# Chapter 1
# Gathering asynchronous and oblivious robots on basic graph topologies under the Look - Compute - Move model*

Gianlorenzo D'Angelo, Gabriele Di Stefano and Alfredo Navarra

**Abstract** Recent and challenging models of robot-based computing systems consider identical, oblivious and mobile robots placed on the nodes of anonymous graphs. Robots operate asynchronously in order to reach a common node and remain with it. This task is known in the literature as the *gathering* or *rendezvous* problem. The target node is neither chosen in advance nor marked differently compared to the other nodes. In fact, the graph is anonymous and robots have minimal capabilities. In the context of robot-based computing systems, resources are always limited and precious. Then, the research of the minimal set of assumptions and capabilities required to accomplish the gathering task as well as for other achievements is of main interest. Moreover, the minimality of the assumptions stimulates the investigation of new and challenging techniques that might reveal crucial peculiarities even for other tasks. The model considered in this chapter is known in the literature as the *Look-Compute-Move* model. Identical robots initially placed at different nodes of an anonymous input graph operate in asynchronous 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 makes an instantaneous move to this neighbor (Move). Cycles are performed asynchronously for each robot. This means that the time between Look, Compute, and Move operations is fi-

Gianlorenzo D'Angelo
MASCOTTE Project, INRIA/I3S(CNRS/UNSA), France. e-mail: gianlorenzo.d_angelo@inria.fr

Gabriele Di Stefano
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Italy. e-mail: gabriele.distefano@univaq.it

Alfredo Navarra
Department of Mathematics and Computer Science, University of Perugia, Italy.
e-mail: alfredo.navarra@unipg.it

* This work has been partially supported by the Fondazione Cassa di Risparmio della Provincia dell'Aquila (Italy) within project "ARISE" (Arising Robust Internetworked System for Emergency contexts).

nite 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 Hence, robots may move based on significantly outdated perceptions.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 perceives all other robots at nodes of the ring and not on edges. Robots are all identical, anonymous, and execute the same deterministic algorithm. They cannot leave any marks at visited nodes, nor can they send messages to other robots. In this chapter, we aim to survey on recent results obtained for the gathering task over basic graph topologies, that are rings, grids, and trees. Recent achievements to this matter have attracted many researchers, and have provided interesting approaches that might be of main interest to the community that studies robot-based computing systems.

## 1.1 Introduction

The chapter surveys on recent results in robot-based computing systems. Two or more robots, starting from distinct initial positions, have to meet at some place and remain there. The problem is known in the literature as the *gathering* problem while sometimes it is referred to as the *rendezvous* problem.

Different assumptions on the capabilities of the robots as well as on the environment where they move, lead to very different scenarios. To have an idea of the work done during the recent years, it is enough to mention that already five different surveys deal with such a problem from different perspectives. The first distinction considers the way the robots may take their decisions in order to move towards some directions. In fact, randomized algorithms can be applied for this purpose or full determinism might be required. For the former case, there is a comprehensive survey book [3] which also includes results contained in an older survey paper [2]. The latter case has captured more attention in recent studies. In particular, for the case where robots are considered to move along the nodes and edges of an input graph, the survey paper [25] and in a more extended form [26] present various scenarios and techniques for different graph topologies. Whereas, the survey book [24] focuses on the gathering over ring networks. In the literature, many results also concern the gathering of robots moving on a continuous two-dimensional Euclidean space have been devised. The interested reader may refer to [8, 15, 20, 27, 29] for the continuous case. However, a recent trend is to study discrete models like the case where robots move over graphs rather than the continuous case.

In this chapter, the aim is to provide in more details the strategies applied to accomplish the gathering task on basic graph topologies like rings, grids, and trees, under a very specific model that has attracted many researchers during the last years. Very few of such results are already contained in the aforementioned surveys. In fact, most of the results come from very recent papers and the last section contains original results for tree topologies.

The model considered in this chapter (sometimes also referred to as CORDA [27]) is known in the literature as the *Look-Compute-Move* model. Robots asynchronously run an operative cycle where first they perceive the global configu-

ration of the robots over the graph (Look phase). That is, during the Look phase a robot is able to perceive the relative locations of the other robots with respect to its own position on the graph. The only cases where a robot might be misled concern the so called *multiplicities*, i.e., when more than one robot occupy the same node. In this case, different assumptions might be considered. Based on the perceived configuration which might reveal the exact disposal of all the robots or just which nodes are occupied, a robot evaluates whether to stay idle or to move towards one of its neighboring nodes (Compute phase). Note that, since robots are asynchronous, the Compute phase might be accomplished by a robot based on outdated configurations perceived while other robots are performing their movements. Finally, the robot enters to the Move phase, where it simply applies the computed movement. Hence, it either remains on its current position or it moves towards the computed neighboring node. The only assumption in this phase is that the movements are instantaneous and hence robots are always perceived over nodes, and never over edges. Robots are all identical, anonymous, and execute the same deterministic algorithm. They cannot leave any marks at visited nodes, nor send messages to other robots. The scheduler that wakes the robots up is assumed to be fair, i.e., all the robots will wake up, eventually, and perform their Look-Compute-Move cycles infinitely many times.

Another assumption that can be considered concerns the ability for the robots to perceive information about the number of robots occupying the same node, during the Look operation. This ability is called the *multiplicity detection* capability and it has been sometimes exploited in various forms. In any case, a robot perceives whether a node is empty or not, but in the *global-strong* version, a robot is able to perceive the exact number of robots that occupy each node. In the *global-weak* version, a robot perceives only whether a node is occupied by one robot or if a multiplicity occurs, i.e., a node is occupied by an undefined number of robots greater than one. In the *local-strong* version, a robot can perceive only whether a node is occupied or not, but it is able to perceive the exact number of robots occupying the node where it resides. Finally, in the *local-weak* version, a robot can perceive the multiplicity only on the node where it resides but not the exact number of robots composing it.

In the context of robot-based computing systems, resources are always limited and precious. Then, the research of the minimal set of assumptions and capabilities required to accomplish the gathering task as well as for other achievements is of main interest. Moreover, the minimality of the assumptions stimulates the investigation of new and challenging techniques that might reveal crucial peculiarities even for other tasks.

Depending on the multiplicity detection capability version chosen for the robots, some scenarios may be unsolvable while some others are solvable. Intuitive concepts like symmetry or periodicity might be involved and sometimes are fundamental to the feasibility of the studied problems. Depending on the assumptions made, the definition of such concepts may vary and require different approaches. This is why in what follows, the same concept might be re-defined according to the current scope. Moreover, the considered scenarios lead to very interesting and different strategies that can be considered also for other areas of applications.

Besides the gathering problem, the Look-Compute-Move model has been studied also for the problem of *graph exploration with stop* and *exclusive perpetual graph exploration* [4, 5, 6, 12, 13, 14]. In the first problem [12, 13, 14], it is required that each node (or each edge) of the input graph is visited for a finite number of times by at least one robot and, eventually, all the robots have to stop. This implies that after performing the exploration step, the algorithms need some mean to empower the robots by the capability of recording the part of the graph that has been already explored. Since the robots are oblivious, this task is performed by identifying particular configurations of the robots indicating that the exploration task has been accomplished. The exclusive perpetual graph exploration [4, 5, 6] requires that each robot visits each node of the graph infinitely many times. Moreover, it adds the constraint that no two robots should concurrently be on the same node or cross the same edge.

