addressed, like the *graph exploration* and the *perpetual graph exploration* [3, 12, 13], while the rendezvous problem has been proved to be unsolvable on rings [20].

Concerning the gathering, different types of robot disposals on rings (configurations) have required different approaches. In particular, periodicity and symmetry arguments have been exploited. A configuration is called *periodic* if it is invariable under non-trivial (i.e., non-complete) rotation. A configuration is called *symmetric* if the ring has a geometrical *axis of symmetry*, that reflects single robots into single robots, multiplicities into multiplicities, and empty nodes into empty nodes. A symmetric configuration with an axis of symmetry has an *edge-edge symmetry* if the axis goes through two edges; it has a *node-edge symmetry* if the axis goes through one node and one edge; it has a *node-node symmetry* if the axis goes through two nodes; it has a *robot-on-axis symmetry* if there is at least one node on the axis of symmetry occupied by a robot. In [20], it is proved that the gathering is not solvable for periodic configurations, for those with edge-edge symmetry, and if the multiplicity detection capability is removed. Then all configurations with an odd number of robots, and all the asymmetric configurations with an even number of robots have been solved by different algorithms. In [19], the attention has been devoted to the symmetric cases with an even number of robots, and the problem was solved when the number of robots is greater than 18. These left open the gatherable symmetric cases of an even number of robots between 4 and 18. Most of the cases with 4 robots have been solved in [21]. The remaining ones, referred to as the set $SP4$, are symmetric configurations of type node-edge with 4 robots and the odd interval cut by the axis bigger than the even one. They are ungatherable, in general, as outlined in [19] for configurations of 4 robots on a five nodes ring. Actually, specific configurations in $SP4$ could be gatherable but requiring suitable strategies difficult to be generalized.

Finally, the case of 6 robots with an initial axis of symmetry of type node-edge, or node-node has been solved in [8]. Besides the cases left open, a unified algorithm that handles all the above cases is also missing.

Other interesting gathering results on rings concern the case of the so called *local weak* multiplicity detection. That is, a robot is able to perceive the mul-

tiplicity only if it is part of it. On this respect, our assumption in the rest of the paper concerns the *global weak* multiplicity detection. Whereas, the *strong* version would provide the exact number of robots on a node.

Using the local weak assumption, not all the cases has been addressed so far. In [16], it has been proposed an algorithm for aperiodic and asymmetric configurations with the number of robots k strictly smaller than $\lfloor \frac{n}{2} \rfloor$, with n being the number of nodes composing the ring. In [17], the case where k is odd and strictly smaller than $n-3$ has been solved. In [18], an algorithm for the case where n is odd, k is even, and $10 \leq k \leq n-5$ is provided. The remaining cases are still open and a unified algorithm like the one we are proposing here for the global weak assumption is not known.

Without any multiplicity detection, in [7] the grid topology has been exhaustively studied.

Our Results. In this paper, we present a new distributed algorithm for solving all the gatherable configurations (but those potentially in $SP4$) by using the (global weak) multiplicity detection. Our technique introduces a new approach and for some special cases makes use of previous ones. In particular, existing algorithms are used as subroutines for solving the basic gatherable cases with 4 or 6 robots from [21] and [8], respectively. Also, we exploit:

Property 1. [20] Let C be a symmetric configuration with an odd number of robots, without multiplicities. Let C' be the configuration resulting from C by moving the unique robot on the axis to any of its adjacent nodes. Then C' is either asymmetric or still symmetric but aperiodic. Moreover, by repeating this procedure a finite number of times, eventually the configuration becomes asymmetric (with possibly one multiplicity).

For all the other gatherable configurations, we design a new approach that has been suitably unified with the used subroutines. Our result answers to the posed conjectures concerning the gathering, hence closing all the cases left open, and providing a general approach that can be applied to all the initial configurations. The main result of this paper can be stated as follows.

Theorem 1. *There exists a distributed algorithm for gathering $k > 2$ robots on a ring, provided that the composed configuration does not belong to the set $SP4$, it is aperiodic, it does not admit an edge-edge axis of symmetry. The algorithm also allows robots to recognize whether a configuration is ungatherable.*

2 Definitions and Preliminaries

We consider an n-nodes anonymous ring without orientation. Initially, k nodes of the ring are occupied by k robots. During a Look operation, a robot perceives the relative locations on the ring of multiplicities and single robots. The current configuration of the system can be described in terms of the view of a robot r. We denote a configuration seen by r as a tuple $Q(r) = (q_0, q_1, \ldots, q_j)$, $j \leq k-1$, that represents the sequence of the numbers of free consecutive nodes broken up

