How to gather asynchronous oblivious robots on anonymous rings

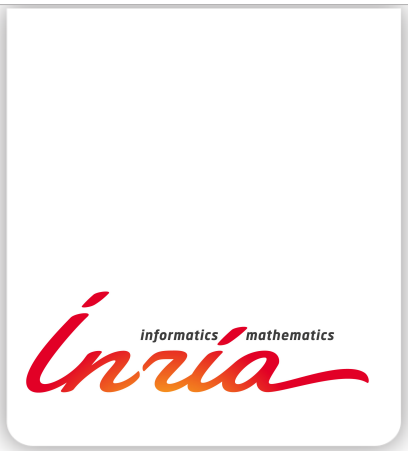
Gianlorenzo d'Angelo, Gabriele Di Stefano, Alfredo Navarra

How to gather asynchronous oblivious robots on anonymous rings

Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra

RESEARCH REPORT
N° 7963
May 2012
Project-Team MASCOTTE

ISSN 0249-6399 ISRN INRIA/RR--7963--FR+ENG



How to gather asynchronous oblivious robots on anonymous rings

Gianlorenzo D'Angelo*, Gabriele Di Stefano†, Alfredo Navarra‡

Project-Team MASCOTTE

Research Report n° 7963 — May 2012 — 24 pages

Abstract: A set of robots arbitrarily placed on the nodes of an anonymous graph have to meet at one common node and remain in there. This problem is known in the literature as the *gathering*. 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 performed asynchronously for each robot. Moreover, each robot is empowered by the so called *multiplicity detection* capability, that is, a robot 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 non-gatherable configurations. It is worth noting that our new algorithm makes use of a unified and general strategy for any initial configuration.

Key-words: distributed algorithm, asynchronous system, gathering, oblivious robots

* MASCOTTE Project, INRIA/I3S(CNRS/UNSA), France. gianlorenzo.d_angelo@inria.fr

† Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Italy. gabriele.distefano@univaq.it

‡ Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy. alfredo.navarra@unipg.it

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Comment rassembler robots asynchrones et inconscients sur les anneaux anonymes

Résumé : Un ensemble de robots placés arbitrairement sur les sommets d'un graphe anonyme doivent se rencontrer sur un sommet commun. Ce problème est connu dans la littérature comme le *gathering*. Les robots utilisent des cycles Look-Compute-Move; dans un cycle, un robot prend un instantané de la configuration actuelle (Look), décide de rester inactif ou de se déplacer sur l'un de ses voisins (Compute), et dans ce cas, fait le déplacement (Move). Les cycles sont exécutés de manière asynchrone pour chaque robot. Chaque robot possède la capacité de *multiplicity detection*: un robot est capable de détecter au cours de son opération Look si un sommet est vide, occupé par un robot, ou occupé par un nombre indéfini de robots. Le problème décrit a été largement étudié au cours des dernières années. Toutefois, les solutions connues ne sont valides que pour des configurations initiales spécifiques. Nous fournissons un algorithme qui résout le problème général, et est capable de détecter toutes les configurations initiales pour lesquelles le problème est impossible. Il est intéressant de noter que notre nouvel algorithme utilise une stratégie unifiée et générale pour chaque configuration initiale.

Mots-clés : distributed algorithm, asynchronous system, gathering, oblivious robots

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., [7, 8, 14] and references therein). In particular, we consider oblivious robots initially located at different nodes of an anonymous ring that have to gather at the same common node (not determined in advance) and remain in there. Neither nodes nor links have any labels. Initially, some of the nodes of the ring are occupied by robots and there is at most one robot in each node. 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. The only constraint is that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges.

Robots are all identical, and execute the same deterministic algorithm. Moreover, they have the so called *multiplicity detection* capability (see e.g. [15]), that is the ability to perceive during the Look operation whether a node is empty, or occupied by one robot, or occupied by an undefined number of robots greater than one. Without this capability the gathering would be impossible [17].

1.1 Related Work and Our Results

The problem of let meet mobile entities on graphs [2, 9, 17] or open spaces [5, 8] has been extensively studied in the last decades. When only two robots are involved, the problem is usually referred to as the *rendezvous* problem [1, 4, 6, 9, 19]. Under the Look-Compute-Move model, many problems have been addressed, like the *graph exploration* and the *perpetual graph exploration* [3, 10, 12, 13], while the rendezvous problem has been proved to be unsolvable on rings [17].

Concerning the gathering, different types of 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 [17], 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 [16], 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 addressed in [18]. The remaining ones, that is when there is an axis of symmetry of type node-edge and the odd interval cut by the axis is bigger than the even one, are, in general, not gatherable. As outlined in [16], it is very easy to see the impossibility to provide a successful strategy for a

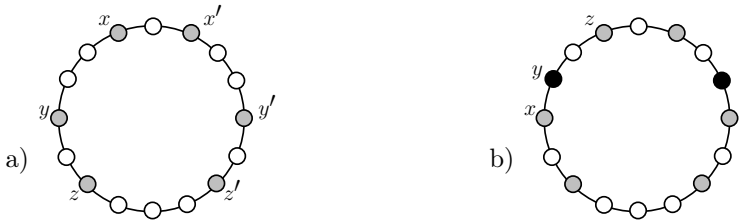


Figure 1: a) The intervals between robots y, z and y', z' are the supermins, while the supermin configuration view is $(1, 2, 1, 2, 1, 3)$. b) Black nodes represent multiplicities.

configuration of 4 robots on a five nodes ring. More complicated, and formally proven in [11], is the case provided by instances with the described property of 4 robots on a seven nodes ring. In what follows, we refer to all such ungatherable configurations as the set $SP4$. Finally, the case of 6 robots with an initial axis of symmetry of type node-edge, or node-node has been solved in [7].