### *1.1.1 Outline*

The chapter is organized in three main sections, dictated by the graph topologies considered. The next section provides techniques and results for the gathering on ring networks. In particular, the section is divided in three parts. First, impossibility results concerning the gathering on rings are summarized. Those hold even though the global-strong multiplicity detection is assumed. Then, results for the case of global-weak multiplicity detection are shown. Under such assumptions, all possible initial gatherable configurations have been addressed. Finally, partial results for the case of local-weak multiplicity detection are described. In Section 1.3, the problem for grids is fully characterized even when no multiplicity detection is assumed. Similarly, in Section 1.4 a full characterization without any multiplicity detection capability is provided for tree topologies. This is indeed an original contribution of the chapter. Finally, Section 1.5 concludes the chapter and outlines some possible research directions for robot-based computing systems.

## 1.2 Gathering on Rings

In this section, the gathering over ring networks is presented. After providing some necessary definitions, impossibility results are summarized when the global-strong multiplicity detection is assumed. Then, differences between the case of global-weak and local-weak multiplicity detection assumptions are presented. In particular, when the global-weak multiplicity detection is assumed, a full characterization of the gatherable configurations is provided. Whereas, when the local-weak multiplicity detection is assumed, only some sub-cases are solved. However, the different techniques used to accomplish the gathering task among the approached scenarios are very interesting for further investigations in robot-based computing systems.

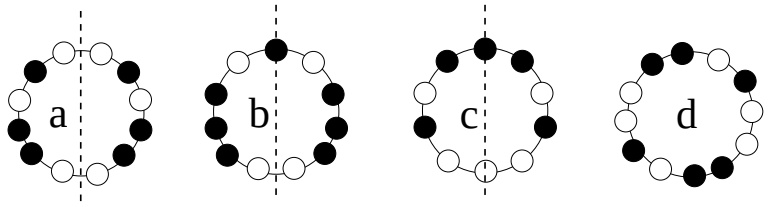


**Fig. 1.1** Symmetric and periodic initial configurations on a ring. Withe nodes are empty while each black node is occupied by one robot.

The model assumes that $k$ robots are placed over the $n$ nodes of a ring, and in the *initial configurations*, nodes are occupied by at most one robot. Depending on the movements imposed by the running algorithms, multiplicities may occur. A configuration is called *symmetric* if the ring admits a geometrical *axis of symmetry*, that defines a bijective function among the robots residing in the two halves of the ring cut by the axis. When the global-weak multiplicity is considered, a configuration is called symmetric if the ring admits a geometrical axis of symmetry that reflects single robots into single robots, multiplicities into multiplicities, and empty nodes into empty nodes. In this case, a configuration might be considered symmetric even though the two halves of the ring cut by the axis do not contain the same number of robots. This can happen if two symmetric multiplicities at the two halves are composed of a different number of robots. If the local-strong (or the local-weak) multiplicity detection is assumed, then a configuration might result symmetric for some robots while asymmetric for others. For instance, if robots are part of a multiplicity and the configuration does not admit an axis of symmetry passing through such a node, then the configuration would result asymmetric for all the robots composing the multiplicity, while it might be symmetric with respect to the perception of all the other ones. However, symmetric peculiarities of initial configurations are invariant with respect to the assumed multiplicity detection, as multiplicities are not allowed at the beginning.

As shown in Figure 1.1, a symmetric configuration with an axis of symmetry has an *edge-edge symmetry* if the axis goes through two edges (Figure 1.1a); it has a *node-edge symmetry* if the axis goes through one node and one edge (Figure 1.1a); it has a *node-node symmetry* if the axis goes through two nodes (Figure 1.1c); it has a *robot-on-axis symmetry* if there is at least one node on the axis of symmetry occupied by a robot (both Figure 1.1b and Figure 1.1c).

A configuration is called *periodic* if it is invariable under non-trivial (i.e., non-complete) rotations (Figure 1.1d).

### *1.2.1 Impossibility results*

In [22], it is proved that the gathering is unsolvable if the multiplicity detection capability is completely removed in either of its forms. When the multiplicity detection is assumed, even in its strong and global form, still there are configurations for which it is impossible to accomplish the gathering task. More precisely, initial configurations composed of only 2 robots, periodic configurations, and those admitting an edge-edge axis of symmetry do not allow to finalize the gathering.

In [21], the case of 4 robots on a ring of five nodes is pointed out as a case of symmetric initial configurations with an even number of robots that does not allow any gathering algorithm. This has been also studied in [16] along with other solvable cases. In general, symmetric configurations of type node-edge with 4 robots and the odd interval cut by the axis bigger than the even one are ungatherable. In the rest of the chapter these configurations are denoted with the set $SP4$. The case of 4 robots on a five nodes ring belongs to $SP4$. Actually, some configurations in $SP4$ could be gatherable but they require strategies that are difficult to generalize.

For all the remaining initial configurations, various gathering algorithms have been provided, depending also on the assumptions concerning the multiplicity detection capability. Whenever clear by the context, we refer to initial configurations simply as configurations.

### *1.2.2 Global-Weak Multiplicity detection*

In this section, a description of the techniques taken from the specific literature are described. Based on the global-weak multiplicity detection capability, the next algorithms cope with all the cases left from the impossibility results previously shown.

#### 1.2.2.1 Asymmetric configurations.

The asymmetric initial configurations have been firstly handled in [22]. When such configurations are aperiodic, they were referred to as *rigid* configurations. The gathering is performed by exploiting the perception of the robots. Perception allows robots to agree and move exactly one robot at time although the model does not allow communication. More precisely, each robot detects which one must perform the next move based on the configuration perceived during the Look phase. This is done until the first (and only) multiplicity occurs. Since the scheduler that wakes the robots up is assumed to be fair, the robot that is allowed to move will eventually wake up and perform all its Look-Compute-Move cycle. This will ensure the robots perform all required moves until the desired multiplicity is created. Once the multiplicity has been created, the robots with only free nodes between themselves and the multiplicity are allowed to move towards the multiplicity, and joining it, until all the robots gather at the same node.

At each step of the proposed strategy, and before creating the multiplicity, the robot allowed to move will be chosen in such a way that the configuration will never lose its original "rigidity". Once captured the current configuration during the Look phase, a robot looks for the pair of robots that are at the maximum distance (in terms of empty nodes in between) from each other. If only one pair of robots is detected, the one allowed to move is the robot with the closest neighbor on the other side of the maximum interval. Possible ties are easily broken by considering the next intervals and so forth, until a difference occurs. Since the configuration is asymmetric, there must be a difference somewhere, and one robot is elected to move. If more than one pair provides the same maximum distance, ties are broken by considering the global view, hence ordering lexicographically the views (in terms of sequences of distances) and choosing the interval that appears as first in the largest view. Once the single robot has been elected to move, it performs the movement that enlarges the maximum interval. This ensures that in the next step there is exactly one maximum distance, with one interval at its side smaller than the one at the other side. Hence, from now on, only the same robot will be allowed to move until creating a multiplicity.

#### 1.2.2.2 Odd number of robots.

Another type of initial configurations addressed and solved in [22] concerns all the configurations with an odd number of robots. In this case, the configuration can be either asymmetric or symmetric. In the former case, the gathering is solved as described in the previous section. In the latter case, it can be observed that one robot resides on the axis of symmetry, necessarily. Then the following property is exploited:

*Property 1.* [22] 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).

When Property 1 holds, symmetric configurations with an odd number of robots will allow only one robot to move until either a multiplicity occurs or the configuration becomes asymmetric and the gathering algorithm changes to that described above.

#### 1.2.2.3 Even number of robots.

The cases left open by the techniques described above are all the symmetric initial configurations with an even number of robots. Note that, configurations with only 2 robots are ungatherable as well as configurations with 4 robots in $SP4$.

A first study that addresses the case of an even number of robots comes from [21]. In that paper, the authors solved all the symmetric cases with an even number of robots grater than 18. When robots are on the axis of symmetry it may be possible to design algorithms which break the symmetry by moving one of the robots located on the axis, as in the case of an odd number of robots described by Property 1.