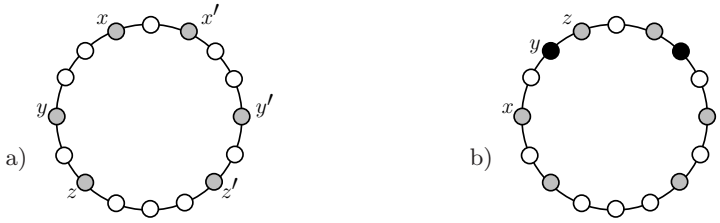

Fig. 1. a) The intervals between robots y, z and y', z' are the supermins, while the supermin configuration view is $(1, 2, 1, 2, 1, 3)$. b) Black nodes represent multiplicities.

by robots when traversing the ring in one direction, starting from r. Abusing the notation, for any $0 \leq i \leq j$, we refer by q_i not only to the length of the i-th interval but also to the interval itself. Unless differently specified, we refer to $Q(r)$ as the lexicographical minimum view among the two possibilities. For instance, in the configuration of Fig. 1a, we have that $Q(x) = (1, 2, 1, 3, 1, 2)$. A multiplicity is represented as $q_i = -1$ for some $0 \leq i \leq j$, regardless the number of robots in the multiplicity. For instance, in the configuration of Fig. 1b, $Q(x) = (1, -1, 0, 1, 0, -1, 1, 1, 3, 1)$. Given a generic configuration $C = (q_0, q_1, \ldots, q_j)$, let $\overline{C} = (q_0, q_j, q_{j-1}, \ldots, q_1)$, and let C_i be the configuration obtained by reading C starting from q_i, that is $C_i = (q_i, q_{(i+1) \bmod j+1}, \ldots, q_{(i+j) \bmod j+1})$. The above definitions imply:

Property 2. Given a configuration C,

i) there exists $0 < i \leq j$ such that $C = C_i$ iff C is periodic;
ii) there exists $0 \leq i \leq j$ such that $C = \overline{(C_i)}$ iff C is symmetric;
iii) C is aperiodic and symmetric iff there exists only one axis of symmetry.

The next definition represents the key feature for our algorithm since it has a twofold advantage. In fact, based on it, a robot can distinguish if the perceived configuration (during the Look phase) is gatherable and if it is one of the robots allowed to move (during the Compute phase).

Definition 1. *Given a configuration $C = (q_0, q_1, \ldots, q_j)$ such that $q_i \geq 0$, for each $0 \leq i \leq j$, the view defined as $C^{SM} = \min\{C_i, \overline{(C_i)}, \mid 0 \leq i \leq j\}$ is called the* supermin configuration view. *An interval is called* supermin *if it belongs to the set $I_C = \{q_i \mid C_i = C^{SM}$ or $\overline{(C_i)} = C^{SM}, 0 \leq i \leq j\}$.*

The next lemma, based on Definition 1, is exploited to detect possible symmetry or periodicity features of a configuration:

Lemma 1. *Given a configuration $C = (q_0, q_1, \ldots, q_j)$ with $q_i \geq 0$, $0 \leq i \leq j$:*

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

Proof. 1.⇒) If $|I_C| = 1$, then if C is symmetric, there exists at least an axis of symmetry. This axis must pass through the supermin, as otherwise there exists another interval of the same size of supermin to which the supermin is reflected with respect to the axis. However, the same should hold for every neighboring interval of the supermin and so forth. Since by hypothesis, supermin is unique, there must exist at least two intervals of different sizes that are reflected by the supposed symmetry, and hence C results asymmetric.

If C is asymmetric then it must be aperiodic, as otherwise there exists $0 < i \leq j$ such that $C = C_i$ and this implies more than one copy of the supermin.

1.⇐) If C is asymmetric and aperiodic, then $C_i \neq \overline{(C_i)}$, $C_i \neq C_\ell$ and $C_i \neq \overline{(C_\ell)}$, for each i and $\ell \neq i$ and hence must exist a unique supermin. If C admits only one axis of symmetry traversing the supermin, then there exists a unique $0 \leq i \leq j$ such that $C^{SM} = C_i = \overline{(C_i)}$ as otherwise Property 2 would imply the existence of other axes of symmetry, one for each supermin.

2.⇒) If $|I_C| = 2$ and C is asymmetric, then by Property 2, it is periodic and the period must be of $\frac{n}{2}$. If $|I_C| = 2$ and C is aperiodic and symmetric, the axis of symmetry cannot pass through both the supermins. In fact, if it does, $C^{SM} = \overline{(C^{SM})} = (C^{SM})_{j/2} = \overline{(C^{SM})_{j/2}}$ that implies $(C^{SM})_{\lfloor j/4 \rfloor} = \overline{(C^{SM})}_{\lceil j/4 \rceil}$, i.e., there exists another axis of symmetry orthogonal to the first one that reflects the supermin into the other supermin. Hence, C would be periodic.