Besides the cases left open, a unified algorithm that handles all the above cases is also missing. In this paper, we present a new distributed algorithm for solving completely all the gatherable configurations. 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 [18] and [7], respectively. Also, we exploit:

Property 1.1 *[17] 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 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 for all the initial configurations. The main result of this paper can be stated as follows.

Theorem 1.2 *There exists a distributed algorithm for gathering any initial configuration of more than two robots on a ring, provided that it does not belong to the set $SP4$, it is aperiodic, it does not admit an edge-edge axis of symmetry.*

2 Definitions and Preliminaries

We consider an n-nodes anonymous ring without orientation. Initially, $\mathbf{k}$ nodes of the ring are occupied by $\mathbf{k}$ 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 $Q(r) = (q_0, q_1, \ldots, q_j)$, $j \leq \mathbf{k} - 1$, that represents the sequence of the numbers of free consecutive nodes broken up by robots when traversing the ring in one direction, starting from r. 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, for any robots in y of Fig. 1b, $Q(y) = (-1, 1, 1, 1, -1, 0, 1, 3, 1, 0)$. 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 2.1 *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;
- If it is one of the robots allowed to move (during the Compute phase).

Definition 2.1 *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 2.1, is exploited to detect possible symmetry or periodicity features of a configuration:

Lemma 2.2 *Given a configuration C:*

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 Let $C = (q_0, q_1, \ldots, q_j)$,

1.$\Rightarrow$) 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 supermin and so forth. Since by hypothesis, supermin is unique, there must exist at least 2 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 1 copy of the supermin.

1.$\Leftarrow$) 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 1 axis of symmetry passing through a supermin, then there exists a unique $0 \leq i \leq j$ such that $C^{SM} = C_i = \overline{(C_i)}$ as, otherwise, by Property 2.1, it would imply the existence of other axes of symmetry, each 1 corresponding to 1 supermin.

2.$\Rightarrow$) If $|I_C| = 2$ and C is asymmetric, then by Property 2.1, 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 symmetry orthogonal to the other 1 that reflects the supermin into the other supermin. Hence, C would be periodic.

2.$\Leftarrow$) 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 1. Moreover, there cannot be more than 2 supermins, as by definition of supermin, these would imply other axes of symmetry and hence C would be periodic by Property 2.1. 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 smaller.

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

3.⇐) If C is periodic with period 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 chapter that all the other configurations are gatherable. From now on, we do not consider the above non-gatherable 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 of 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 is realized. In fact, 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 did not start its Look-Compute-Move cycle, 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.

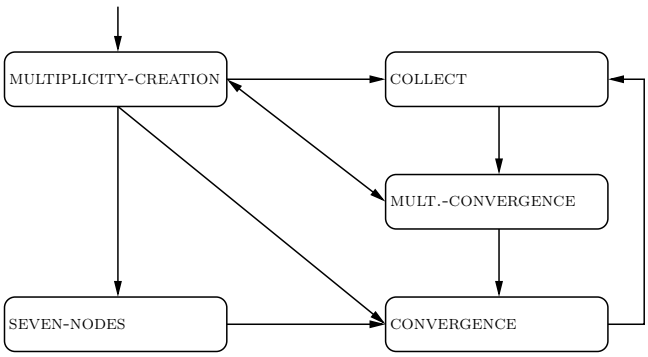


Figure 2: Phases interchanges.

3 Resolution algorithm and its correctness

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 two symmetric 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 almost all the robots into the two multiplicities and then to join the two multiplicities into a single one. After that, phase CONVERGENCE is performed. Special cases of 7 nodes rings and 6 robots are considered separately in phase SEVEN-NODES.

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 in this phase is to reduce the supermin by enlarging the largest interval adjacent to it as follows:

Definition 3.1 *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 given in Fig. 1. 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 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

Procedure: REDUCTION
Input: $C = (q_0, q_1, \ldots, q_j)$
1 **if** $C = C^{SM}$ **then** move towards q_0;

only one supermin and it is not symmetric, 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 3.1 *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. Then: 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 2.2, 2 cases may arise: there exists only 1 supermin in C or the configuration is symmetric and contains exactly 2 supermins.

We now show that in the case that there exists only 1 supermin in C, then C' is at least 2 moves from any axis of symmetry different from that passing through the supermin. By Lemma 2.2, it is enough to show that C' requires more than 1 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 2.2, 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 $i = 1, 2, \ldots, j$, 2 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 2 units. It follows that C' is at least 2 moves from any axis of symmetry different from that passing through the supermin. In fact, in order to obtain another axis of symmetry by performing only 1 move on C', $(C'^{SM})_i$ has to differ from C'^{SM} by at most 1 unit. This is enough to show the statement for the case of symmetric configurations with exactly 1 supermin. Regarding the asymmetric case, it remains to show that C' is at least 2 moves from 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 supermin configuration view is $C'^{SM} = (q'_0, q'_1, \ldots, q'_j) = (q_0 - 1, q_1, \ldots, q_j + 1)$ and 2 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 2 moves from the axis of symmetry 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 $i \in \{1, 2, \ldots, j\} \setminus \{h\}$ 2 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 2 units. Similar arguments to the ones used for the asymmetric case can show that C' is at least 2 moves far apart from the axis passing through the supermin. □

Procedure: MULTIPLICITY-CREATION
Input : CT, $C = (q_0, q_1, \ldots, q_j)$

```
case CT =W1
    if C = C_j then move towards q_0;
case CT =W2
    if GATHER-FOUR-NODES(C) then REDUCTION (C);
case CT =W3
    if GATHER-SIX-NODES(C) then REDUCTION (C);
case CT =W4
    if C_j = C_j and q_j is odd then move towards q_j;
    if C = C and q_0 is odd then move towards q_0;
case CT =W5
    if C = C^SSM then move towards q_0;
case CT =W6
    Let k the index such that
    (SYMMETRIC(q_0, q_1, ..., q_k + 1, q_{k+1} - 1, ... q_j) and q_k + 1 is odd) or
    (SYMMETRIC(q_0, q_1, ..., q_k - 1, q_{k+1} + 1, ... q_j) and q_{k+1} + 1 is odd);
    if k = 0 or k = j then move towards q_k;
    if no such k exists then REDUCTION (C);
case CT =W7
    if SYMMETRIC(C) then
        REDUCTION (C);
    else
        (b, Ĉ) = CHECKREDUCTION (C);
        if b then
            PENDING REDUCTION (C);
        else
            REDUCTION (C);
```

Figure 3: Procedure MULTIPLICITY-CREATION.