When no robots reside on the axis, the algorithm works in four phases. During the first phase, since the configuration is symmetric, two robots are always allowed to move. In order to detect the two symmetric robots that must perform their moves, the robots have to elaborate the perceived configuration during their Compute phase. Based on the sequence of distances between robots along the ring, two symmetric minimal interval are detected and reduced concurrently, until two multiplicities are created. The number of robots grater than 18 comes from this computational step. In fact, the need to guarantee to break possible ties among minimal intervals, and the fact that some robots are needed between the detected intervals and the two poles defined by the axis of symmetry on the ring, gives a minimal number of required robots equal to 20.

It is very important to remark that the proposed technique is the first one that forces robots to maintain the original symmetry rather than breaking it. In fact, based on the perceived configurations, robots are always able to detect whether the current configuration is at one step from a reachable symmetry or not. In the latter case, the algorithm from [22] for asymmetric configurations can be applied. In the former case, the robot that can re-establish the symmetry will be the only one allowed to move. Note that, such a robot could have been already started its Look-Compute-Move cycle concurrently with its symmetric one, or it simply starts later. In any case, the algorithm guarantees to recover the original symmetry with two steps less towards the desired configuration with two multiplicities where the second phase starts.

When two multiplicities have been created, the idea is to move all the remaining single robots but few of them towards the two multiplicities. During the second phase, it is necessary to decide on one of the two poles of the axis of symmetry as the gathering point (the North pole). The poles are chosen so that the northern arc between multiplicities contains more robots than the southern arc; in the case of a tie, the side on which the nearest robots are closer to the multiplicities is the northern one. The robots are moved in symmetrical pairs towards their respective multiplicities, starting from the robots on the northern arc. In this way, North and South are consistently preserved throughout the phase. The phase ends with two multiplicities, two symmetric robots located at the southern part far from the multiplicities of at least one node, and two symmetric robots located at the northern part neighbors of the multiplicities. The two robots on the south are called *guards*.

The third phase is based on the position of the guards that maintain the direction to the gathering node. During this phase, the remaining single robots and those belonging to the multiplicities can move towards the North pole. The movement is performed always maintaining the robots associated to each multiplicity either as part or as neighbors of it. In this way, the configuration pattern is maintained

throughout the process, until all robots except for the guards gather at the North pole in a single multiplicity.

The fourth phase simply moves the guards towards the multiplicity, and the gathering will be eventually finalized.

The algorithm has been also integrated with the one from [22], hence obtaining a full characterization of the gatherable configurations with an odd number of robots, with an even number of robots but asymmetric, with an even number of robots admitting a robot-robot axis of symmetry, and with more than 18 robots admitting a node on the axis of symmetry. This has left open the cases of an even number of robots between 4 and 18 admitting a node on the axis of symmetry. Note that, the cases left for few robots might require more effort and different techniques for the resolution. In fact, lesser the robots lesser the information encoded by their disposal. This encouraged further investigation on configurations with few robots.

Gatherable configurations with 4 robots have been addressed in [16, 23]. The main idea is still to define a North and a South pole on the axis of symmetry (of type node-node). Then similarly to [21], the two northern nodes are moved while preserving the symmetry until creating a multiplicity on the North pole. After that, the other two robots join the multiplicity, hence finalizing the gathering.

The case of 6 robots is more intriguing as it requires different techniques from the older ones in order to fully characterize the gatherable configurations. It has been addressed in [9]. A symmetric configuration can be represented as shown in Figure 1.2. In detail, without multiplicities, the ring is divided by the robots into 6 intervals: $A$, $B$, $C$, $B'$, $C'$, and $D$ with $a$, $b$, $c$, $b$, $c$, and $d$ free nodes, respectively. In the case of node-edge symmetry, $A$ is the interval where the axis passes through a node and $D$ is the interval where the axis passes through an edge; in the case of node-node symmetry, $A$ and $D$ are the intervals such that either $a < d$ or $a = d$ and $b < c$; the case where $a = d$ and $b = c$ cannot occur as it generates two axis of symmetry. Note that, in the case of node-node symmetry, $a$ and $d$ are both odd, while, in the case of node-edge symmetry, $a$ is odd and $d$ is even. Robots between $A$ and $B$ ($B'$, respectively) are denoted by $x$ ($x'$, respectively); those between $B$ and $C$ ($B'$ and $C'$, respectively) are $y$ ($y'$, respectively); those between $C$ ($C'$, respectively) and $D$ are $z$ ($z'$, respectively), see Figure 1.2.

A robot $r \in \{x, y, z, x', y', z'\}$ can perform only two moves: it moves *up* ($r\uparrow$) if it goes towards $A$; it moves *down* ($r\downarrow$) if it goes towards $D$.

The main idea of the algorithm is to perform moves $x\uparrow$, $x'\uparrow$, $y\uparrow$ and $y'\uparrow$, with the aim of preserving the symmetry and gathering in the middle node of interval $A$, where the axis is directed. In some special cases, it may happen that the axis of symmetry changes at run time. Before multiplicities are created, the algorithm in a symmetric configuration allows only two robots to move in order to create a new symmetric configuration.

In the general case, the algorithm compares $b$ and $d$, and performs a pair of moves such as when $b > d$, then $b$ is enlarged, while, if $b < d$, then $b$ is reduced. In this way, the axis of symmetry and its direction do not change.

Apart from some special cases, the algorithm works as follows. When $b > d$, $x\uparrow$ and $x'\uparrow$ are performed, while, when $b < d$, $y\uparrow$ and $y'\uparrow$ are performed. In both cases,

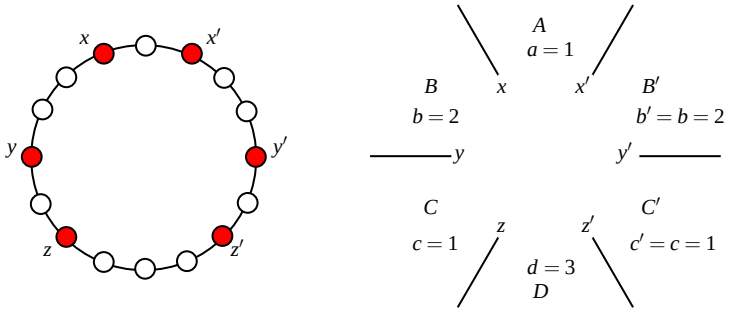


**Fig. 1.2** A symmetric configuration and its representation.

(apart for some special cases) the ordering between $b$ and $d$ is maintained in the new configuration. Eventually, either one multiplicity is created at the middle node of the original interval $A$ by means of robots $x$ and $x'$, or two symmetric multiplicities are created on the positions originally occupied by $x$ and $x'$ by means of the moves of $y$ and $y'$, respectively. In the second case, the two multiplicities will move up again to the middle node of the original interval $A$ by allowing at most 4 robots to move all together. Once such a multiplicity has been created, the remaining robots join it, and conclude the gathering. In the special case of $b = d$, which can only happen in the initial configuration, the algorithm tries to break this equality by enlarging or reducing $d$ by means of either $z\uparrow$ and $z'\uparrow$ (when $C > 0$) or $z\downarrow$ and $z'\downarrow$ (when $C = 0$ and $D > 0$). The special cases when $C = D = 0$ require specific arguments that can be found in [9].

#### 1.2.2.4 Unifying algorithm.

Recently in [11], a new technique has been proposed for addressing all the gatherable initial configurations by means of a single algorithm that exploits some of the described strategies while also solving the remaining cases left open. In particular, existing algorithms are used as subroutines for solving the basic gatherable cases with 4 or 6 robots from [23] and [9], respectively. Also, Property 1 is exploited in some cases. Then, the main strategy is based on the definition of a particular read of the configurations perceived by the robots during their Look phase.