2.⇐) If C is aperiodic and symmetric with the unique axis not passing through any supermin, then each supermin must be reflected by the axis to another one. Moreover, there cannot be more than 2 supermins, as by definition of supermin, these imply other axes of symmetry, i.e., by Property 2, C is periodic. If C is periodic with period $\frac{n}{2}$, then any supermin has an exact copy after $\frac{n}{2}$ intervals, and there cannot be other supermins, as otherwise the period would be smaller.

3.⇒) If $|I_C| > 2$, then there are at least 3 supermins, and hence C has a period of at most $\frac{n}{3}$.

3.⇐) If C has a period of at most $\frac{n}{3}$, then a supermin is repeated at least 3 times in C. □

2.1 A first look to the algorithm

The above lemma already provides useful information for a robot when it wakes up. In fact, during the Look operation, it can easily recognize if the configuration contains only 2 robots, or if it belongs to the set $SP4$, or if $|I_C| > 2$ (i.e., the configuration is periodic), or in case $|I_C| = 2$, if the configuration admits an edge-edge axis of symmetry or it is again periodic. After this check, a robot knows if the configuration is gatherable, and proceeds with its computations. Indeed, we will show in the next section that all the other configurations are gatherable. From now on, we do not consider the above ungatherable configurations.

The main strategy allows only movements that affect the supermin. In fact, if there is only one supermin, and the configuration allows its reduction, the subsequent configuration would still have only one supermin (the same as before but reduced), or a multiplicity is created. In general, such a strategy would

lead asymmetric configurations or also symmetric ones with the axis passing through the supermin to create one multiplicity where the gathering will be easily finalized by collecting at turn the closest robots to the multiplicity.

For gatherable configurations with $|I_C| = 2$, our algorithm requires more phases before creating the final multiplicity where the gathering ends. In this case there are two supermins that can be reduced. If both are reduced simultaneously, then the configuration is still symmetric and gatherable. Possibly, it contains two symmetric multiplicities. In fact, this is the status that we want to reach even when only one of the two supermins is reduced. In general, the algorithm tries to preserve the original symmetry or to create a gatherable symmetric configuration from an asymmetric one. It is worth to remark that in all symmetric configurations with an even number of robots, the algorithm always allows the movement of two symmetric robots. Then it may happen that, after one move, the obtained configuration is either symmetric or it is asymmetric with a possible *pending* move. In fact, if only one robot among the two allowed to move performs its movement, it is possible that its symmetric one either has not yet started its Look phase, or it is taking more time. If there might be a pending move, then the algorithm forces it before any other decision.

In contrast, asymmetric configurations cannot produce pending moves as the algorithm allows the movement of only one robot. In fact, we reduce the unique supermin by deterministically distinguish among the two adjacent robots, until one multiplicity is created. Finally, all the other robots will join the multiplicity one-by-one. In some special cases, from asymmetric configurations at one "allowed" move from symmetry (i.e., with a possible pending move), robots must guess which move would have been realized from the symmetric configuration, and force it in order to avoid unexpected behaviors. By doing this correctly, the algorithm brings the configuration to have two symmetric multiplicities as above, eventually. From here, a new phase that collects all the other robots but two into the multiplicities starts. Still the configuration may move from symmetric configurations to asymmetric ones at one move from symmetry. Once the desired symmetric configuration with two multiplicities and two single robots is reached, a new phase starts and moves the two multiplicities to join each other. The node where the multiplicities join represents the final gathering location.

3 Gathering algorithm

The algorithm works in 5 phases that depend on the configuration perceived by the robots, see Fig. 2. First, it starts from a configuration without multiplicities and performs phase MULTIPLICITY-CREATION whose aim is to create one multiplicity, where all the robots will eventually gather, or a symmetric configuration with two multiplicities. In the former case, phase CONVERGENCE is performed to gather all the robots into the multiplicity. In the latter case, phases COLLECT and then MULTIPLICITY-CONVERGENCE are performed in order to first collect all the robots but two into the two multiplicities and then to join the two multiplicities into a single one. After that, phase CONVERGENCE is performed. Special

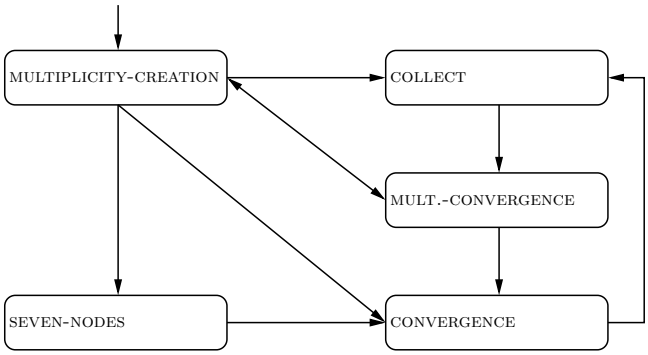

Fig. 2. Phases interchanges.

cases of 7 nodes and 6 robots are considered separately in phase SEVEN-NODES. Due to space constraints, we do provide the details only for the first phase MULTIPLICITY-CREATION. The formal descriptions of the other phases can be found in the full version of the paper [9].