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 3.2 *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.

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

```
1 for i = 0, 1, ..., j do
2     if C = C̄_i then return true;
3 return false;
```

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 C̄_k = C^SM;
2 if q_{(k−1) mod j+1} > q_{(k+1) mod j+1} then
3     Ĉ := (q_0, q_1, ..., q_{(k−1) mod j+1} − 1, q_k + 1, ..., q_j);
4 else
5     if q_{(k−1) mod j+1} < q_{(k+1) mod j+1} then
6         Ĉ := (q_0, q_1, ..., q_k + 1, q_{(k+1) mod j+1} − 1, ..., q_j);
7     else return (false,∅);
8 if SYMMETRIC(Ĉ) then return (true, Ĉ);
9 return (false,∅);
```

As shown above, Procedure SYMMETRIC checks whether a configuration C is symmetric by exploiting Property 2.1. 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 been reduced by a possible REDUCTION. Then, at lines 2–6, 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. In this case, 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 not gatherable 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:

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;

Definition 3.3 *Let C be a configuration:*

- *If C is symmetric and there are no multiplicities,* XN *corresponds to move 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;*[1]
- *If C is symmetric and there is only one multiplicity,* XN *corresponds to move 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 move 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 move 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 perform. 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 one supermin of size 0 or with two 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 one supermin of size 0 or with two 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

[1] By Lemma 2.2, in gatherable configurations, the axis of symmetry cannot pass through two supermins hence there are always two robots allowed to move.

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 later in this section and the configurations with multiplicities will be described in the next section. Regarding configurations in W1, from Property 1.1, 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 [18] and [7] 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 [18] and [7] 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 3.1 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 3.2). 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.

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

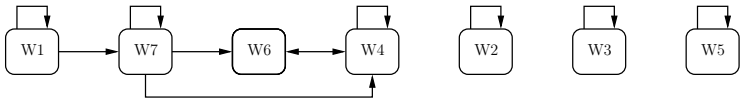


Figure 4: Phase MULTIPLICITY-CREATION.

This phase of the algorithm is summarized in Fig. 4, and it is realized by means of Procedure MULTIPLICITY-CREATION given in Figure 3. The next lemma states that MULTIPLICITY-CREATION eventually finishes with at least one multiplicity and hence one of the other phases starts.

Lemma 3.2 *Phase* MULTIPLICITY-CREATION *terminates with at least one multiplicity after a finite number of moves.*

Proof From the description it follows that the graph in Fig. 4 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 1 multiplicity is created.

From Property 1.1, and results in [18], and [7] 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 2 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 2 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 Phase COLLECT

Before proceeding with the description of this phase, we need to distinguish among the 2 poles of an axis of symmetry in the case that there are 2, 3 or 4 multiplicities. We call one of the poles `north`, according to the following definition.

Definition 3.4 *In a symmetric, aperiodic configuration with 2, 3 or 4 multiplicities, if the axis is of type*

- *node-edge, then the* ***north*** *is the middle node of the odd interval crossed by the axis;*
- *node-node or robot-robot, then let us consider the 2 sub-rings which are crossed by the axis and between 2 symmetric multiplicities. The* ***north*** *is the middle node of the smallest sub-ring, if the 2 sub-rings have different sizes. If the 2 sub-rings have the same size, in case the symmetry is of type robot-robot, then the* ***north*** *is the node occupied by the robot crossed by the axis from which by reading all the configuration 1 obtains the lexicographically smallest string; in case the symmetry is of type node-node, then the* ***north*** *is the middle node of the interval crossed by the axis from which by reading all the configuration 1 obtains the lexicographically largest string.*

The other node or edge crossed by the axis is called ***south****.*

We assume that in symmetric configurations with multiplicities, a robot reads a configuration starting from the northern interval between the 2 adjacent ones, we also assume that if the robot belongs to a multiplicity the first interval q_0 is equal to -1. For instance, robots x and z of Fig. 1b, read $(1, 1, -1, 0, 1, 3, 1, 0, -1, 1)$ and $(0, -1, 1, 1, 1, -1, 0, 1, 3, 1)$, respectively. In asymmetric configurations with multiplicities, we need to recognize the `north` of a possible symmetric configuration from which the current configuration has been obtained. In case that there is only 1 multiplicity, let us consider the configuration read by a robot in an arbitrary order $C = (q_0, q_1, \ldots, q_i, \ldots, q_j)$, where $q_i = -1$. The robot first verifies whether there is another symmetric multiplicity to be created by checking whether the configuration C' defined as $C' = (q_0, q_1, \ldots, q_{i-1} - 1, 0, \ldots, q_j)$ if $q_{i-1} > q_{i+1}$ or $C' = (q_0, q_1, \ldots, 0, q_{i+1} - 1, \ldots, q_j)$ if $q_{i-1} \leq q_{i+1}$ is symmetric. In the affirmative case, the robot chooses the proper reading between C and $\overline{C_j}$ according to the minimal between C' and $\overline{C'_j}$. Otherwise and in the case that $q_{i-1} = q_{i+1}$, it chooses C if $\sum_{\ell=0}^{i-1} q_\ell \leq \sum_{i+1}^{j} q_\ell$ and $\overline{C_j}$ otherwise, i.e. the 1 towards the multiplicity. In case there are 2 multiplicities, we need to define a `north` node allowing a robot to read the configuration starting from the northern interval among its adjacent ones. To this aim, the robot looks for an interval or a robot which is at the same distance from the 2 multiplicities. If there exists an interval of even size at the same distance from the 2 multiplicities, then the middle edge of such interval is identified as `south` and the node at distance $\lfloor \frac{n}{2} \rfloor$ from the `south` is identified as the `north`. In any other case, we compare the sizes of the 2 sub-rings between the 2 multiplicities. If they are different, we consider the configuration obtained by removing all the single robots adjacent to the multiplicities and the middle robot or the middle node of the middle interval of the smallest sub-ring is identified as the `north`. If they are equal, the algorithm ensures that there always exists an odd size interval or a robot at the same distance from the 2 multiplicities. Therefore, we lexicographically compare the configurations read by such interval (robot, resp.) with that read from the node at distance $\frac{n}{2}$ from the middle of this interval (from the robot, resp.) and consider the largest (smallest, resp.) as the `north` (`south`, resp.) and the other as `south` (`north`, resp.).

Phase COLLECT is performed if one of the next configurations occurs:

COLL-A-1 Asymmetric configurations with more than 6 nodes occupied, 1 multiplicity at more than 1 move from the symmetry passing through the multiplicity and at one move from the symmetry before performing REDUCTION or reducing the second supermin;

COLL-S-2 Symmetric configurations with 2 multiplicities and

a) more than 6 nodes occupied;
b) 6 nodes occupied and more than 1 single robot at distance grater than 0 from any multiplicity on the northern side;
c) 6 nodes occupied and more than 2 single robots at distance grater than 0 from any multiplicity on the southern side;

COLL-A-2 Asymmetric configurations with 2 multiplicities and

a) more than 6 nodes occupied;
b) 6 nodes occupied and 0, 1 or 3 single robots adjacent to a multiplicity.

When the configuration is in COLL-A-1, then the algorithm leads to a symmetric configuration within 1 move which is necessarily containing 2 multiplicities. In fact, the original configuration was at more than 1 move from the symmetry passing through the multiplicity. After this move, a configuration in COLL-S-2 is obtained. Then, the algorithm generates only configurations in COLL-S-2 or in COLL-A-2.

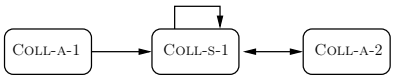


Figure 5: Phase COLLECT.

When the configuration is in Coll-s-2 we need to distinguish among the types of symmetry. In node-node and node-edge symmetries, we consider the 2 symmetric robots which are the closest to the multiplicities among those in the nodes between the multiplicities and the `north`. The algorithm moves this 2 robots symmetrically (generating again configurations in Coll-s-2 or in Coll-a-2 if the symmetric robots move synchronously or not, respectively) until they join the 2 multiplicities. This operation is iteratively performed by all the robots between the multiplicities and the `north`. The last 2 robots are allowed to join the multiplicities only if there are more than 6 nodes occupied. Otherwise, these 2 robots are moved until they become neighbors of the multiplicities. Then, the robots between the multiplicities and the `south`, except for the 2 southern symmetric robots, perform the same operation in turn, starting by the 2 symmetric robots closest to the multiplicities.

In robot-robot symmetric configurations, the behavior is similar but it is realized by first collecting the robots on the `south` and then those on the `north`, apart from the robots on the axis. In both cases, the algorithm eventually leads to a symmetric configuration with six nodes occupied by 2 multiplicities and 4 single robots, with at least 2 of them neighbors to the multiplicities. If there are at least 2 single robots on the northern side, then 2 of them are neighbors to the multiplicities. The pseudo-code of procedure COLLECT is given in Fig. 6. The next lemma shows that this phase eventually terminates in the described symmetric configuration.

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

case CT =Coll-a-*1*
  PENDING REDUCTION (C);
case CT =Coll-s-*2*
  if *The symmetry is node-node or node-edge* **then**
    if $q_{j-1} = -1$ *and* $q_j > 0$ **then**
      move towards q_j
    if $q_1 = -1$ *and* $q_2 = q_4 = 0$ *and* $q_5 = -1$ *and* $q_{j-2} \neq -1$ **then**
      move towards q_0
  if *The symmetry is robot-robot* **then**
    if $q_1 = -1$ *and* $q_{j-1} \neq -1$ **then**
      move towards q_0
    if $q_{j-1} = -1$ *and* $q_j > 0$ *and* $q_{j-2} = q_{j-3}$ *and* $q_{j-4} = -1$ **then**
      move towards q_j
case CT =Coll-a-*2*
  if $q_1 = -1$ *and* $((q_0 = 0$ *and* SYMMETRIC$(q_1, q_2, \ldots, q_j + 1))$ *or* $(q_0 > 0$ *and* SYMMETRIC$(q_0 - 1, q_1, \ldots, q_j + 1)))$ **then**
    move towards q_0;