The current configuration of the system can be described in terms of the view of a robot $r$ that is performing the Look operation. A configuration seen by $r$ is represented 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 by robots when traversing the ring in one direction, starting from $r$. Unless differently specified, $Q(r)$ represents the configuration providing the lexicographical minimum among the two possible views. For instance, in the configuration of Figure 1.2a, robot $x$ can read either $Q =$

$(1,2,1,3,1,2)$ or $Q' = (2,1,3,1,2,1)$, hence $Q(x) = Q$. A multiplicity is represented as $q_i = -1$ for some $0 \leq i \leq j$, regardless the number of robots composing it.

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 $q_i$ as first interval, 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 the gathering algorithm.

**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 $I_C = \{q_i \mid C_i = C^{SM} \text{ or } \overline{(C_i)} = C^{SM}, 0 \leq i \leq j\}$.

Once a robot is able to distinguish where a supermin is located, the next lemma provides a useful mean for computing whether the current configuration is gatherable or not.

**Lemma 1.** *[11] 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 $\frac{n}{3}$.*

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, all the remaining configurations are shown to be gatherable.

The main strategy allows only the movements which 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. This strategy reminds the one used in [22] but with the difference to deal with the minimum rather than with the maximum.

For gatherable configurations with $|I_C| = 2$, the 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 one wants 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 after the initial movement, it is possible to obtain a symmetric configuration or an asymmetric one 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, it reduces 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 eventually brings the configuration to have two symmetric multiplicities as above. 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. This strategy reminds the one used in [21] as it tries to preserve the symmetry until the guards can join all the other robots in the gathering node.

Actually, sometimes the strategy that affects only the supermin cannot be applied, as a move may produce some undesired "side-effects", i.e., leading the configuration to ungatherable cases. In order to cope with such cases, two other moves have been defined. However, it can be shown that a robot is always able to understand the correct move to be performed.

An alternative move is to try to reduce the second supermin, i.e., the supermin of the configuration is evaluated after the real one. Another move, called XN, is applied when specific configurations occur. The definition of XN and the description of the cases where it must be applied are not provided in this chapter.

The algorithm works in 5 phases and depends on the configuration perceived by the robots, see Figure 1.3. 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

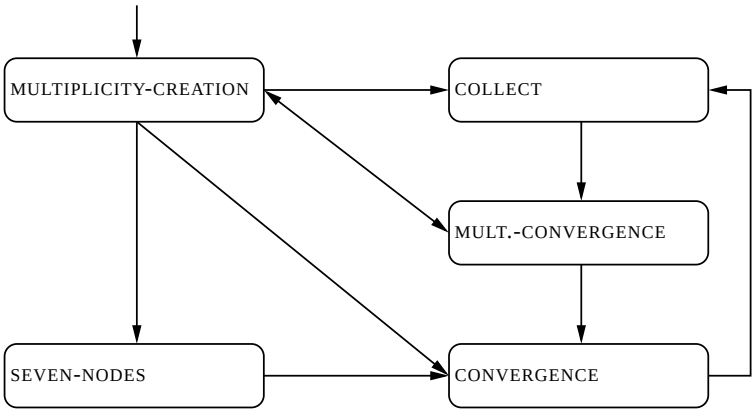


**Fig. 1.3** Phases interchanges.

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 cases of 6 robots on a seven nodes ring are considered separately in phase SEVEN-NODES.

In each phase, the robots can distinguish the type of configuration and apply the suitable strategy/move. The way a robot can identify the type of configuration is based on basic and simple calculations. Given a configuration $C = (q_0, q_1, \ldots, q_j)$, a robot compute 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, OCCUPIED$(C)$;
4. distance between single robots and multiplicities;
5. if $C$ is symmetric;
6. 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 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 checking whether $C = \overline{C_i}$ for some $0 \leq i \leq j$.

To understand when $C$ is at one move from a symmetry allowed by the algorithm, it is sufficient to simulate such a move backwards and checking whether the obtained configuration is symmetric.

Based on the perceived configuration, and once calculated the above parameters, a robot is able to answer to basic questions that check the accomplishment of the gathering task. In particular, a robot can distinguish if the current configuration is gatherable, which type of configuration it perceived, which strategy/move should be applied, if it is allowed to move and towards which direction.

The algorithm solves all the gatherable cases, hence closing also the ones left open by other strategies. However, different assumptions on the model may constitute very interesting directions for further investigations.

### *1.2.3 Local-Weak Multiplicity detection*

Using the local-weak multiplicity detection capability, not all the cases has been addressed so far. In [17], it has been proposed an algorithm for the case of rigid initial configurations where the number of robots $k$ is strictly smaller than $\lfloor \frac{n}{2} \rfloor$. In [18], the case where $k$ is odd and strictly smaller than $n-3$ has been solved. In [19], the authors provide and algorithm for the case where $n$ is odd, $k$ is even, and $10 \leq k \leq n-5$. The remaining cases are still open and a unified algorithm like that for the case where the global-weak multiplicity detection capability is allowed is still not known. In the following, the mentioned algorithms are summarized.

#### 1.2.3.1 Asymmetric configurations with $k < \lfloor \frac{n}{2} \rfloor$.

This algorithm assumes, without loss of generality, that a configuration view seen by a robot is the lexicographically maximal that the robot can read, instead of the lexicographically minimal as it was in the case of global-weak multiplicity detection. These two assumptions are equivalent thus in the rest of the chapter we keep on using the one in [17]. As the configuration is asymmetric, by Property 2, the views seen by the robots are all different. Therefore, let $C = (q_0, q_1, \ldots, q_j)$ be the lexicographically maximal configuration view, $j \leq k$, and $r_i$ be the robot (or the set of robots in the case of a multiplicity) before the interval $q_i$ of empty nodes. First, an algorithm to achieve the gathering for the case where $q_0 \geq 3$ and $q_1 \geq 2$ is given. Then, a strategy to create a configuration of the above type starting from a configuration where $q_0 \geq 3$ and $q_1 < 2$ is devised. Finally, the case to increase $q_0$ from 2 to 3 is addressed. As it is assumed that $k < \lfloor \frac{n}{2} \rfloor$ and $q_0$ is the maximal interval, then $q_0$ cannot be smaller than 2. All the three algorithms keep the configuration asymmetric and aperiodic. Here, the algorithm for the case where $q_0 \geq 3$ and $q_1 \geq 2$ is described, while the details for the other cases can be found in [17].

The idea is to generate a configuration with only two occupied nodes where $k-1$ robots are gathered on the same node and the other occupied node contains a single robot. From this configuration the robots can distinguish which is the node occupied by a single robot by using the local-weak multiplicity detection. Therefore, the single robot moves towards the multiplicity, eventually achieving the gathering.

The algorithm when at least three nodes are occupied ($j \geq 2$) is as follows.