We can show how robots interchange from one phase to another until the final gathering is achieved. In each phase we can distinguish the type of configuration and provide the algorithm to be performed by robots for each of these types. The way how a robot can identify the type of configuration will be outlined later.

3.1 Phase multiplicity-creation

The main idea is to reduce the supermin by enlarging the largest interval adjacent to it as follows:

Definition 2. *Let* $Q(r) = (q_0, q_1, \ldots, q_j)$ *be a supermin configuration view, then robot* r *performs the* REDUCTION *if the obtained configuration after its move is* $(q_0 - 1, q_1, \ldots, q_j + 1)$.

The pseudo-code of REDUCTION is shown above. The procedure, first checks whether the robot perceives the supermin configuration view by comparing the configuration C perceived by the robot with C^{SM}. Note that, in asymmetric configurations, the robot that perceived C^{SM} is the one among the two robots at the sides of the supermin allowed to move. In fact, the robot on the other side would perceive the configuration $\overline{C}$ and, by definition of C^{SM}, we have $C^{SM} = C < \overline{C}$, as the configuration is asymmetric. Then, the procedure moves the robot towards the supermin. In symmetric configurations, the test at line 1 returns true for both robots adjacent to the unique supermin or for the two symmetric robots that perceive C^{SM} in case that $|I_C| = 2$.

It is worth to note that, from a symmetric configuration, always two robots can perform the REDUCTION when it is possible to perform it. If only one of them does it, the obtained configuration will contain exactly one supermin. However, if the perceived configuration contains only one supermin and it is not symmetric,

Procedure: REDUCTION
Input : $C = (q_0, q_1, \ldots, q_j)$

1 **if** $C = C^{SM}$ **then** move towards q_0;

the robots are able to understand whether there might be a pending move to re-establish the original symmetry or not. This constitutes one of the main results of the paper and it is obtained from the following lemma.

Lemma 2. *Let C be a configuration with more than 2 single robots and let C' be the one obtained from C after a* REDUCTION *performed by a single robot. If C is asymmetric then C' is at least at two moves from a symmetric configuration; if C is symmetric then C' is at least at two moves from any other symmetric configuration with an axis of symmetry different from that of C.*

Proof. By Lemma 1, two cases may arise: there exists only one supermin in C or the configuration is symmetric and contains exactly two supermins.

We now show that in the case that there exists only one supermin in C, then C' is at least two moves from any symmetric configuration with the axis different from that passing through the supermin. By Lemma 1, it is enough to show that C' requires more than one move to create another supermin different from that of C. Let us consider the supermin configuration view $C^{SM} = (q_0, q_1, \ldots, q_j)$. For the sake of simplicity, let us assume that, for each $i = 1, 2, \ldots, j$, $(C^{SM})_i < \overline{(C^{SM})_i}$, the case where, for some i, $(C^{SM})_i > \overline{(C^{SM})_i}$ is similar. The case that $(C^{SM})_i = \overline{(C^{SM})_i}$ cannot occur as, otherwise, there exists an axis of symmetry passing through q_i, but, by Lemma 1, as $|I_C| = 1$, the possible axis of symmetry can only pass through q_0. By definition of supermin, for each $(C^{SM})_i$, $i = 1, 2, \ldots, j$, there exists $k_i \in \{0, 1, \ldots, j\}$ such that: $q_\ell = q_{(i+\ell) \bmod j+1}$, for each $\ell < k_i$; and $q_{k_i} < q_{(i+k_i) \bmod j+1}$. Note that $(i + k_i) \bmod j + 1 \neq 0$ as otherwise it contradicts the hypothesis of minimality of q_0. Moreover, $k_i \neq j$ as otherwise $\sum_{\ell=0}^{j} q_\ell = \sum_{\ell=0}^{k_i} q_\ell < \sum_{\ell=0}^{k_i} q_{(i+\ell) \bmod j+1} = \sum_{\ell=0}^{j} q_{(i+\ell) \bmod j+1}$, that is a contradiction. From C', the supermin configuration view is $C'^{SM} = (q'_0, q'_1, \ldots, q'_j) = (q_0 - 1, q_1, \ldots, q_j + 1)$ and we have that, for each $i = 1, 2, \ldots, j$, two cases may arise: if $k_i > 0$, then $q'_0 = q_0 - 1 < q_i = q'_i$ and $q'_{k_i} = q_{k_i} < q_{(i+k_i) \bmod j+1} = q'_{(i+k_i) \bmod j+1}$; if $k_i = 0$, then $q'_0 = q_0 - 1 < q_i - 1 = q'_i - 1$. In any case, C'^{SM} differs from $(C'^{SM})_i$ by two units. It follows that C' is at least two moves from any symmetric configuration with the axis different from that passing through the supermin. In fact, in order to obtain another axis of symmetry by performing only one move on C', $(C'^{SM})_i$ has to differ from C'^{SM} by at most one unit. This is enough to show the statement for the case of symmetric configurations with exactly one supermin. Regarding the asymmetric case, it remains to show that C' is at least two moves from any symmetric configuration with the axis passing through the supermin. In an asymmetric configuration $C^{SM} = (q_0, q_1, \ldots, q_j)$ there exists a q_k, $1 \leq k \leq \frac{j}{2}$, such that $q_\ell = q_{(j+1-\ell) \bmod j+1}$, for each $\ell < k$, and $q_k < q_{j+1-k}$. From C', the