  if $q_{j-1} = -1$ *and* $((q_j = 0$ *and* SYMMETRIC$(q_0 + 1, q_1, \ldots, q_{j-1}))$ *or* $(q_j > 0$ *and* SYMMETRIC$(q_0 + 1, q_1, \ldots, q_j - 1)))$ **then**
    move towards q_j;

Figure 6: Procedure COLLECT.

Lemma 3.3 *Phase* COLLECT *terminates after a finite number of moves by reaching a symmetric configuration with 6 nodes occupied by 2 multiplicities and 4 single robots, with at least 2 of them neighbors to the multiplicities. If there are at least 2 single robots on the northern side, then 2 of them are neighbors to the multiplicities.*

Proof From the above description, it follows that the graph in Figure 5 models the execution of phase COLLECT. We now show that all the cycles are traversed a finite number of times until one of the two configurations in the statement is created.

There are only 2 cycles in the graph: the self-loop in COLL-S-2 and the bidirectional edge between COLL-S-2 and COLL-A-2. Each time that one of these 2 cycles is traversed, either the distance between a multiplicity and a robot or the number of single robots is decreased until 6 nodes are occupied and 4 single robots with at least 2 of them are neighbors to the multiplicities. In fact, if at least 2 of the 4 single robots reside on the northern side, then 2 of them must be neighbors to the multiplicities, as otherwise the configuration would belong to COLL-S-2, and the corresponding move can be performed, hence reducing the distance between single robots and multiplicities. □

3.3 Phase MULTIPLICITY-CONVERGENCE

Phase MULTIPLICITY-CONVERGENCE is performed if one of the next configurations occurs:

MC-S-x Symmetric configurations with

a) 2 multiplicities with more than 7 nodes and exactly 4 nodes occupied,

b) 2 multiplicities and 6 nodes occupied with at least 1 single robot adjacent to each multiplicity;

c) 3 multiplicities or 2 multiplicities and exactly 5 nodes occupied;

d) 4 multiplicities;

MC-A-x Asymmetric configurations with

a) 2 multiplicities, more than 7 nodes and less than 6 nodes occupied;

b) 2 multiplicities, 6 nodes occupied and at most 2 robots at distance grater than 0 from the multiplicities;

c) 3 multiplicities;

MC-A-1 Asymmetric configurations with more than 7 nodes, five nodes occupied, 1 multiplicity at more than 1 move from the symmetry passing through the multiplicity and at one move from a symmetry allowed by algorithm in [7].

From MC-S-x, if there are 2 multiplicities, the algorithm moves them towards `north` if exactly 4 nodes are occupied, or if exactly 6 nodes are occupied with 1 single robot adjacent to the northern side of each multiplicity. If exactly 6 nodes are occupied with 1 single robot adjacent to the southern side of each multiplicity, the algorithm moves towards `north` the 2 symmetric robots adjacent to the multiplicities, making such robots joining the multiplicities.

This operations may lead to configurations in MC-S-x (with the northern side reduced), or in MC-A-x.

The configurations in MC-S-x with 3 multiplicities or 2 multiplicities and exactly 5 nodes occupied can occur only when the 3 multiplicities are consecutive or there are 3 consecutive

nodes occupied with 1 single robot is in the middle of 2 multiplicities. In both cases, the middle robot/multiplicity is crossed by the axis. In these cases, the algorithm moves the 2 external multiplicities towards the middle robot/multiplicity leading to a symmetric configuration with only 1 multiplicity crossed by the axis or to an asymmetric configuration with 2 multiplicities, at one move from the symmetry passing through one of the two multiplicities.

If the configuration is in Mc-s-x and it has 4 multiplicities the algorithm moves the southern multiplicities towards the 2 symmetric northern ones, hence obtaining a configuration still in Mc-s-x with 4 multiplicities (but with less robots to move upwards), or in Mc-s-x with 2 multiplicities (but with a largest southern side), or to an asymmetric configuration in Mc-a-x with 2 or 3 multiplicities.

From Mc-a-x, the algorithm allows only moves towards `north` that can re-establish the symmetry. Note that, unless the configuration is made of just 6 robots, there will always be at least 2 multiplicities. The case of 6 robots, instead, may lead to a configuration in W3 or in Mc-a-x. The pseudo-code of Procedure MULTIPLICITY-CONVERGENCE is given in Fig. 7.

Procedure: MULTIPLICITY-CONVERGENCE
Input : CT, $C = (q_0, q_1, \ldots, q_j)$

case CT =Mc-s-x
  if $q_0 = -1$ *and* OCCUPIED$(C) = 4$ *and* $|\{q_i = -1, 0 \leq i \leq j\}| = 2$ **then**
    move towards q_0
  else
    if $q_0 = -1$ *and* $C \neq \overline{C}$ *and* *(*$|\{q_i = -1, 0 \leq i \leq j\}| = 3$ *or* *(*OCCUPIED$(C) = 5$ *and* $|\{q_i = -1, 0 \leq i \leq j\}| = 2$*))* **then**
      move towards q_0
    else
      if $q_0 = -1$ *and* $|\{q_i = -1, 0 \leq i \leq j\}| = 4$ *and* $q_1 = -1$ **then**
        move towards q_0
      else
        if *(*$q_0 = 0$ *and* $q_1 = -1$*)* *or* *(*$q_0 = -1$ *and* $q_1 = 0$*)* **then**
          move towards q_0
case CT =Mc-a-x
  if OCCUPIED$(C) < 6$ *and* $\sum_{q_i \geq 0}(q_i + 1) > 7$ *and* $|\{q_i = -1, 0 \leq i \leq j\}| = 2$ **then**
    if OCCUPIED$(C) = 5$ *and* *((*$q_0 = 0$ *and* $q_1 = -1$*)* *or* *(*$q_0 = -1$ *and* $q_1 = 0$*))* **then**
      move towards q_0
    if OCCUPIED$(C) = 4$ *and* $q_0 = -1$ *and* SYMMETRIC$((-1, q_1 + 1, -1, q_3 - 1, q_4, q_5))$ **then**
      move towards q_0
  else
    if $|\{q_i = -1, 0 \leq i \leq j\}| = 3$ *and* $q_0 = -1$ *and* $q_1 = -1$ **then**
      move towards q_0
    else