R1: If $q_j \geq 1$ move $r_0$ towards $q_j$;
R2: If $j \neq 2$, $q_j = 0$, and $q_{j-1} \geq 1$

R2-1: If $q_0$ is the only maximum interval of empty nodes, move $r_j$ towards $q_{j-1}$;
R2-2: Otherwise move $r_1$ towards $q_1$;

R3: If $j \neq 2$, $q_j = 0$, and $q_{j-1} = 0$ move $r_0$ towards $q_j$;
R4: If $j = 2$ and $q_2 = 0$ move $r_0$ towards $q_2$;

First, it is assumed that $q_0$ is the only maximum interval of empty nodes. The algorithm allows moves where $q_0$ is increased, $q_1$ is not changed, $q_j$ is kept shorter than $q_1$, and the other intervals are decreased. This ensures that $q_0$ remains the only maximum interval of empty nodes. The algorithm starts by moving the robots $r_0$ towards $q_j$ until they become neighbors of robots in $r_j$ (see rules **R1** and **R3**). Then robots in $r_j$ move towards $q_{j-1}$ until they become neighbors of robots in $r_{j-1}$ (see rule **R2-1**). At this point the robots in $r_0$ join those in $r_j$. By applying these rules, eventually a configuration with only three nodes occupied is achieved where $j = 2$ and $q_2 = 0$. In this case, all the robots in $r_0$ join those in $r_2$ (see rule **R4**) achieving a configuration where $k-1$ robots are gathered on the same node. Finally, the single robot joins the other ones. In the case that $q_0$ is not the only maximum interval of empty nodes, the algorithm moves $r_1$ towards $q_1$, enlarging $q_0$ (see rule **R2-2**). After this moment, $q_0$ is the only maximum interval of empty nodes. It can occur that $q_1$ becomes smaller than 2 or the maximal configuration view is reversed. For instance this can happen when $q_1 = q_j$. In the first case, the algorithm for $q_0 \geq 3$ and $q_1 < 2$ is applied and in the second case the algorithm is applied on the new maximal configuration view.

#### 1.2.3.2 Configurations with an odd number of robots.

The description of the algorithm requires some definitions and terminology. Let $d$ be the size of the minimum interval of empty nodes plus one, a *d-block* is a maximal path where there is exactly one occupied node every $d$ edges. The size of a $d$-block is the number of robots that it contains.

The algorithm works in two phases. The first phase builds a configuration made of a single 1-block, and the second phase achieves gathering.

In the first phase, the robots move towards the $d$-blocks with the biggest size. By using the hypothesis of an odd number of robots, the $d$-blocks of biggest size can merge together in order to create a unique $d$-block. The obtained configuration is symmetric and, as the number of robots is odd, there is a robot on the axis of symmetry. The algorithm proceeds by moving the two robots adjacent to that on the axis of symmetry towards it, achieving a $(d-1)$-block of size 2 or 3. The algorithm is then iterated until a single 1-block is achieved.

The second phase starts with a configuration with a single 1-block. Note that this configuration is symmetric and has a robot $r$ on the axis of symmetry. The algorithm moves the two robots adjacent to $r$ towards it, creating a multiplicity (see Figure 1.4a). If the two robots move synchronously, the configuration achieved is still symmetric and a multiplicity containing $r$ is created on the axis of symmetry

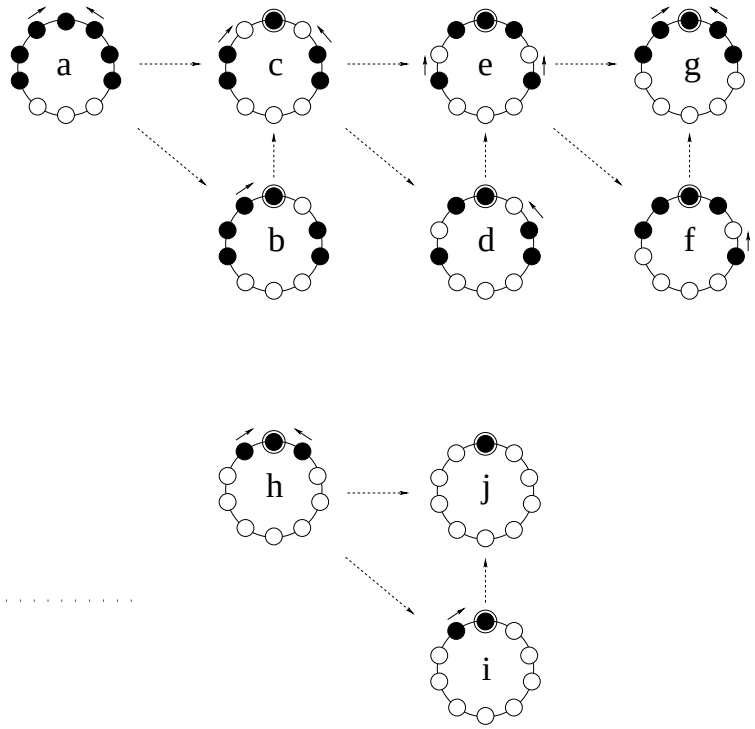


**Fig. 1.4** Second phase of the gathering algorithm for an odd number of robots with local-weak multiplicity detection capability. A multiplicity is denotes as a circle around an occupied node.

(see Figure 1.4c). Due to the asynchronicity, only one of the two robots adjacent to $r$ can move, creating a configuration made of two 1-blocks separated by an empty node (see Figure 1.4b). Such configuration can be distinguished by observing that this is the only case where the number of occupied nodes is even. Hence the robot which did not move can easily identify itself and perform the correct move. In the obtained configuration, the neighbors of the multiplicity on the axis are two empty nodes followed by two 1-blocks. At this point, the algorithm moves the two robots on the border of the 1-blocks that are closest to the multiplicity, towards it. For the hypothesis of asynchornicity, it can occur that three 1-blocks are created (see Figure 1.4d). In this case, the robot that has to move can be identified thanks to the hypothesis that the number of occupied nodes is always smaller that $n-3$. In fact, in these configurations, the 1-blocks are interleaved by 2 single empty nodes and by a path of empty nodes of size at least three. By iterating this process, a new 1-block is created with the size reduced by 2 with respect to the original 1-block and where there is a multiplicity on the axis of symmetry (see Figure 1.4g). The algorithm is then iterated until the gathering is achieved. In the final step of the algorithm a single 1-block of size 2 can be created as a consequence of asynchronicity (see Figure 1.4i). At this point the algorithm exploits the local-weak multiplicity detection capability

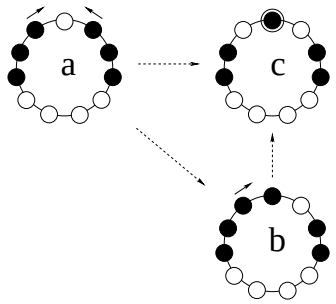


**Fig. 1.5** Third phase of the gathering algorithm for an even number of robots in an odd ring with local-weak multiplicity detection capability. A multiplicity is denotes as a circle around an occupied node.

and moves the robot which is not on the multiplicity towards the other occupied node.

#### 1.2.3.3 Configurations with an even number of robots on an odd size ring.

In this case, the algorithm is divided into three phases. The first two phases aim at creating a *terminal* configuration, i.e., a configuration made of only two 1-blocks of size $\frac{k}{2}$ which are separated by exactly one empty node. Finally, the third phase finalizes the gathering.

The first phase starts from any allowed configuration and creates a configuration with either a single 1-block or two 1-blocks of size $\frac{k}{2}$. The idea is similar to that of the case of odd robots. First, only $d$-blocks are created by moving all the isolated blocks towards the $d$-blocks until joining them. Then, the robots move with the aim of creating a unique $d$-block. When all the robots belong to the same $d$-block, some robots move in order to decrease the $d$ and repeat the algorithm until $d = 1$. The correctness of the algorithm relies on the fact that the number of nodes in the ring is odd and the number of robots is even. This implies that if the configuration is symmetric, then the axis of symmetry passes through exactly one empty node and one edge. In the second phase, the algorithm moves the two 1-blocks towards the empty node crossed by the axis of symmetry until a terminal configuration is created. In the case that the first phase ends with a single 1-block, this is split into two 1-blocks. All these movements are done by preserving the symmetry of the configuration. The third phase achieves the gathering from terminal configuration by moving the two robots that are on the border of the two 1-blocks and that are neighbors of a single empty node. These two robots move towards the single empty node. For an example, see Figure 1.5. When these two robots are moved one of the following cases (depending on the activation schedule) can occur:

- The two robots move synchronously and create a symmetric configuration with a multiplicity crossed by the axis of symmetry (see Figure 1.5c);

- Only one robot moves and creates a configuration with two 1-blocks separated by a single empty node and whose sizes differ by 2 (see Figure 1.5b). This configuration is easy to recognize and hence the pending move can be performed, achieving again a symmetric configuration with a multiplicity crossed by the axis of symmetry (see Figure 1.5c).