Function: SYMMETRIC
Input : $C = (q_0, q_1, \ldots, q_j)$
Output : true if C is symmetric, false otherwise

1 **for** $i = 0, 1, \ldots, j$ **do**
2 **if** $C = \overline{C_i}$ **then return** true;
3 **return** false;

supermin configuration view is $C'^{SM} = (q'_0, q'_1, \ldots, q'_j) = (q_0 - 1, q_1, \ldots, q_j + 1)$ and two cases may arise: if $k > 1$, then $q'_1 = q_1 = q_j < q_j + 1 = q'_j$ and $q'_k = q_k < q_{j-1-k} = q'_{j-1-k}$; if $k = 1$, then $q'_1 = q_1 < q_j = q'_j - 1$. It follows that C' is at least two moves from any symmetric configuration with the axis passing through the supermin.

Regarding the case of symmetric configurations with exactly 2 supermins, we use similar arguments as above. Let us consider the supermin configuration view $C^{SM} = (q_0, q_1, \ldots, q_j)$ and let us assume that h is the index such that $C^{SM} = \overline{(C^{SM})_h}$. By definition, for each $(C^{SM})_i$, $i \in \{1, 2, \ldots, j\} \setminus \{h\}$, there exists $k_i \in \{0, 1, \ldots, j\}$ such that: $q_\ell = q_{(i+\ell) \bmod j+1}$, for each $\ell < k_i$, and $q_{k_i} < q_{(i+k_i) \bmod j+1}$. As above we are assuming that $(C^{SM})_i < \overline{(C^{SM})_i}$ and we can show that $k_i \neq j$, $k_i \neq (j + h) \bmod j + 1$, $(k_i + i) \bmod j + 1 \neq 0$, and $(k_i + i) \bmod j + 1 \neq h$. From C', the supermin configuration view is $C'^{SM} = (q'_0, q'_1, \ldots, q'_j) = (q_0 - 1, q_1, \ldots, q_j + 1)$ we have that, for each $i \in \{1, 2, \ldots, j\} \setminus \{h\}$ two cases may arise: if $k_i > 0$, then $q'_0 = q_0 - 1 < q_i = q'_i$ and $q'_{k_i} = q_{k_i} < q_{(i+k_i) \bmod j+1} = q'_{(i+k_i) \bmod j+1}$; if $k_i = 0$, then $q'_0 = q_0 - 1 < q_i - 1 = q'_i - 1$. In any case, C'^{SM} differs from $(C'^{SM})_i$ by two units. Similar arguments to the ones used for the asymmetric case can show that C' is at least two moves from any symmetric configuration with the axis passing through the supermin. □

It follows that we can derive C from C' by enlarging the supermin of C'. This equals to reduce the largest adjacent interval (i.e., by performing the REDUCTION backwards) hence deducing the possible original axis of symmetry and then performing the possible pending REDUCTION. Before providing the designed procedures, we need the following definition:

Definition 3. *Given a configuration $C = (q_0, q_1, \ldots, q_j)$, the view defined as $C^{SSM} = \min\{C_i,\ \overline{(C_i)} \mid C_i \neq C^{SM}$ and $\overline{(C_i)} \neq C^{SM},\ 0 \leq i \leq j\}$ is called the* second supermin configuration view.

As shown above, Procedure SYMMETRIC checks whether a configuration C is symmetric by exploiting Property 2. Procedure CHECK_REDUCTION checks whether an asymmetric configuration C has been obtained from some symmetric configuration $\hat{C}$ by performing REDUCTION. Procedure PENDING_REDUCTION performs the pending REDUCTION.