      if *(*$q_0 = 0$ *and* $q_1 = -1$*)* *or* *(*$q_0 = -1$ *and* $q_1 = 0$*)* **then**
        move towards q_0
case CT =Mc-a-*1*
  GATHER-SIX-NODES(C)

Figure 7: Procedure MULTIPLICITY-CONVERGENCE.

Lemma 3.4 *Phase* MULTIPLICITY-CONVERGENCE *terminates after a finite number of moves with a configuration composed of 1 multiplicity and between 1 and 4 further single robots.*

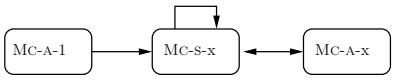


Figure 8: Phase MULTIPLICITY-CONVERGENCE.

Proof From the above description, it follows that the graph in Figure 8 models the execution of phase MULTIPLICITY-CONVERGENCE. We have already shown that all the cycles are indeed traversed a finite number of times until a configuration as described in the statement is achieved. In fact, for each described configuration, a set of robots is allowed to move towards `north`, reducing the northern part, or enlarging the southern part, or decreasing the number of robots that have to move towards `north`. The only exception is given by the 6 robots case, which however is guaranteed to converge to the desired symmetric configuration by means of the arguments in [7]. □

3.4 Phase CONVERGENCE

Phase CONVERGENCE is performed if one of the next configurations occurs:

CONV-A-S Asymmetric configurations with 1 multiplicity at one XN move from configurations also achievable when gathering from six robots, as described in Figure 9. Formally, the configurations are: $(-1, q_1, q_1-1, 0, q_4, q_4+2)$, $(-1, q_1, q_2, q_1-2, 0, q_5)$, and $(-1, q_1, q_2, q_1-1, 0, q_5, 0)$.

CONV-S-1 Symmetric configurations with 1 multiplicity having a number of nodes different from 7 or a number of nodes occupied different from 5;

CONV-A-1 Asymmetric configurations with 1 multiplicity;

a) at one XN move from the symmetry passing through the multiplicity and different from the configurations in CONV-A-S;

b) at more than 1 move from any symmetry.

In this phase, the idea is to gather all the remaining single robots to the only multiplicity by performing XN. However there might occur some exceptions that must be carefully considered. In fact, whenever the gathering leads to configurations with only 5 nodes occupied, these might be "confused" with configurations in phase MULTIPLICITY-CONVERGENCE obtainable when initially only 6 robots occur. In order to address such occurrences, we better avoid these situations. In particular, the algorithm do not perform an XN move whenever it leads to configurations to 1 move from a symmetric configuration with 2 multiplicities achievable when gathering from 6 robots (shown in Figure 9). These are the configurations in CONV-A-S: in this case we perform another move. That is, we allow again an XN move, but without considering the original robot which was supposed to move. In practice, this equals to move the first robot on the other side of the 1 planned to move with respect to the multiplicity. It is worth to note that the configurations in CONV-A-S are the only ones to take care of. The other 2 possible configurations are shown in Fig. 10. After the XN move, they led to a configuration to 1 move from a symmetric configuration with 2 multiplicities, but, in this case, the algorithm in [7] would never generate it and then there is no possible misclassification.

It must be observed that the above method applied from CONV-A-S decreases the number of moves required to move the single robots to the only multiplicity.

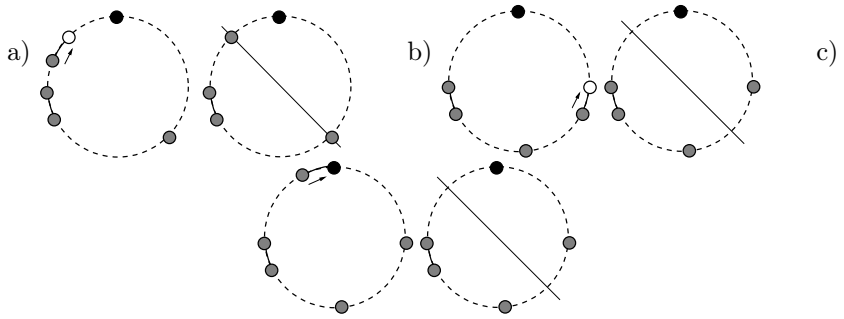


Figure 9: The 3 special types of configurations. A white circle represents an empty node, a grey 1 a node with a robot, and a black 1 a multiplicity. A dashed arc indicates an arbitrary sequence of empty nodes, a plain arc 2 consecutive nodes. If during the CONVERGENCE phase a robot should move towards the multiplicity, as indicated by the arrow, then the resulting configuration is at one move from a symmetric configuration with 2 multiplicities achievable when gathering from six robots. The symmetric configurations arise when, in each shown configuration, the 2 adjacent robots join into a multiplicity.

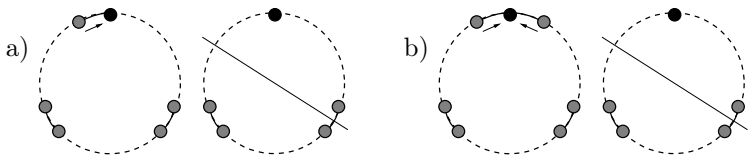


Figure 10: The 2 other special asymmetric configurations. After the moves indicated by the arrows the resulting configuration is at one move from a symmetric configuration with 2 multiplicities, but this symmetric configuration is never generated by the algorithm in [7].

From Conv-s-1, the algorithm allows by an xn move the 2 closest single robots to the multiplicity (or the only remaining 1) to move towards the multiplicity. If both make synchronously the move, then the obtained configuration is still in Conv-s-1, but with 1 move less before the final gathering. If only 1 robot performs the allowed move, then the configuration is in Conv-a-1. From Conv-a-1a, either there is only 1 single robot left, which will be the only 1 allowed to move towards the multiplicity until the gathering is completed, or there are at least 2 single robots, each of which has no other robots in between itself and the multiplicity. If the farthest among those 2 robots re-build the symmetry by 1 move towards the multiplicity, then it will be the only 1 allowed to move and the obtained configuration is in Conv-s-1, otherwise the other single robot is the only 1 allowed to move towards the multiplicity and the obtained configuration might either remain in Conv-a-1 or move to Conv-s-1. Also from Conv-a-1b, the xn move is performed.

In any case, the property that less moves remain is preserved until the final gathering. The pseudo-code of Procedure convergence is given in Fig. 11.