At this point, the robots on the multiplicity are not allowed to move. The other robots see an odd number of robots and perform the last phase of the algorithm for an odd number of robots (see Figures 1.4c–1.4j), so that the gathering is eventually achieved. This implies that the maximum number $k$ of allowed robots has to satisfy $k-1 < n-3$, that is $k \leq n-5$.

## 1.3 Gathering on Grids

In this section, results achieved in [10] are reported. The authors consider the gathering problem on an anonymous and undirected grid of $n \times m$ nodes, with $m \geq n$. The main assumption that distinguish these results from those obtained on rings is the lack of any multiplicity detection capability: if a node is occupied by more than one robot, it is not perceived by the robots, even if they reside on such a node.

Initially, each node is occupied by at most one robot. During a Look operation, a robot perceives the relative locations on the grid of occupied nodes, regardless of the number of robots at a node.

The current configuration of the system can be described in terms of the view of a robot $r$ which is performing the Look operation at the current moment. A configuration seen by $r$ is denoted as an $n \times m$ matrix $M$ that has elements belonging ot the set $\{0,1\}$. Value 0 represents an empty node, and 1 represents an occupied node.

Since the grid is anonymous and undirected, each robot can perceive the current configuration with respect to different rotations and reflections leading to any view of the grid satisfying the $n \times m$ dimension. In particular, when $n = m$ each of the 4 rotations and 4 reflections provides a feasible view.

**Definition 2.** A configuration is *periodic* if it is invariant with respect to rotations of 90 or 180 degrees, where the rotation point coincides with the geometric center of the grid.

**Definition 3.** A configuration is *symmetric* if it is invariant after a reflection with respect to a vertical, horizontal, or diagonal (in case of square grids) axis passing through the geometric center of the grid.

### *1.3.1 Odd×odd grids*

This case is trivially solvable, in fact in odd×odd grids, a robot can always detect, during its Look operation, the central node of the grid $M[\lceil \frac{n}{2} \rceil, \lceil \frac{m}{2} \rceil]$, regardless of

its possible view. This means that all the robots can move toward the center, concurrently.

### *1.3.2 Odd×even grids*

In this case, the gathering is not always feasible. In fact, similarly to the ring case on periodic or symmetric configurations of type edge-edge [22], if a configuration $C$ is periodic, or symmetric with respect to an axis passing through the edges (i.e., dividing the grid into two halves from the even side), then $C$ is ungatherable.

When the starting configuration does not belong to the above ungatherable configurations, it always possible to devise an algorithm achieving the gathering without multiplicity detection.

The idea is to distinguish between the two nodes that are the central nodes of the odd borders of the grid. If $m$ ($n$, respectively) is odd, then the two mentioned nodes are given by positions $M[1, \lceil \frac{m}{2} \rceil]$ and $M[n, \lceil \frac{m}{2} \rceil]$ ($M[\lceil \frac{n}{2} \rceil, 1]$ and $M[\lceil \frac{n}{2} \rceil, m]$, respectively). The line connecting those two nodes will be denoted as the NS line. One of the two extreme nodes on the NS line will be the place where the gathering is finalized. In order to select the gathering node, a robot considers the line passing through the central edges of the even sides of the grid (denoted as the EW line) dividing the grid into two halves. The idea is to distinguish a north and a south part among the two halves and the gathering node will be the one in the north half. See Figure 1.6a for a visualization. The north is the half with more nodes occupied by robots, if any. If the number of occupied nodes in the two halves is the same, then some more computations are required. In both cases, the robots move from the south to the north until all the robots will be in the north part. Note that, during such a stage, if multiplicities are created in the south, then the number of occupied nodes decreases with respect to the north part. If multiplicities are created in the north, it means that a robot has moved from the south to the north part, still preserving the required distinction.

In order to distinguish the north from the south in the case of configurations with the same number of robots among the two halves obtained by the EW line, a robot associates to each configuration $C$ a binary string as follows. Starting from each corner of the grid, and proceeding in the direction parallel to the NS line, a robot records the elements of $M$ row by row, or column by column (according to the direction specified by the NS line). Once it has computed the four strings, it associates to $C$ the lexicographically largest one. For instance, starting from corner $M[1,1]$, and assuming $m$ odd, the corresponding binary string would be composed by the sequence $M[1,1]$, $M[2,1]$, ..., $M[n,1]$, $M[1,2]$, ..., $M[n,2]$, $M[1,m]$, ..., $M[n,m]$. See Figure 1.6a for an example.

It is possible to show that if $C$ is a gatherable configuration, then, among the four possible strings coming from a robot view of the input grid, at most two strings can be the lexicographically largest ones. If there are two largest strings, then they represent the views of $C$ starting from two symmetric corners with respect to the NS

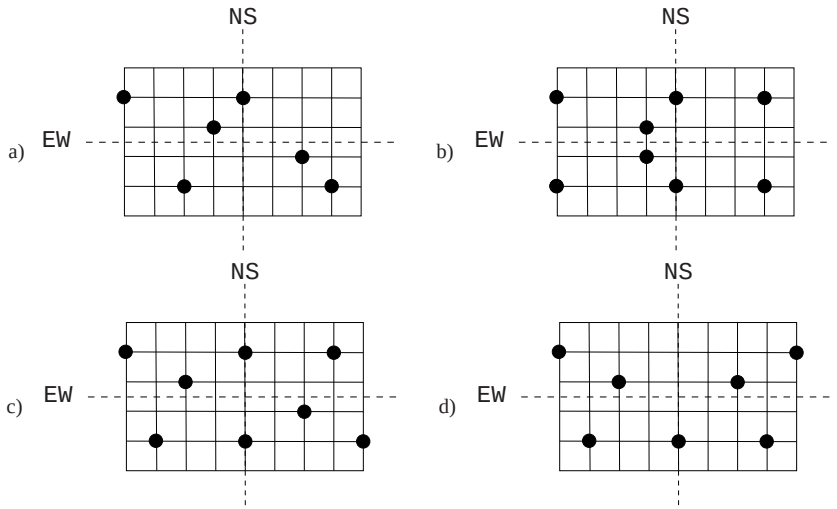


**Fig. 1.6** Case of a $6 \times 9$ grid. a) The two halves have the same number of nodes. The maximal binary string is that read from the upper left corner which starts with $(0,1,0,0,0,0,\ldots)$. b),c),d) Examples of $6 \times 9$ grids with two lexicographically largest strings: b) The two lexicographically largest strings correspond to the views starting from two symmetric corners with respect to the EW line; c) The two lexicographically largest strings correspond to the views starting from two corners residing on one of the two diagonals of the grid; d) The two lexicographically largest strings correspond to the views starting from two symmetric corners with respect to the NS line.

line, see Figure 1.6d. Note that, instances of Figures 1.6b and 1.6c are ungatherable as they admit an edge-edge symmetry or a periodicity.

Then, the *gathering node* is defined as the one residing on the same odd side where the corner(s) providing the lexicographically largest string resides. The gathering node will determine also the directions along the NS line: the gathering node is called the north pole.

Configurations on odd$\times$even grids that are aperiodic and do not admit an axis of symmetry passing through edges are always gatherable, and the algorithm is the following.