At line 1, Procedure CHECK_REDUCTION looks for the index k such that q_k is the supermin, as it is the only candidate for being the interval that has

Function: CHECK_REDUCTION
Input : $C = (q_0, q_1, \ldots, q_j)$
Output : (true, $\hat{C}$) if C is obtained from $\hat{C}$ by performing REDUCTION, (false, $\emptyset$) if C has not been obtained by performing REDUCTION

1 Let k such that $C_k = C^{SM}$ or $\overline{C}_k = C^{SM}$;
2 **if** $q_{(k-1) \bmod j+1} > q_{(k+1) \bmod j+1}$ **then**
$\hat{C} := (q_0, q_1, \ldots, q_{(k-1) \bmod j+1} - 1, q_k + 1, \ldots, q_j)$;
3 **else**
4 **if** $q_{(k-1) \bmod j+1} < q_{(k+1) \bmod j+1}$ **then**
$\hat{C} := (q_0, q_1, \ldots, q_k + 1, q_{(k+1) \bmod j+1} - 1, \ldots, q_j)$;
5 **else return** (false,$\emptyset$);
6 **if** SYMMETRIC($\hat{C}$) **then return** (true, $\hat{C}$);
7 **return** (false,$\emptyset$);

Procedure: PENDING REDUCTION
Input : $C = (q_0, q_1, \ldots, q_j)$

1 $(b, \hat{C})$ = CHECK_REDUCTION (C) ;
2 **if** b *and* ($\min\{\hat{C}, \overline{\hat{C}_j}\} = \hat{C}^{SM}$ *or* $\min\{\hat{C}, \overline{\hat{C}_j}\} = \hat{C}^{SSM}$) *and* $\min\{C, \overline{C_j}\} \neq C^{SM}$ **then** move towards q_0;

been reduced by a possible REDUCTION. Then, at lines 2–4, it computes the configuration $\hat{C}$ before the possible REDUCTION. This is done by enlarging q_k and reducing the largest interval among $q_{(k-1) \bmod j+1}$ and $q_{(k+1) \bmod j+1}$. If $q_{(k-1) \bmod j+1} = q_{(k+1) \bmod j+1}$ or $\hat{C}$ is not symmetric, then C has not been obtained by performing a REDUCTION from a symmetric configuration. Then, the procedure returns (false, $\emptyset$). If $\hat{C}$ is symmetric, then C has been obtained by performing REDUCTION on $\hat{C}$ and hence the procedure returns (true, $\hat{C}$).

Procedure PENDING_REDUCTION uses CHECK_REDUCTION to check whether C has been obtained by performing REDUCTION on a configuration $\hat{C}$ (lines 1 and 2). At line 2 the procedure checks whether the robot is at a side of one of the two supermins of $\hat{C}$ ($\min\{\hat{C}, \overline{\hat{C}_j}\} = \hat{C}^{SM}$) and if it has not yet performed REDUCTION ($\min\{C, \overline{C_j}\} \neq C^{SM}$). In the affirmative case, the robot has to move towards the supermin (line 2). The robot moves towards q_0 also if it is at the side of the second supermin of $\hat{C}$ ($\min\{\hat{C}, \overline{\hat{C}_j}\} = \hat{C}^{SSM}$). This corresponds to a move different from REDUCTION that will be explained later in this section.

In general, it is not always possible to perform REDUCTION. In fact, there are cases where it may lead to ungatherable configurations. These cases will be managed separately. However, we will show that a robot is always able to understand that there might be a pending move also for the other moves allowed by our algorithm from symmetric configurations.

When it is not possible to perform REDUCTION, we either reduce the second supermin or we perform the XN move that is defined in the following:

Definition 4. *Let C be a configuration:*

- *If C is symmetric and there are no multiplicities,* XN *corresponds to moving towards the axis the two symmetric robots closest to the axis of symmetry that are divided by at most one robot and are not adjacent to a supermin;*[4]
- *If C is symmetric and there is only one multiplicity,* XN *corresponds to moving towards the multiplicity the two symmetric robots closest to the multiplicity;*
- *If C is asymmetric and it has been possibly obtained by applying* XN *from a symmetric configuration C' (that is, from C' only one of the two robots on the above cases has moved), then* XN *on C corresponds to moving the second closest robot towards the axis/multiplicity;*
- *If C is asymmetric with a multiplicity and it cannot be obtained by applying* XN *from a symmetric configuration, then* XN *corresponds to moving the robot lexicographically closest to the multiplicity towards it.*

Each time a robot wakes up, it needs to find out which kind of configuration it is perceiving, and, if it is allowed to move, it needs to compute the right move to be performed. We need to distinguish among several types of configurations, requiring different strategies and moves. In this phase, as there are no multiplicities, a robot must distinguish among the following configurations:

W1 Symmetric configurations with an odd number of robots;
W2 Configurations with 4 robots;
W3 Configurations with 6 robots;
W4 Symmetric configurations with an even number of robots greater than 6, only 1 supermin of size 0 or with 2 supermins of size 0 divided by one interval of even size with no other intervals of size 0;
W5 Symmetric configurations with an even number of robots greater than 6, only 1 supermin of size 0 or with 2 supermins of size 0 divided by one interval of even size, and other intervals of size 0;
W6 Asymmetric configurations with an even number of robots greater than 6 and:
  a) only one interval of size 0, and it is in between two intervals of equal size;
  b) only two intervals of size 0, with only one in between two intervals of equal size;
  c) only two intervals of size 0, with one even interval in between;
  d) only three intervals of size 0, with only two of them separated by an even interval;
  e) only three consecutive intervals of size 0;
  f) only four intervals of size 0, with only three of them consecutive;
W7 Remaining gatherable configurations.