```
Procedure: CONVERGENCE
Input: CT, C = (q_0, q_1, ..., q_j)
case CT =Conv-a-s
    if q_1 = −1 and q_0 > q_2 then
        move towards q_0
case CT =Conv-s-1
    if q_1 = −1 then
        move towards q_0
case CT =Conv-a-1
    if q_1 = −1 then
        if q_0 = q_2 + 1 and SYMMETRIC(q_0, −1, q_2 + 1, q_3 − 1, q_4, ..., q_j) then
            move towards q_0
        else
            if C < C_2 then
                move towards q_0
```

Figure 11: Procedure convergence.

Lemma 3.5 *Phase* convergence *terminates after a finite number of moves finalizing the gathering or in a configuration in* Coll-a-1.

Proof From the above description, it follows that the graph in Figure 12 models the execution of phase convergence. We have already shown that all the cycles are indeed traversed a finite number of times until the gathering is completed. In fact, for each described configuration, 1 or 2 robots are allowed to move towards the multiplicity, reducing the number of moves necessary for each single robot to reach the multiplicity. Eventually the gathering is necessarily accomplished. The only exception occurs when a configuration in Conv-a-b with more than 6 nodes occupied

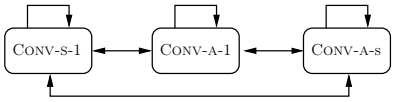


Figure 12: Phase convergence.

leads, after a REDUCTION, to a configuration in COLL-A-1. However this may happen only once. □

3.5 Phase SEVEN-NODES

Phase SEVEN-NODES is performed if one of the next configurations occurs:

7N-2 Configurations with 2 multiplicities on a ring of 7 nodes;

7N-1 Configurations with 1 multiplicity and 5 nodes occupied on a ring of 7 nodes.

All the above configurations can occur only when there are 6 robots on a ring of 7 nodes. In fact, for 7N-2 there cannot be initially more than six robots on a ring of 7 nodes, and no strategy leads to have 2 multiplicities apart from the case of 6 robots. For 7N-1, it may only happen starting from 6 single robots. In these cases, the algorithm for 6 robots from [7] is used.

3.6 Selecting the type of configuration

In this, section we show how to recognize the type of configurations among those described above.

We first show that all the above types are pairwise disjoint and that they cover all the possible configurations achieved by the algorithm. Let CTP = $\{W1, W2, \ldots, W7$, COLL-A-1, COLL-A-2, COLL-S-2, MC-S-x, MC-A-x, MC-A-1, CONV-A-s, CONV-A-1, CONV-S-1, $7\mathrm{N}-2$, $7\mathrm{N}-1\}$ be the set of all the possible configuration types above.

Corollary 3.6 *Configurations in* CTP *covers all the possible configurations achieved by the algorithm.*

Proof It is easy to see that the configurations in W1–W7 cover all the possible initial configurations, as by hypothesis, any initial configuration has no multiplicities. All the configurations in phases COLLECT, MULTIPLICITY-CONVERGENCE, and CONVERGENCE are generated by the algorithm. In the description of the phases, it has been shown that from any configuration, the configuration after the performed move is in CTP. Therefore, the statement follows. □

Lemma 3.7 *Configuration types in* CTP *are pairwise disjoint.*

Proof We compare only the configuration types with the same number of multiplicities and degree of symmetry. The only configuration types with no multiplicities are W1–W7 and they are clearly pairwise disjoint. The symmetric configuration types with 1 multiplicity are CONV-S-1 and 7N-1 which are disjoint as they differ for the number of nodes or the number of nodes occupied. The symmetric configuration types with 2 multiplicities are COLL-S-2, MC-S-a, MC-S-b, MC-S-c, and 7N-2 which are disjoint as they differ for the number of nodes or the number of nodes occupied, or the distance between the single robots and the multiplicities. The remaining symmetric configuration types have 3 or 4 multiplicities.

The asymmetric configuration types with 1 multiplicity are COLL-A-1, MC-A-1, CONV-A-1, CONV-A-S, and 7N-1. Configurations MC-A-1, CONV-A-S, and 7N-1 are disjoint with respect to the others as they differ for the number of nodes or the number of nodes occupied. Among them, 7N-1 has a different number of nodes from MC-A-1 and it differs from CONV-A-S as it is at one move from a symmetry with 2 multiplicities, whereas CONV-A-S needs at least 2 moves to

generate a symmetric configuration with 2 multiplicities. For the same reason Mc-a-1 differs from Conv-a-s.

It remains to compare configuration types belonging to Conv-a-1 and to Coll-a-1a and Coll-a-1b (these latter configurations are clearly different). Configurations in Conv-a-1 are at more than 1 move from the symmetry passing through the multiplicity, and this is not the case for configurations in Coll-a-1a. Moreover, they are at one move from a symmetry whereas Coll-a-1b are at more than 1 move from any symmetry.

The asymmetric configuration types with 2 multiplicities are Coll-a-2, Mc-a-a, Mc-a-b, and 7N-2 which are disjoint as they differ for the number of nodes or the number of nodes occupied, or the distance between the single robots and the multiplicities. There is only 1 remaining asymmetric configuration type with 3 multiplicities. □

By the proof of the above lemmata, follows that in order to distinguish among the configuration types, it is sufficient to compute, given a configuration $C = (q_0, q_1, \ldots, q_j)$, the following parameters:

1. number of nodes in the ring, $n(C)$;
2. number of multiplicities, $m(C)$;
3. number of nodes occupied or number of robots in the case without multiplicities, $\textsc{occupied}(C)$;
4. distance between single robots and multiplicities;