Once the gathering node has been unambiguously identified by a robot during its Compute operation, if the robot resides on the half grid where the south pole is, then it moves towards the north pole. Note that, each time a robot in the southern half of the grid performs such a movement, the gathering node cannot change. In fact, two following two cases can occur: 1) the number of occupied nodes decreases in the southern part of the grid, either because a robot moves to the northern part or because a multiplicity is created; 2) the string associated to the corners in the south are decreasing due to the robots' movements. In this case, the corresponding strings defining the current configuration starting from the northern corners are increasing. This clearly leaves unchanged the direction on the NS line. Note that the corner to which the lexicographically largest string was associated might change during the described process, but the only option is the other corner on the same odd side of

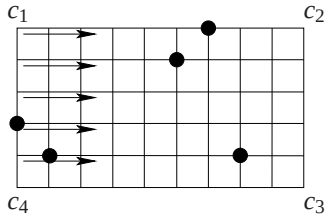


**Fig. 1.7** Case of a $6 \times 10$ grid. The arrows indicate the horizontal direction of the reading from corner $c_1$, it gives $(6,8,14,10,5,12)$. The other seven sequences read by the robots are: $(3,6,20,4,9,13)$ from $c_1$ vertically, $(3,10,24,2,5,11)$ and $(16,1,6,26,4,2)$ from $c_2$ horizontally and vertically, respectively, $(12,5,10,14,8,6)$ and $(13,9,4,20,6,3)$ from $c_3$, $(11,5,2,24,10,3)$ and $(2,4,26,6,1,16)$ from $c_4$. The *minimal* sequence is $(2,4,26,6,1,16)$ and $c = c_4$.

the original one. This preserves the direction on the NS line. By keeping on moving in the described way, all the robots will reach the northern part. The case in which a subset of robots from a multiplicity move, increasing the number of occupied nodes, does not require any special argument. More precisely, since the initial configuration does not contain multiplicities, either the minimality of the number of robots in one half of the grid is preserved, or case 2) ensures that the lexicographically largest string is associated to a corner in the north.

Once all the robots belong to one half of the grid, then they are allowed to move, during their Move operation, towards the gathering node. In fact, such a node is well-defined and cannot change as the robots are not allowed to move to the other half of the grid.

### *1.3.3 Even×even grids*

In this section, the case of grids whose sides are both even is studied. Also in this case, there are some configurations which are ungatherable, namely the periodic configurations and those configurations having a vertical or a horizontal axis of symmetry. In [10], it is shown that all the other cases are gatherable without any multiplicity detection, but for the case of $2\times2$ grids.

On $2\times2$ grids, configurations with two or four nodes occupied are ungatherable due to periodicity and edge-edge symmetries. If three nodes are occupied with robots having the local-weak multiplicity detection, the configuration is gatherable by moving the robot in between the other two occupied nodes arbitrarily, and then moving the robot not in the multiplicity towards the other occupied node. Hence, the remaining gatherable configurations are the aperiodic, asymmetric, and those with only one axis of symmetry passing through the diagonal of a square grid of dimensions larger than $2\times2$. All such configurations are referred to as the set EG (Even-Gatherable) and it is proved that all the configurations in EG are indeed gatherable without any multiplicity detection. In order to achieve this results, it is first assumed

that at least one node on the border of the grid is occupied. Then, the gathering node is identified among the eight sequences of distances (number of empty nodes) between occupied nodes obtained by traversing the grid starting from the four corners and proceeding towards the two possible directions (see, e.g. Figure 1.7).

The lexicographically smallest sequence between the two readings from any corner is associated to the corner itself. In rectangular grids, these two sequences can be equal but it is possible to distinguish one of them by assuming the reading in the direction of the smallest side.

The *minimal* sequence is defined as follows. If the configuration is symmetric, it is the smallest sequence between the two sequences associated to the two corners through which passes the axis of symmetry, otherwise it is the smallest among the four sequences associated to the four corners. In any case there exists a minimal sequence $C = (q_0, q_1, \ldots, q_j)$ which identifies a single corner $c$, unambiguously.

An important property of the gathering strategy is that a robot is not allowed to move to a corner different from $c$.

First it is proven that for any EG configuration with no corners occupied and at least one robot on the border there exists a strategy that leads to a configuration with exactly one corner occupied. The idea is to reduce $q_0$ by moving the robot towards $c$ (or the two robots, when the configuration is symmetric) on the border which is (are) closest to $c$. The authors show that no symmetric configuration, other than the possible original one, can be created.

Then, it is shown that for any configuration in EG with more than three nodes occupied and at least two corners occupied there exists a strategy that leads to a configuration with either exactly one corner occupied or exactly three corners occupied.

Finally the main contribution is proven: Aperiodic configurations on even$\times$even grids larger than $2\times2$, that do not admit an axis of symmetry passing through edges, are gatherable, without assuming any multiplicity detection.

To achieve this result, it has been first observed that the set of possible grids can be restricted by considering the minimal even$\times$even sub-grid which is centered in the geometrical middle of the original grid and includes all the occupied nodes of it. Such minimal *wrapping* grid is still of type even$\times$even and preserves the possible symmetry of the original one. Moreover, it always has at least an occupied node on the border. Then it is possible to apply the first partial result mentioned above.

The proposed algorithm only uses such sub-grid without changing its size, i.e., it neither enlarges nor reduces the sub-grid by moving robots outside the border or from the border to the inside.

If no corners are occupied, by reducing $q_0$, a configuration with one corner occupied can be reached. In this case, all the robots move towards $c$ by reducing the Manhattan distance to $c$ and then achieving the gathering.

When two corners are occupied, as said above, it is possible to reach a configuration with one or three corners occupied. In the former case the gathering can be easily finalized, in the latter case all the robots, but those in the corners, are moved towards the corner that does not share any coordinate with the empty corner. This process finishes with a symmetric configuration with exactly three corners occu-

pied. In this configuration, $c$ is the corner on the axis of symmetry, and the other two robots move one step towards $c$ either concurrently or alternately, until creating a configuration with only one corner occupied.

If four corners are occupied, the robot which occupies the corner farthest from $c$ is moved in an arbitrary direction, generating a configuration where only three corners are occupied.

It remains the case where the minimal wrapping even$\times$even sub-grid which includes all the occupied nodes of the original grid has dimension $2\times2$. The configuration is ungatherable on this sub-grid without multiplicity detection. However, in the case of exactly three nodes occupied, it is possible to exploit the larger dimensions of the original grid in order to avoid the multiplicity detection. The cases of two or four nodes occupied clearly remain ungatherable. The strategy is then to move the robot on the corner of the $2\times2$ grid which is in between the other two occupied corners towards the external row or column, arbitrarily. The case where the minimal wrapping grid has dimension $4\times4$ is obtained and no corners are occupied.

## 1.4 Gathering on Trees

In this section, gathering results on trees are presented. To the best of our knowledge, these are original contributions as trees were never treated before under the considered model. Given a tree, a node at minimal distance from all the other ones is called *center*. Based on well-known results [28] about the tree topology, within a tree there is either one center or there are two neighboring centers. In the former case, no matter the initial distribution of the robots, each of them can move towards the center, concurrently. The gathering will be eventually finalized, even without any multiplicity detection assumption. In the latter case, some more specific arguments are required. In fact, some impossibility results hold.

**Lemma 2.** *If the two subtrees rooted at the centers along with the disposal of the robots are isomorphic, then the gathering is impossible.*

*Proof.* Any algorithm designed to accomplish the gathering on the tree must work regardless the delays on the decisions made by robots. In particular, also the synchronous case must be solved. Since the two considered subtrees are isomorphic, with the same disposal of robots, if one robot is allowed to move within one subtree, there must exist another robot that is allowed to accomplish the same specular movement. If both robots perform such movements, again a configuration with two isomorphic subtrees is obtained. In proceeding so, there will not exist any move that can break such a situation. Hence, the gathering cannot be finalized as it requires to distinguish one single node belonging to one of the two subtrees.