From configurations in W1, only the robot on the axis can move in one of the two directions, arbitrarily. After this move either the configuration contains one multiplicity or it belongs to W1 or W7. Configurations in W7 will be described

[4] By Lemma 1, in gatherable configurations, the axis of symmetry cannot pass through two supermins hence there are always two robots allowed to move.

later in this section and the configurations with multiplicities will be described within the other phases. Regarding configurations in W1, from Property 1, we know that the number of times that the obtained configuration can belong again to W1 after this move is bounded.

When the configuration is in W2 or W3, a modified version of algorithms in [21] and [8] are performed, respectively. In particular, both the algorithms are able to manage symmetric configurations and to check whether in an asymmetric configuration there is a possible pending move. If the configuration is not symmetric and there are no pending moves, then REDUCTION is performed. The resulting configuration is still in W2 or W3 or at least one multiplicity is created. From the correctness of algorithms in [21] and [8] and from the fact that performing REDUCTION results in reducing the supermin, it follows that eventually at least one multiplicity is created.

When the configuration is in W7 and it is symmetric, then the algorithm performs REDUCTION on two symmetric robots that leads to another symmetric configuration in W7, or to a configuration with at least one multiplicity, or to an asymmetric configuration with a pending move. In this latter case, by Lemma 2 the algorithm recognizes that the configuration is at one "allowed" move from symmetry and performs the pending move (even though it was not pending, indeed). When the configuration is asymmetric, again REDUCTION is performed. By performing the described movements, at least one multiplicity is created.

Configurations in W4–W6 correspond to the cases where REDUCTION is not allowed to be performed. In fact, if the configuration is symmetric and there is only one supermin of size 0, then REDUCTION may result in swapping the robots at the borders of the supermin, hence obtaining infinitely many times the same configuration. Similarly, if the configuration is symmetric and there are two symmetric supermins of size 0 divided by one interval of even size, then REDUCTION would produce two multiplicities divided by the interval of even size and we won't be able to join such multiplicities afterwards.

From W4, the algorithm performs XN, hence leading to configurations in W4, W6 or to configurations with one multiplicity on the axis.

From W5, the algorithm performs REDUCTION on the configuration obtained without considering the supermin (that is, it reduces the second supermin, according to Definition 3). Note that, as in this case the second supermin has size 0, we obtain at least one multiplicity.

The asymmetric configurations in W6 are either asymmetric starting configurations or are obtained from the symmetric configurations in W4 after performing XN. In this cases, the algorithm checks whether the configuration is obtained after an XN move. This is realized by moving backward the robot closest to the other pole of the axis of symmetry that is assumed to pass through: The supermin in case a); The only interval of size 0 adjacent to two intervals of equal size in case b); The even intervals mentioned in cases c) and d); the only interval of size 0 in between other two intervals of size 0 in cases e) and f). If a backwards XN produces a symmetric configuration, then the symmetric XN is performed, otherwise, REDUCTION is performed and this move creates a multiplicity.

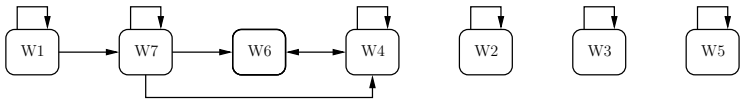

Fig. 3. Phase MULTIPLICITY-CREATION.

For each configuration type, the algorithm checks whether the robot perceiving the configuration C is allowed to move and eventually, performs the move.

This phase of the algorithm is summarized in Fig. 3. The next lemma states that such a phase eventually ends with at least one multiplicity and hence one of the other phases starts.

Lemma 3. *Phase* MULTIPLICITY-CREATION *terminates with at least one multiplicity after a finite number of moves.*

Proof. From the description provided before this lemma, it follows that the graph in Fig. 3 models the execution of phase MULTIPLICITY-CREATION. We now show that all the cycles are traversed a finite number of times. This implies that eventually at least one multiplicity is created.

From Property 1, and results in [21], and [8] follows that the self-loops in W1, W2, and W3, respectively, are traversed a finite number of times.

The self-loop in W7 is traversed by performing REDUCTION or PENDING_REDUCTION. Each time such moves are performed, the supermin decreases until, after a finite number of moves, it either creates a multiplicity or leads to configurations in W4 or W6. The number of moves is at most two times the size of the initial supermin and this is obtained for symmetric configurations with the axis not passing through the supermin.