5. if C is symmetric;
6. if C belongs to Conv-a-s;
7. if C is at one move from one of the symmetries allowed by the algorithm;

Parameters 1–3 can be computed by formulas $n(C) = \sum_{q_i \geq 0}(q_i + 1)$, $m(C) = |\{q_i = -1, 0 \leq i \leq j\}|$, and $\textsc{occupied}(C) = j + 1 - m(C)$, respectively. The distance between single robots and multiplicities is easily computed by summing the intervals between a single robot and a multiplicity. The symmetry of a configuration is computed by procedure $\textsc{symmetric}(C)$. If C belongs to Conv-a-s can be easily checked by comparing all the intervals of C_i and $\overline{C_i}$, for each $i \in \{0, 1, \ldots, j\}$ to the configurations in Conv-a-s. To understand whether C is at one move from a symmetry allowed by the algorithm, it is sufficient to perform such move backwards and checking whether the obtained configuration is symmetric. As an example, to check whether C is at one move from reduction, Procedure check reduction is used.

3.7 The complete algorithm

In this section, we show how the above phases interact and how the algorithm can switch from 1 phase to another. We show that starting from any initial configuration among W1–W7, we necessarily end up with configurations in either Conv-a-1 or Conv-s-1 (possibly passing through phases collect and multiplicity-convergence), hence finalizing the gathering.

By Lemmata 3.2–3.5, follows that the interaction between the phases is that given in the graph in Figure 2. In what follows we show how the edges of such graph are traversed.

The starting configuration can only belong to W1–W7. By Lemma 3.2, follows that after a finite number of moves any other phase can be reached. Moreover, once reached a configuration with at least 1 multiplicity, the algorithm never goes back to configurations without multiplicities,

but in the case of 6 robots. In this latter case in fact, the configurations can swap between those in W3 and those in Mc-a-1 or in Mc-s-a. However, the number of times that this cycle can be traversed is finite as each time the `north` interval is reduced, until creating 1 single multiplicity on the final gathering node (that is, when the system moves to CONVERGENCE).

By Lemma 3.3, follows that phase COLLECT terminates after a finite number of moves in a configuration in phase MULTIPLICITY-CONVERGENCE. In particular, the algorithm can only move from configurations in Coll-a-1, Coll-s-2-b, Coll-s-2-c and Coll-a-2-b, to configurations in Mc-s-b.

By Lemma 3.4, follows that MULTIPLICITY-CONVERGENCE terminates after a finite number of moves in a configuration in phase CONVERGENCE. In particular, the algorithm can only move from configurations in Mc-s-a, Mc-s-c, Mc-a-a, and Mc-a-1, to configurations in Conv-s- and Conv-a-a.

From [7], follows that configurations in SEVEN-NODES terminates in configurations in CONVERGENCE.

Finally, from configurations in CONVERGENCE, the algorithm terminates the gathering with the only exception of the case where a configuration in Conv-a-b can lead, after a REDUCTION, to a configuration in Coll-a-1. Note that this case can occur only when there are more than 6 nodes occupied. As the transition from phase MULTIPLICITY-CONVERGENCE to phase CONVERGENCE can occur only when there are at most 6 nodes occupied, it follows that the edge between CONVERGENCE and COLLECT can be traversed only once.

4 Conclusion

Our distributed algorithm answers to the posed conjectures concerning the gathering on the studied model by providing a complete characterization for gatherable configurations. The obtained result is of main interest for robot-based computing systems. In fact, it closes all the cases left open while providing new analytical approaches. Our technique, mostly based on the supermin concept, provides a powerful mean for investigating related distributed problems. Moreover, the concept of supermin combined with the REDUCTION move can be used to solve other problems in the Look-Compute-Move model.

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, 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.

[8] 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.

[9] A. Dessmark, P. Fraigniaud, D. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46:69–96, 2006.

[10] S. Devismes, F. Petit, and S. Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *Proc. of the 16th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 5869 of *LNCS*, pages 195–208, 2009.

[11] G. Di Stefano and A. Navarra. About ungatherable configurations of oblivious and asynchronous robots on anonymous rings. Technical Report R.12-113, Dipartimento di Ingegneria Elettrica e dell'Informazione, Università dell'Aquila, 2012.

[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.

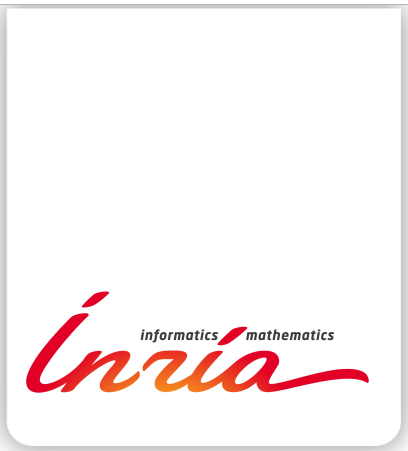
[15] T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Randomized gathering of mobile robots with local-multiplicity detection. In *Proc. of the 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 5873 of *LNCS*, pages 384–398, 2009.

[16] 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.

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

[18] M. Koren. Gathering small number of mobile asynchronous robots on ring. *Zeszyty Naukowe Wydzialu ETI Politechniki Gdanskiej. Technologie Informacyjne*, 18:325–331, 2010.

[19] 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.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

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

ISSN 0249-6399