In the case the isomorphism among the two subtrees along with the disposal of the robots does not hold, the following strategy can be applied. Let $c_1$ and $c_2$ be the two centers of the input tree $T$, and let $T_1$ and $T_2$ be the two subtrees rooted

at $c_1$ and $c_2$, respectively, when the edge connecting $c_1$ and $c_2$ is removed. If the number of nodes occupied in $T_1$ is smaller than that in $T_2$, then all robots in $T_1$ are moved towards $c_2$. Once $T_1$ gets empty, all robots in $T_2$ should be moved towards $c_2$ in order to end the gathering. If the number of nodes occupied in $T_1$ is equal to that in $T_2$, it is always possible to determine which subtree is less than the other with respect to a natural ordering on labeled trees (see [1, 7]). To define the smaller tree as the one with the robots closer to the root, we associate label 1 to empty nodes, and label 0 to nodes occupied by robots. Then the algorithm would exploit this ordering in order to detect the robots to move from one subtree towards the root of the other one. If a robot moves over a node already occupied, the number of occupied nodes in the original subtree decreases. As soon as one robot moves towards the other subtree, the number of robots in the two subtrees is no longer equal and the previous strategy can be applied. Similarly to what happened in the case of even$\times$odd grids of Section 1.3.2, the occurrence of multiplicities does not affect the proposed algorithm.

## 1.5 Conclusion

In this chapter, we surveyed recent results about the gathering problem under the Look-Compute-Move model in various graph topologies.

For most of the cases under investigation, it turned out that the problem has been fully characterized. For trees and grids, the multiplicity detection capability does not strengthen the model, that is, all the cases which admit gathering are still solvable without such a capability. The only exception is provided by the very specific case of 3 robots on a $2\times2$ grid. However, the multiplicity detection capability can strengthen the model in the case of ring topologies. In the literature, the case that assumes global-weak multiplicity detection has been fully characterized while that assuming local-weak multiplicity detection still lacks of a unified algorithm, and there are some open cases that deserve further investigations.

The study of different topologies has required very different and sometimes opposite approaches that stimulate main advances in robot-based computing systems. On the other hand, some of the techniques described for different topologies share common ideas that can be, therefore, used in other topologies. Infinite grids, tori, and hypercubes might represent just a sampling set.

Another challenging direction would be that of investigating the minimum number of steps required by the robots to accomplish the gathering task. So far, the research has mainly focused on the feasibility of the gathering, while few results concern the minimization of the robots' movements. Similarly, low effort has been spent in order to increase the opportunity to parallelize movements. As we have seen for ring networks, at most two robots are allowed to move concurrently unless robots composing multiplicities must move. Whereas, on grids and trees, less restrictions are imposed on the robots' movements.

It would be interesting to investigate how the proposed techniques may affect the resolution of different tasks, as well as how different assumptions on the capability of the robots may change the required strategies.

## References


1. Aho, A., Hopcroft, J., Ullman, J.: Data Structures and Algorithms. Addison Wesley (1983)
2. Alpern, S.: Rendezvous search: A personal perspective. Operations Research **50**(5), 772–795 (2002)
3. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous, *International Series in Operations Research & Management*, vol. 55. Springer (2003)
4. Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: On the solvability of anonymous partial grids exploration by mobile robots. In: Proceedings of the 12th International Conference On Principles Of Distributed Systems (OPODIS), *Lecture Notes in Computer Science*, vol. 5401, pp. 428–445. Springer-Verlag (2008)
5. Blin, L., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Exclusive perpetual ring exploration without chirality. In: Proceedings of the 24th International Symposium on Distributed Computing (DISC), *Lecture Notes in Computer Science*, vol. 6343, pp. 312–327. Springer (2010)
6. Bonnet, F., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Asynchronous exclusive perpetual grid exploration without sense of direction. In: Proceedings of the 15th International Conference On Principles Of Distributed Systems (OPODIS), *Lecture Notes in Computer Science*, vol. 7109, pp. 251–265. Springer (2011)
7. Buss, S.: Alogtime algorithms for tree isomorphism, comparison, and canonization. In: Kurt Gödel Colloquium, *Lecture Notes in Computer Science*, vol. 1289, pp. 18–33. Springer (1997)
8. Czyzowicz, J., Gasieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theoretical Computer Science **410**(6-7), 481–499 (2009)
9. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering of six robots on anonymous symmetric rings. In: Proceedings of the 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 6796, pp. 174–185 (2011)
10. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering of robots on anonymous grids without multiplicity detection. In: Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 7355, pp. 327–338 (2012)
11. D'Angelo, G., Di Stefano, G., Navarra, A.: How to gather asynchronous oblivious robots on anonymous rings. In: Proceedings of the 26th International Symposium on Distributed Computing (DISC), *Lecture Notes in Computer Science*, vol. 7611, pp. 330–344 (2012)
12. Devismes, S., Petit, F., Tixeuil, S.: Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In: Proceedings of the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 5869, pp. 195–208 (2009)
13. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing without communicating: Ring exploration by asynchronous oblivious robots. Algorithmica. To appear.
14. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Remembering without memory: Tree exploration by asynchronous oblivious robots. Theoretical Computer Science **411**(14-15), 1583–1598 (2010)
15. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous robots with limited visibility. Theoretical Computer Science **337**, 147–168 (2005)
16. Haba, K., Izumi, T., Katayama, Y., Inuzuka, N., Wada, K.: On gathering problem in a ring for 2n autonomous mobile robots. In: Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), poster (2008)

17. Izumi, T., Izumi, T., Kamei, S., Ooshita, F.: Mobile robots gathering algorithm with local weak multiplicity in rings. In: Proceedings of the 17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 6058, pp. 101–113 (2010)
18. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S.: Asynchronous mobile robot gathering from symmetric configurations. In: Proceedings of the 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 6796, pp. 150–161 (2011)
19. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S.: 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), vol. 7464, pp. 542–553. Springer-Verlag (2012)
20. Kempkes, B., Meyer auf der Heide, F.: Continuous local strategies for robotic formation problems. In: Proceedings of the 11th International Symposium on Experimental Algorithms (SEA), *Lecture Notes in Computer Science*, vol. 7276, pp. 9–17. Springer-Verlag (2012)
21. Klasing, R., Kosowski, A., Navarra, A.: Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. Theoretical Computer Science **411**, 3235–3246 (2010)
22. Klasing, R., Markou, E., Pelc, A.: Gathering asynchronous oblivious mobile robots in a ring. Theoretical Computer Science **390**, 27–39 (2008)
23. Koren, M.: Gathering small number of mobile asynchronous robots on ring. Zeszyty Naukowe Wydzialu ETI Politechniki Gdanskiej. Technologie Informacyjne **18**, 325–331 (2010)
24. Kranakis, E., Krizanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. Morgan & Claypool (2010)
25. Kranakis, E., Krizanc, D., Rajsbaum, S.: Mobile agent rendezvous: a survey. In: Proceedings of the 13th International Colloquium on Structural Information and Communication Complexity (SIROCCO), *Lecture Notes in Computer Science*, vol. 4056, pp. 1–9. Springer-Verlag (2006)
26. Pelc, A.: Deterministic rendezvous in networks: A comprehensive survey. Networks **59**(3), 331–347 (2012)
27. Prencipe, G.: Impossibility of gathering by a set of autonomous mobile robots. Theoretical Computer Science **384**, 222–231 (2007)
28. Santoro, N.: Design and Analysis of Distributed Algorithms. John Wiley & Sons (2007)
29. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999)