The self-loop in W4 and the cycle between W4 and W6 are traversed by performing XN. Each time this happens, the interval between the two symmetric robots closest to the axis of symmetry (excluding those adjacent to the supermin) is reduced until creating a multiplicity on the axis. The number of moves performed equals the initial size of such an interval. □

3.2 Further notes on the algorithm

We can show (see [9]) that all the types of configurations defined for the 5 phases are pairwise disjoint and that they cover all the possible configurations reachable by the algorithm.

Given a configuration C, in order to distinguish among the types, it is sufficient for a robot to compute simple parameters: number of nodes in the ring; number of multiplicities; number of robots (if possible) or number of occupied nodes; distances between robots and multiplicities; if C is symmetric; if C is at one move from the symmetries allowed by the algorithm.

The starting configuration can only belong to W1–W7. By Lemma 3, it follows that after a finite number of moves any other phase can be reached. Moreover, once reached a configuration with at least one multiplicity, the algorithm

never goes back to configurations without multiplicities, but for a bounded number of times on some symmetric configurations with 6 robots.

The possible interactions among the phases are shown in Fig. 2, and we prove that all the possible cycles can be traversed a limited number of times, until reaching phase CONVERGENCE without leaving it anymore.

4 Conclusion

The proposed algorithm answers to the posed conjectures concerning the gathering on the studied model by providing a complete characterization for the initial configurations. The obtained result is of main interest for robot-based computing systems. In fact, it closes all the cases left open with the exception of potentially gatherable configurations in $SP4$. Our technique, mostly based on the supermin concept, may result as a new analytical approach for investigating related distributed problems.

References

1. S. Alpern. The rendezvous search problem. *SIAM J. Control Optim.*, 33:673–683, 1995.
2. E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, and A. Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *Proc. of the 24th Int. Symp. on Distributed Computing (DISC)*, volume 6343 of *LNCS*, pages 297–311, 2010.
3. L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *Proc. of the 24th Int. Symp. on Distributed Computing (DISC)*, volume 6343 of *LNCS*, pages 312–327. Springer, 2010.
4. J. Chalopin and S. Das. Rendezvous of mobile agents without agreement on local orientation. In *Proc. of the 37th Int. Conf. on Automata, Languages and Programming (ICALP)*, volume 6199 of *LNCS*, pages 515–526, 2010.
5. A. Cord-Landwehr, B. Degener, M. Fischer, M. Hüllmann, B. Kempkes, A. Klaas, P. Kling, S. Kurras, M. Märtens, F. M. A. Der Heide, C. Raupach, K. Swierkot, D. Warner, C. Weddemann, and D. Wonisch. A new approach for analyzing convergence algorithms for mobile robots. In *Proc. of the 38th Int. Conf. on Automata, Languages and Programming (ICALP)*, volume 6756 of *LNCS*, pages 650–661, 2011.
6. J. Czyzowicz, A. Labourel, and A. Pelc. How to meet asynchronously (almost) everywhere. In *Proc. of the 21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 22–30, 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 the 19th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 7355 of *LNCS*, pages 327–338, 2012.
8. G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering of six robots on anonymous symmetric rings. In *Proc. of the 18th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 6796 of *LNCS*, pages 174–185, 2011.

9. G. D'Angelo, G. Di Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. Rapport de recherche RR-7963, INRIA, 2012.
10. B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *Proc. of the 23rd ACM Symp. on Parallelism in algorithms and architectures (SPAA)*, pages 139–148, 2011.
11. A. Dessmark, P. Fraigniaud, D. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46:69–96, 2006.
12. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*. To appear.
13. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theoretical Computer Science*, 411(14–15):1583 – 1598, 2010.
14. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337:147–168, 2005.
15. T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Randomized gathering of mobile robots with local-multiplicity detection. In *Proc. of the 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 5873 of *LNCS*, pages 384–398, 2009.
16. T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *Proc. of the 17th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 6058 of *LNCS*, pages 101–113, 2010.
17. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations. In *Proc. of the 18th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 6796 of *LNCS*, pages 150–161, 2011.
18. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Springer-Verlag, 2012. To appear.
19. R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci.*, 411:3235–3246, 2010.
20. R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390:27–39, 2008.
21. M. Koren. Gathering small number of mobile asynchronous robots on ring. *Zeszyty Naukowe Wydzialu ETI Politechniki Gdanskiej. Technologie Informacyjne*, 18:325–331, 2010.
22. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.
23. M. Yamashita, S. Souissi, and X. Défago. Gathering two stateless mobile robots using very inaccurate compasses in finite time. In *Proc. of the 1st int. Conf. on Robot communication and coordination (RoboComm)*, pages 48:1–48:4, 2007.